ASPECTS OF AN OPEN ARCHITECTURE ROBOT CONTROLLER AND ITS INTEGRATION WITH A STEREO VISION SENSOR

by

Nongji Chen

Thesis submitted for the degree of Doctor of Philosophy

November 1994
ABSTRACT

The work presented in this thesis attempts to improve the performance of industrial robot systems in a flexible manufacturing environment by addressing a number of issues related to external sensory feedback and sensor integration, robot kinematic positioning accuracy, and robot dynamic control performance. To provide a powerful control algorithm environment and the support for external sensor integration, a transputer based open architecture robot controller is developed. It features high computational power, user accessibility at various robot control levels and external sensor integration capability. Additionally, an on-line trajectory adaptation scheme is devised and implemented in the open architecture robot controller, enabling a real-time trajectory alteration of robot motion to be achieved in response to external sensory feedback. An in depth discussion is presented on integrating a stereo vision sensor with the robot controller to perform external sensor guided robot operations. Key issues for such a vision based robot system are precise synchronisation between the vision system and the robot controller, and correct target position prediction to counteract the inherent time delay in image processing. These were successfully addressed in a demonstrator system based on a Puma robot.

Efforts have also been made to improve the Puma robot kinematic and dynamic performance. A simple, effective, on-line algorithm is developed for solving the inverse kinematics problem of a calibrated industrial robot to improve robot positioning accuracy. On the dynamic control aspect, a robust adaptive robot tracking control algorithm is derived that has an improved performance compared to a conventional PID controller as well as exhibiting relatively modest computational complexity.

Experiments have been carried out to validate the open architecture robot controller and demonstrate the performance of the inverse kinematics algorithm, the adaptive servo control algorithm, and the on-line trajectory generation. By integrating the open architecture robot controller with a stereo vision sensor system, robot visual guidance has been achieved with experimental results showing that the integrated system is capable of detecting, tracking and intercepting random objects moving in 3D trajectory at a velocity up to 40mm/s.

PhD. 1994 Nongji Chen
ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Graham Parker for his constant support, encouragement, guidance and invaluable advice throughout the course of this study. Without his help in all matters this work would never have been possible.

Thanks are also due to the members of the Mechatronic Systems and Robotics Research Group, in particular Mr Shaun Lawson, Mr Gary Bonser and Miss Angela Kaplish, for their assistance and cooperation, as well as undertaking many fruitful technical discussions. It is a great pleasure for me to work in such a friendly and pleasant environment, and I can not express enough my appreciation for their generous encouragement and help during this course. I can only offer them my sincere thanks here, in return for their patience and precious time.

My very special thanks goes to Dr John Pretlove with whom I have worked closely during this course. His development of a stereo vision system, technical assistance, cooperation and constant encouragement have made the work described in this thesis possible.

I would also like to thank my family for their support, understanding and patience.

Finally, the financial support from Engineering and Physical Science Research Council is gratefully acknowledged.
ASPECTS OF AN OPEN ARCHITECTURE ROBOT CONTROLLER AND ITS INTEGRATION WITH A STEREO VISION SENSOR

TABLE OF CONTENTS

Abstract............................................................................................................................ii
Acknowledgement........................................................................................................iii
Table of Contents..........................................................................................................iii
List of Figures................................................................................................................v

CHAPTERS

I. Introduction...............................................................................................................1
  1.1 Automated manufacturing...................................................................................1
  1.2 Robot control............................................................................................................7
    1.2.1 Forward and inverse kinematics.................................................................8
    1.2.2 Trajectory planning and generation.........................................................10
    1.2.3 Robot dynamic control.............................................................................13
  1.3 External sensors......................................................................................................15
    1.3.1 Robot vision...............................................................................................16
    1.3.2 Other sensors..............................................................................................18
    1.3.3 External sensor interfacing......................................................................20
  1.4 Summary.................................................................................................................22
  1.5 Outline of study.....................................................................................................23
  1.6 Organisation of the thesis....................................................................................25

II. A review of robot controller architecture and robot control..........................28
  2.1 Introduction............................................................................................................28
  2.2 Robot controller architectures...........................................................................29
  2.3 Kinematic models and inverse kinematics algorithms.................................39
    2.3.1 Robot Kinematic Models........................................................................40
    2.3.2 Inverse kinematics algorithms..................................................................44
  2.4 Dynamic models and control algorithms........................................................50
    2.4.1 Dynamic models of a robot......................................................................51
    2.4.2 Control algorithms....................................................................................53
  2.5 Trajectory generation...........................................................................................59
  2.6 Summary................................................................................................................62

III. Design and development of an open architecture robot controller ..........65
  3.1 Introduction............................................................................................................65
  3.2 System design considerations............................................................................66
    3.2.1 Computational requirement......................................................................67
    3.2.2 Joint actuator position measurement..........................................................68

PhD. 1994 Nongji Chen
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potentiometers</td>
<td>69</td>
</tr>
<tr>
<td>Incremental encoders</td>
<td>70</td>
</tr>
<tr>
<td>3.2.3 Driving the joint actuator</td>
<td>72</td>
</tr>
<tr>
<td>3.2.4 Interface to a Unimation Mark III controller</td>
<td>73</td>
</tr>
<tr>
<td>3.2.5 Summary</td>
<td>75</td>
</tr>
<tr>
<td>3.3 Hardware system</td>
<td>76</td>
</tr>
<tr>
<td>3.3.1 Architecture of the controller</td>
<td>76</td>
</tr>
<tr>
<td>3.3.2 Interface design and implementation</td>
<td>80</td>
</tr>
<tr>
<td>The encoder interface board</td>
<td>80</td>
</tr>
<tr>
<td>The ADC input board</td>
<td>83</td>
</tr>
<tr>
<td>The DAC output board</td>
<td>84</td>
</tr>
<tr>
<td>The digital input/output board</td>
<td>85</td>
</tr>
<tr>
<td>The Unimation controller switch-interface</td>
<td>85</td>
</tr>
<tr>
<td>3.3.3 System integration</td>
<td>88</td>
</tr>
<tr>
<td>3.4 Software system</td>
<td>89</td>
</tr>
<tr>
<td>3.4.1 Software hierarchy in a robot controller</td>
<td>90</td>
</tr>
<tr>
<td>3.4.2 Overview of the controller software implementation</td>
<td>91</td>
</tr>
<tr>
<td>3.4.3 The device driver</td>
<td>94</td>
</tr>
<tr>
<td>Servo control synchronisation</td>
<td>96</td>
</tr>
<tr>
<td>3.5 Summary</td>
<td>97</td>
</tr>
<tr>
<td>IV. Inverse kinematics solution to a calibrated Puma 560 robot</td>
<td>99</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>99</td>
</tr>
<tr>
<td>4.2 Nominal kinematic model and its inverse solution</td>
<td>100</td>
</tr>
<tr>
<td>4.3 Inverse kinematics solution to a calibrated Puma 560 robot</td>
<td>107</td>
</tr>
<tr>
<td>4.3.1 The calibrated kinematic model</td>
<td>107</td>
</tr>
<tr>
<td>4.3.2 The inversion algorithm</td>
<td>108</td>
</tr>
<tr>
<td>4.3.3. Model conversion</td>
<td>112</td>
</tr>
<tr>
<td>4.3.4 Numerical examples</td>
<td>118</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>120</td>
</tr>
<tr>
<td>V. Robust adaptive tracking control of robot manipulators</td>
<td>122</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>122</td>
</tr>
<tr>
<td>5.2 Preliminaries</td>
<td>124</td>
</tr>
<tr>
<td>5.3 A robust adaptive control algorithm</td>
<td>126</td>
</tr>
<tr>
<td>5.3.1 The robust adaptive control algorithm</td>
<td>127</td>
</tr>
<tr>
<td>5.3.2 Discussion</td>
<td>136</td>
</tr>
<tr>
<td>5.4 Simulation Study</td>
<td>137</td>
</tr>
<tr>
<td>5.5 Summary</td>
<td>143</td>
</tr>
<tr>
<td>VI. On-line trajectory adaptation and vision guidance</td>
<td>144</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>144</td>
</tr>
<tr>
<td>6.2 Trajectory planning and On-line alteration strategy</td>
<td>147</td>
</tr>
<tr>
<td>6.2.1 Trajectory planning</td>
<td>147</td>
</tr>
<tr>
<td>6.2.2 On-line trajectory alteration</td>
<td>152</td>
</tr>
<tr>
<td>Trajectory alteration: joint or translational element</td>
<td>153</td>
</tr>
<tr>
<td>Trajectory alteration: rotational element</td>
<td>157</td>
</tr>
</tbody>
</table>

PhD. 1994 Nongji Chen
### Contents

6.3 The Surrey stereo vision system and robot visual guidance...
   6.3.1 The Surrey active stereo vision system...
   The Surrey active stereo vision head...
   The vision processing system...
   6.3.2 Robot visual guidance...
   Static guidance...
   Dynamic guidance...
   Target motion prediction techniques...
6.4 System integration issues...
6.5 Summary...

VII. Experiments

7.1 Introduction...
7.2 The ISO pose accuracy tests...
   7.2.1 Objectives...
   7.2.2 Description of the experiments...
   7.2.3 Results...
   7.2.4 Conclusions...
7.3 Servo control tests...
   7.3.1 Objectives...
   7.3.2 Description of the experiments...
   7.3.3 Results...
   7.3.4 Conclusions...
7.4 Static visual guidance tests...
   7.4.1 Objectives...
   7.4.2 Description of the experiments...
   7.4.3 Results...
   7.4.4 Conclusions...
7.5 Dynamic visual guidance test...
   7.5.1 Objectives...
   7.5.2 Description of the experiment...
   7.5.3 Results...
   7.5.4 Conclusions...
7.6 Summary...

VIII. Conclusions

8.1 Conclusions...
8.2 Suggestions for future research...

Bibliography...

List of contributions...

Appendix A  Incremental encoder start-up resetting...
LIST OF FIGURES

Figure 1.1.1 Relative cost-effectiveness of soft automation...............................3
Figure 1.2.1 Forward and inverse kinematics ........................................................9
Figure 1.2.2 Description of trajectory planning.....................................................11
Figure 1.2.3 Trajectory in joint space for a pick-and-place operation...............13
Figure 1.2.4 Illustration of robot dynamic control ...............................................15
Figure 1.3.1 Robot operation command levels ......................................................22
Figure 2.2.1 Architecture of Unimation Mark III robot controller ..................30
Figure 2.2.2 Controller architecture proposed by Nigam and Lee ..................32
Figure 2.2.3 Architecture of the SIERA system .....................................................33
Figure 2.2.4 The architecture for RAL hand and manipulator control ..........34
Figure 2.2.5 Architecture of the CONDOR system ...............................................35
Figure 2.2.6 Hardware structure of the SPARTA system .....................................37
Figure 2.3.1 DH model definition ............................................................................41
Figure 2.3.2 Definition of n, o, a vectors in robot end-effector (tool) ..........43
Figure 2.3.3 Definition of arm and elbow configuration indicators ...............46
Figure 2.4.1 Linear control with fuzzy logic adaptation ......................................59
Figure 3.2.1 A servo motor based joint configuration ........................................68
Figure 3.2.2 Incremental encoder signals ...............................................................70
Figure 3.3.1 Configuration of the open architecture robot controller .............76
Figure 3.3.2 Incremental encoder interface board block diagram .................81
Figure 3.3.3 A picture of the in-house made encoder interface board .........83
Figure 3.3.4 Functional description of the switch-interface ...............................86
Figure 3.3.5 The switch-interface in the Unimation controller chassis ..........87
Figure 3.3.6. Illustration of the robot control platform .......................................88
Figure 3.4.1 Software hierarchy of a robot controller ....................................91
Figure 3.4.2 Description of the controller software ..............................................92
Figure 3.4.3 The extended device driver ...............................................................95
Figure 3.4.4 Servo control synchronisation interaction .....................................96
Figure 4.2.1 Link coordinate system and parameters of a Puma robot .....100
Figure 4.3.1. Illustration of a2, d2, q2 ...................................................................113
Figure 4.3.2. Geometric relationship between frames .......................................114
Figure 5.1.1 Closed-loop robot control layout .......................................................122
Figure 5.4.1 A two-link planar articulated robotic arm ......................................138
Figure 5.4.2 Joint 1 trajectory tracking results of the proposed controller .........139
Figure 5.4.3 Tracking error comparison .................................................................140
Figure 5.4.4 Tracking error comparison when the signal frequency doubled .................................................................................................... 140
Figure 6.2.1 The basic speed ramp profile in trajectory planner .......... 148
Figure 6.2.2 Illustration of trajectory planning with no path alteration ........................................................................................................... 152
Figure 6.2.3 On-line trajectory alteration ................................................ 153
Figure 6.2.4 Illustration of one dimensional trajectory alteration .......... 154
Figure 6.3.1 A photograph of the Surrey Active Stereo Head .................... 161
Figure 6.3.2 Overview of the Surrey stereo vision system hardware............. 164
Figure 6.3.3 Static vision-based robot guidance procedure ......................... 166
Figure 6.3.4 Dynamic vision-based robot guidance procedure ..................... 169
Figure 7.1.2 A photograph of the experimental set-up ................................. 177
Figure 7.4.1 The accuracy results for single and two shot robot guidance ........................................................................................................... 190
Figure A.1 Start-up resetting signals for rotary incremental encoders ........ 213
Figure A.2 Start-up resetting for an encoder via a potentiometer ............... 215
1.1 Automated manufacturing

Achieving higher productivity and efficiency has been widely viewed as a crucial factor for improving the competitiveness of manufacturing industry. Since the early years of this century, specialised machines have been designed and developed for high-volume automated production lines manufacturing mechanical and electrical parts. These special-purpose machines are very efficient in performing the predetermined functions in a manufacturing process. However, when each production cycle ends and new models of the parts are to be introduced, the specialised machines have to be shut down and the hardware retooled or even the whole machine replaced to accommodate the newer models. The inflexibility and generally high cost of these machines, often called hard automation systems, have led to a broad-based interest in the use of more flexible machines in a manufacturing environment.

To alleviate the inconvenience of the continued use of hard automation systems, the automobile industry and other industries have introduced more flexible forms of automation in the manufacturing cycle. Programmable mechanical manipulators are used to perform such tasks as spot and arc welding, spray painting, material handling, machining, and component assembly. More recently, flexible manufacturing systems (FMS), in which robot systems have been widely regarded as playing a significant role, have been considered as the solution to satisfy the twin goals of lowering the manufacturing costs through increased productivity and
efficiency and maintaining flexibility. An FMS is a functionally reprogrammable system that comprises specialised flexible manufacturing cells (FMC) and transport systems. Each FMC may further comprise a variety of machining and robot systems for various specific operations, such as assembly, routing, milling, drilling, grinding and inspection. Within an FMS, pieceparts are transported from one FMC to another for machining or processing. The transportation can be achieved by using automated guided vehicles (AGVs) or conveyer belt systems. For tasks where manipulation of the pieceparts is required, a robot system would provide the all-purpose tool of maximum flexibility.

The programmable feature of these manipulators and FMSs contrasts with the hard automation systems in that, by changing the software, the computer controlled manufacturing equipment can be easily converted to do a variety of tasks. Such flexibility in automated manufacturing production lines has been referred to as soft automation. A qualitative illustration of where soft automation can be justified in terms of cost-
effectiveness is shown in figure 1.1.1. The figure compares the cost effectiveness of manual labour, hard automation, and soft automation as a function of the production volume. The unit cost for manual labour is almost a constant line parallel to the production volume axis. It is only cost-effective when the production volume is too small to be manufactured by an automated production line. As the production volume increases, there comes a point \( v_1 \) where the soft automation based manufacturing system becomes most cost-effective. As the production volume increases still further, it eventually reaches a point \( v_2 \) where hard automation takes the lead in cost-effectiveness. The curves in figure 1.1.1 are representative of general qualitative trends, with the exact data dependent on the characteristics of the unit being produced. As the soft automation equipment become more sophisticated or less expensive, the corresponding cost-effective range shown in figure 1.1.1 continues to expand at both ends over the production spectrum.

The force behind the drive for soft automation lies in the change of market needs. The now established trend of higher customer demands on suppliers is not a new one but it is becoming increasingly more imperative and encompassing. For many traditional single-design products the days of mass production to achieve low-cost manufacture for the market are gone. The greater numbers of competing producers for product types means that manufacturers must cater for the customers' tastes to sell their products. Increased competition since the late 1960s has generally resulted in decreasing manufacturing life cycle time of end-products and their parts. Consequently, many manufacturers face the challenge of customer demands for:

- The highest quality and precision.
- Wide product variety.
• Frequent product design changes.
• Variable batch sizes.
• Short delivery times.
• Competitive prices.

The cost-effective way to meet the above requirements is to adopt soft automation in production lines. This is especially true if small batch production and mixed product types are frequently requested. The flexibility provided by robot systems and FMSs in these production lines means that the preparation work for manufacturing a new product is minimal and is mainly oriented to the change of operational software. Indeed, with the help of off-line programming (OLP) techniques, the change of the operational software for the next type product can be undertaken while the production line is in full operation with the current type product being manufactured.

Limited success has been achieved by using industrial robot systems in a flexible manufacturing environment. The problems stem from the fact that, in general, current industrial robots lack both sophisticated external sensing capability and high accuracy. Conventional applications of industrial robots are implemented through the manual teach-and-repeat approach. By doing so it relies more on a robot's repeatability rather than its accuracy. The teach-mode approach is quite time consuming and requires a skilled operator to produce a usable program. It also requires that the production line is shut down to allow robots to be taught whenever their environments are modified. This is increasingly unacceptable due to the high cost of production down-time in small and medium batch size applications. Although the off-line programming technique offers the ideal solution to such a problem, the actual implementation of it has suffered from the fact that, in general, current industrial robots exhibit good repeatability but poor accuracy. In the teaching method, the accuracy issue
does not cause serious problems since the operator ensures that the end-effector is in the appropriate pose at each task point. The conversion from task space to joint space is therefore made at each key task point with a visual confirmation by the operator. Since the key task points have been previously taught, the ability of the robot to attain these poses is governed by the manipulator's repeatability. On the other hand, off-line programming relies on the assumption that a correct joint space description of a pose may be determined from the task space description. It requires that the model used for generating the robot programs in a computer aided design (CAD) environment matches with the real world. In other words, a task point defined by the off-line generated program actually must be reached by the robot that executes the program. How well this can be done is determined by the accuracy of the robot under question.

Using a robot without sensors requires a well conditioned application environment. Such systems require highly accurate placement of the piecepart in the workcell. Any unforeseen events or occurrences such as variations in the workcell or imprecision and imperfections in the pieceparts cannot be handled by the robot system. To relax the constraints for a more realistic environment, a means of attaining information about the workcell is necessary and the robot controller should be able to react sensibly to this workcell feedback sensory information.

Attempts to improve robot performance have been undertaken in several aspects. To better use the higher repeatability, robot calibration techniques have been developed to improve the accuracy of current industrial robots. Kinematic calibration of robotic manipulators attempts to overcome the discrepancies between the real manipulator static performance and that predicted by the nominal kinematic model. The objective is to identify the 'real' kinematic model that makes it possible for
the robot to be as accurate as it is repeatable (Mooring et al., 1990). Traditionally, the convention developed by Denavit and Hartenburg (1965), known as the DH model, has been used to describe a robot’s kinematic structure. This model uses 4 parameters, the minimum number required, to describe the relationship between consecutive links in a robot. Despite its simplicity and popularity in modelling robot kinematics, the DH model is widely considered inappropriate to be used in a calibration procedure due to a number of problems stemmed from the fact that the link coordinate frames under DH convention are located at the intersection of the joint axis and the common normal (Mooring, 1990). These problems include: a) selection of the base frame is not arbitrary; b) the ‘zero position’ of the manipulator is not arbitrary; and c) model parameters vary by large amounts for revolute joints with nearly parallel axes. To overcome these problems, a variety of alternative models have been developed which are well suited for calibration purpose (Stanton, 1991). In general, these models are complex and use more than 4 parameters to describe the coordinate frame relationship between consecutive links. For example, the modified S-model, developed initially by Stone et al (1986) and then improved by Stanton (1991), has 6 parameters. It allows arbitrary positioning and orientation of the link frame on the joint axis which enables the robot link parameters to be identified accurately by a set of simple decoupled identification problems. Typical improvements of as much as 12 times in a robot’s absolute positioning accuracy have been reported after the robot is calibrated with the modified S-model description (Stanton and Parker, 1992).

A further focal point for improving the performance of a robot system is to incorporate sophisticated control algorithms. It is widely recognised that the current industrial practice of using a simple PID control algorithm to control each manipulator joint is not able to provide satisfactory dynamic performance over the whole workcell, especially when the robot is moving...
at a higher speed. Future high performance robot systems are expected to be equipped with sophisticated control strategies that offer better trajectory tracking capability and transitional performance.

Another major effort, perhaps is more fundamental, to improve the robot performance is to employ external sensors. The external sensors provide feedback information about the workcell which is used by the robot control system to adapt the planned operations. Its integration with a robot provides effective solutions, at least in static sense, to many difficult problems including overcoming the kinematic modelling errors and the effects of backlash, compliance in the joints and flexibility of the link structure. Indeed, as the robot system and its workcell now constitutes a feedback loop, it implies that any sort of static error and uncertainty involved in the robot operations can be compensated.

1.2 Robot control

The flexibility and efficiency provided by robots in a manufacturing environment is largely determined by the functionality of robot controllers, within the limits set by the design of the mechanical structure. A robot controller system is normally built upon a specially tailored computing machine. Its commonest task by far is to drive the robot arm to the demanded target positions or to follow some pre-specified paths with satisfactory transitional and static accuracy and fulfil the demanded gripper operations. Additional tasks may include velocity control and force control.

Control of a robot involves the operations that perform different space description conversions, establish the desired motion trace in a specified space and actually effect the robot motion. This is a complex computing activity that requires performing multi-tasks on-line. The major functional parts related to a robot task operation can be broadly viewed as comprising
kinematic calculation, trajectory generation and dynamic control to achieve the required operation.

1.2.1 Forward and inverse kinematics

Robot kinematics deals with the geometry of motion of a robot arm with respect to a fixed reference co-ordinate system without regard to the force/moments that cause the motion. The objective is to control both the position and the orientation of the end-effector in three dimensional space. In order to program the robot motion, the relationship between the joint variables and the position and the orientation of the end-effector must firstly be established.

The position and orientation of the end-effector is determined by the joint variables through the mechanical structure of the manipulator. A set of mathematical equations can be established to describe this relationship. It has commonly been referred to as the forward kinematic model or kinematic model. The forward kinematic model maps an eligible joint vector into its corresponding position and orientation of the end-effector, or, in another phrase, the robot pose. It provides the relationship which explicitly shows the dependence of the end-effector configuration on the joint variables. This can be utilised, for example, in determining the size and shape of the work envelope.

The most important benefit provided by the solution of the forward kinematics problem is that it lays a foundation for solving a related important problem, the inverse kinematics problem. In many cases, robot tasks are described in the workcell space (Cartesian space). This is the most natural way that a human operator can perceive the motion and operation of the robot. It is also convenient for incorporating the information instrumented by external sensors which provide the workcell feedback.
However, since the actual control of a robot's movement is carried out in the joint space, the conversion from the Cartesian space to the joint space is inevitable in a robot application. This mapping from the Cartesian space to a robot's joint space is referred to as the inverse kinematics problem. A schematic that illustrates the relationship between the forward kinematics and the inverse kinematics is shown in figure 1.2.1.

![Figure 1.2.1 Forward and inverse kinematics.](image)

The inverse kinematics problem is in general more difficult than the forward kinematics problem. There is no single explicit systematic procedure that offers the solution to the inverse kinematics problem for all robots. As a result, each robot or generically similar class of robots has to be treated separately. Furthermore, unlike the forward kinematics problem, which gives a unique answer to a given set of joint variables, the inverse kinematics problem is ill-defined because it normally has multiple solutions for a given position and orientation (Craig, 1986; Fu et al, 1987). Therefore, extra constraints are needed to uniquely define an inverse kinematics solution for a given robot pose.

Finding the inverse kinematics solution for a given desired robot pose is an important issue in implementing robot control. Most of today's
industrial robots, due to their special nominal mechanical configurations, have a closed form solution to their inverse kinematics problem. However, this solution is only valid in a nominal sense. If a robot has a discrepancy between its nominal kinematic model and its 'true' mechanical configuration, a pose error will exist when the control is based on an inverse kinematics solution to its nominal model. To improve the pose accuracy, a calibrated kinematic model should be used. However this will, in general, result in another difficulty: the closed form inverse kinematics solution is no longer available since the calibrated results normally do not bear the same simple mechanical configuration features as the nominal model has (Judd and Knasinski, 1987; Broderie and Cipra, 1988). Numerical methods have to be adopted to solve the inverse kinematics problem under these circumstances. Thus the development of real-time applicable numerical methods to the inverse kinematics problem becomes an important issue given that most of current general numerical algorithms are computationally too time consuming to be used in real-time applications (Wang and Chen, 1991).

1.2.2 Trajectory planning and generation

In robot applications it is frequently necessary for the manipulator to move between goal points in a smooth, controlled fashion. This has generally been achieved, in practice, by causing each joint of the robot arm to move in accordance to a specified smooth function of time. Commonly, the beginning and end of each joint's motion are arranged by the robot controller at the same time instant respectively, so the manipulator motion appears co-ordinated. Exactly how these motion functions are generated is termed as the problem of trajectory planning or trajectory generation (Paul, 1981; Craig, 1986).
Moving from an initial location (position and orientation) to a final destination, the end-effector of a robot traverses a space curve. This space curve is called a path. It is determined by the operational requirement of the task that the robot is performing. A path is a purely spatial representation. But it becomes a trajectory when temporal information is superimposed by specifying the times at which the end-effector must be at various points of the path. A functional description of trajectory planning is given in figure 1.2.2.

![Diagram of trajectory planning](image)

**Figure 1.2.2** Description of trajectory planning

Trajectory planning schemes generally "interpolate" or "approximate" the desired path by a class of polynomial functions and generate a sequence of time-based "control set points" for the control of the robot from the initial location to its destination. It can be conducted either in the joint space or in the Cartesian space depending on the operational requirement. For joint space planning, the time history of all joint variables and their speeds and accelerations are planned to describe the desired motion of the robot. For Cartesian space planning, the time history of the end-effector's position, velocity and acceleration are planned, and the corresponding joint positions, velocities and accelerations are derived from the planned Cartesian information to actually affect the robot motion. Compared to the Cartesian space planning scheme, planning in the joint space is simple,
straight forward and computationally efficient. It suffers, however, from the fact that the end-effector's trace in the Cartesian space becomes very complex. Thus it is inappropriate for some applications where the path of the end-effector (tool) is required to follow a specified Cartesian space curve, such as arc welding, obstacle avoidance, etc.

The most basic robotic manipulation motion type perhaps is the pick-and-place motion. Such a motion type is needed, for example, in the automated loading and unloading of machines. More generally, pick-and-place motions are used to alter the distribution of parts within the workspace. This type of motion control usually only needs simple point-to-point smooth trajectory with a few constraints on the lift-off and set-down positions. Figure 1.2.3 gives an illustrative description of the joint-based trajectory for the $i$th joint of a robot that is performing a pick-and-place task.

![Figure 1.2.3 Trajectory in joint space for a pick-and-place operation.](image)

PhD. 1994 Nongji Chen
Chapter 1: Introduction

The lift-off and set down points in figure 1.2.3 provide the directed motions of the robot at the starting position and the final reach position respectively. This is generally required to guide the robot to move in the correct direction for departure and approach. The operational speed of the above trajectory is reflected by the time points $t_1$, $t_2$ and $t_f$. They are determined in the planning stage in accordance to the given motion speed profile or explicitly specified by the operational requirement.

A more complex and computationally expensive trajectory planning task is the continuous-path motion in Cartesian space. This motion type is widely required for arc welding, paint spraying, sealing, machining, etc. It also provides convenience for interactions with external sensors since the workcell feedback is in general Cartesian space based.

If a desired motion path of a manipulator operation is known prior to the actual operation taking place, the task of trajectory planning can be carried out off-line (Yamamoto et al., 1988; Park & Lee, 1992). A number of complex planning algorithms can then be employed to search for an optimal solution under the given constraints as there is no runtime restrictions. This, however, is not applicable to more general application scenarios, especially when workcell feedback exists. For applications involving external sensors, such as vision guided target tracking, intercepting and docking, real-time trajectory generation and trajectory alteration are essential for performing the intended tasks (Koivo and Houshangi, 1991; Piccardo and Honderd, 1991).

1.2.3 Robot dynamic control

Trajectory information provides the desired motion for a robot to perform the intended tasks. To cause the manipulator to faithfully “track” or follow the planned trajectory, appropriate driving forces must be applied
to the manipulator's actuators. In industrial applications, very few robots use stepper motors or other actuators which can be controlled in an open loop fashion. The majority are powered by actuators which output a torque or a force at each joint. In order to realise the desired motion, the torque or force delivered by each actuator must be regulated by feedback to achieve an acceptable dynamic performance. This torque/force regulation issue is frequently referred to as the robot control problem.

![Diagram of robot dynamic control](image)

**Figure 1.2.4 Illustration of robot dynamic control**

High performance control of a manipulator is a difficult task because of the complex dynamic behaviour of a manipulator system. To describe the dynamic behaviour of a manipulator, some classical theories, such as Newtonian and Lagrangian mechanics, can be applied to establish the manipulator motion equations or, more commonly referred to, the dynamic model. These equations are useful for computer simulation of the robot arm motion, the design of suitable control algorithms for a robot arm, and the evaluation of the kinematic design and structure of a robot arm.
A more essential problem in the study of robot dynamics is to find the required joint torques which will cause the robot arm to move in accordance to the given motion trajectory, speed and acceleration. This is generally called the inverse dynamics problem. It is directly relevant to the aim of robot control. In fact, if a robot arm's dynamic model is accurately derived and its inverse problem can be solved with accuracy and efficiency (i.e., in real-time), optimal control performances can then be achieved. However this precise dynamics inversion is in general unavailable in a practical application, where many factors, which affect the robot arm's dynamic behaviour, have been excluded from or simplified in the mathematical dynamic model so that the resultant model can be described within a manageable form. In addition, the computational burden of the inverse dynamics evaluation is enormous and is generally out of reach of current commercial robot controllers (Graham, 1989).

Conventional industrial robot controllers make use of simple linear control algorithms which ignore the coupling and nonlinear nature of the dynamics of a manipulator and thus treat each link as a separate decoupled system. Such an approach has been widely recognised as not being able to provide satisfactory performance over the whole workcell, especially when the robot is moving at a higher speed (Fu, Gonzalez & Lee, 1987). Further robot performance improvement requires advanced control algorithms that take the complex dynamic behaviour of a manipulator into consideration (Hsia, 1986; Ortega and Spong, 1988, Abdallah et al., 1991). A recent trend of research, in handling the robot dynamic control issue, is to adopt a more sophisticated control strategy such as adaptive control, robust control, variable structure control, etc. In general, the implementation of more advanced control algorithms requires considerable computational power to support real-time control operations.
1.3 External sensors

A key to making robots more versatile in a flexible manufacturing environment lies in using feedback from external sensors. In general, sensors used in a robot system may be divided into two principal categories: internal state sensors and external state sensors. Internal state sensors deal with the detection of variables such as arm joint positions. They are the basic information sources for low-level robot motion control. External state sensors, on the other hand, provide environmental information such as the part location and part orientation in a robot workcell. Their incorporation provides a workcell feedback channel that makes it possible for the robot to adapt its operations to environment variations.

Many types of sensors can be used to serve as external sensors for robot applications. They cover the fields of range, proximity, touch and force sensing. The selection of what type of sensors should be used is governed by the task or function it must carry out. For example, if a task requires the robot system to avoid collisions, then some sort of proximity sensors should be used. And, perhaps more sensibly, an additional gross range sensor should also be employed to provide a precautionary 'alarming signal' so that the robot is well prepared (say with speed reduced) for a potential collision danger.

1.3.1 Robot vision

Among the most commonly encountered external sensors, machine vision has been recognised as the most general and powerful robot external sensory technique (Fu, Gonzalez & Lee, 1987; Schilling, 1990). It supplies valuable information that can be used to automate the manipulation of objects. With the use of a vision system, the position, orientation, identity, and condition of each part in the field of view can be obtained. This high
level information can then be used to plan robot operations such as how to grasp a part, how to avoid collisions with obstacles and how to intercept a moving target.

A basic robot vision system has a single stationary camera mounted over the workspace. In this case, the deductions about the geometric properties of objects within the field of view can only be made under the condition that the world model is sufficiently constrained. The problems with single camera machine vision are the difficulties in handling real-world irregularities such as reflections, shadows and specular highlights, the severe constraints placed upon reasoning about 3D objects from 2D image data and the inability to cope with non-prismatic and curved objects.

Stereo vision represents a more general class of deriving 3D information by computer vision. The method relies on correctly matching the corresponding points in two (or more) views from different perspectives of the same scene. Such a feature point match issue is fundamental in recovering the depth information and is termed the stereo correspondence problem; for which, in general, both area-matching and feature-matching techniques can be employed. However, when the scene contains distinct features, such as prominent corners, a feature-matching approach will generally yield a faster and more robust solution.

Machine vision can be achieved by a fixed configuration or using a dynamic approach, which is more flexible and has been referred to as active vision. Active vision is an emerging technique that has been variously referred to as smart sensing (Burt, 1988), attentive vision (Clark and Ferrier, 1988), active perception (Bajcsy, 1988), purposive vision (Pahlavan and Eklundh, 1991), animate vision (Ballard, 1991) and reactive vision (Sharkey et al., 1992). Images available to a vision system from a camera are
Chapter 1: Introduction

characterised by far more data than can be analysed by existing vision processing systems in real-time. The information that is important for the particular task tends to be episodic (gathered in space and time) and is often surrounded by much information that is redundant. Such features in vision sensing can not be exploited by fixed configuration vision systems which treat the whole working space in a uniform manner. On the contrary, active vision aims to overcome these problems by focusing attention and sparse computational resources on the critical regions, ignoring the irrelevant data. This helps to ease the computational burden and to achieve the objectives of reliable extraction of information within certain time constraints.

The advantages that active vision offer include the ability to overcome a limited field of view offered by a fixed configuration camera, to increase the spatial resolution of the vision system by being able to examine the full visual field and by reducing the computational burden by selecting portions of the scene containing potentially interesting features. Other advantages include the ability to stabilise the images, aiding motion estimation, figure-ground separation, better range estimates fused from stereo, focusing and sensor geometry, and lessening the effects of occlusions.

Mounting an active vision sensor on the end-effector of an industrial robot has several distinct advantages over fixed camera configurations. The first is that it utilises the flexibility of the robot in providing the six degrees-of-freedom necessary to accomplish 3D positioning and orientation of the active device. The sensor can also be moved over the entire workcell avoiding difficulties of obscured views to give a complete view of the workcell. The active vision sensor can also be brought closer to the various parts of the workcell to allow higher resolution images to be obtained.
1.3.2 Other sensors

Depending upon a particular application, external sensors other than machine vision are frequently required to provide workcell feedback information. These sensors, however, do not offer the generality of machine vision; instead, they tend to be more specific in a particular functional aspect and their role in robot operations tend to be application dependent.

Table 1.3.1 Common robot external sensors or sensing techniques

<table>
<thead>
<tr>
<th>type</th>
<th>usage</th>
<th>working principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>laser</td>
<td>measuring</td>
<td>deriving from the elapsed time of laser light travel</td>
</tr>
<tr>
<td></td>
<td>range</td>
<td></td>
</tr>
<tr>
<td>ultrasonic</td>
<td>measuring</td>
<td>deriving from the elapsed time of ultrasonic wave travel</td>
</tr>
<tr>
<td></td>
<td>range</td>
<td></td>
</tr>
<tr>
<td>inductive</td>
<td>proximity</td>
<td>change of inductance when approaching a metallic object</td>
</tr>
<tr>
<td>sensor</td>
<td>sensing</td>
<td></td>
</tr>
<tr>
<td>capacitive</td>
<td>proximity</td>
<td>change of capacitance when approaching a surface object</td>
</tr>
<tr>
<td>sensor</td>
<td>sensing</td>
<td></td>
</tr>
<tr>
<td>Hall-effect</td>
<td>proximity</td>
<td>Hall-effect when approaching ferromagnetic materials</td>
</tr>
<tr>
<td>sensor</td>
<td>sensing</td>
<td></td>
</tr>
<tr>
<td>micro-switch</td>
<td>touch</td>
<td>state change of the switch</td>
</tr>
<tr>
<td>strain</td>
<td>force</td>
<td>change of resistance etc. when device strained</td>
</tr>
<tr>
<td>gauges</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are two basic categories of robot external sensors: contact sensors or noncontact sensors. As their name implies, the former class of sensors respond to physical contact, such as touch, slip and torque. Noncontact sensors rely on the response of a detector to variations in acoustic or
electromagnetic radiation, laser amplitude and phase changes, etc. Table 1.3.1 summarises some commonly encountered sensors or sensing techniques that can be employed in robot applications.

1.3.3 External sensor interfacing

The use of external sensing mechanisms allows a sensor-robot integrated system to interact with its environment in a flexible manner. Although external sensor-based robotics are seen as a key issue in advanced manufacturing technology (AMT) and flexible manufacturing systems (FMS), in general robots used in industry today lack robust external sensing capabilities and are thus not effective in adapting to minor imperfections in the task. This partly accounts for the fact that only limited success has been achieved by using robots in a flexible manufacturing environment and other related application areas. The difficulties of integrating an external sensor into a robot system lies not only with the sensor and the sensed information interpretation aspects, but also with how the robot controller should interact with the former to affect a successful robot response (Walters, 1993). Conventional industrial robot controllers are rather limited in their capacity for offering such support. They are commonly designed as self-contained controller systems and provide no means for users to alter their internal kinematic control and dynamic control loops. The lack of the ability of the current industrial robot controllers to effectively interact with external sensors except in a specialised manner has contributed significantly to the motivation in design and development of more openly user accessible robot controllers in a number of research institutes and universities. In general, these research related efforts have brought about a number of bus-based multi-processor robot controllers (Kazanzides et al., 1987; Kim et al., 1987; Narasimhan et al., 1989).
Chapter 1: Introduction

The capability of a sensor-based robotic system is dependent upon the level and quality of the sensory information that is provided to the controller. To take advantage of sensory information the robot controller and the sensory system must be integrated so that the workcell feedback can be utilised efficiently in supporting robot tasks. In general, four levels of operations can be classified according to the commands that affect a robot operation. Figure 1.3.1 illustrates this classification.

![Diagram](image)

**Figure 1.3.1 Robot operation command levels.**

Workcell feedback normally interacts at the lower levels of the operational hierarchy shown in figure 1.3.1. This is because at a higher level a command is more likely to be related to the job description rather than the workcell conditions. Exactly which level or levels will be involved in a sensor-based application is dependent on the sensor used and the anticipated tasks. For example, if a vision based guidance application is required, the interface is likely to be implemented at the manipulator level.
Chapter 1: Introduction

External sensor interactions at the joint level or even lower (in trajectory control or dynamic control) requires that the robot controller provides the user with the flexibility of altering the internal trajectory control and servo control operations. This is a severe challenge to current industrial robot controllers due to their real-time operation limitations. Also, as sensor based robotics is still in its infancy, no standards on sensor-controller interaction levels and protocols have been established. Indeed, as many novel sensors are still been developing, it is wise not to constrain too many of the interaction levels in order to leave sufficient user implementation freedom for accommodating new technology.

1.4 Summary

High performance robot systems are essential to reduce the difficulty in building up flexible manufacturing systems and in implementing highly desirable off-line-programming techniques. Industrial robots used today, in general, have only achieved limited success in attaining flexible manufacturing requirements. Attempts to improve robot performance have been made in the areas of robot calibration, external sensor integration and developing advanced dynamic control algorithms. Robot calibration techniques aim at overcoming some of the manufacturing errors of each robot by identifying the ‘true’ geometry or ‘signature’ of the manipulator. The resultant kinematic model is expected to provide system accuracy close to the repeatability characteristics.

A more fundamental issue in improving the robot performance is to employ external sensors. They provide feedback information about the workcell and promise effective solutions to the accuracy and repeatability problems and, further more, require a much less constrained environment
as the robot system is able to adapt its planned operations to the uncertainties existing in the workcell.

Improving the dynamic performance of a robot system is also attractive since it promises higher productivity through the increased operational speed. Current industrial robot controllers in general treat each joint as a simple linear system and ignore any coupling and nonlinear effects. This is widely regarded as being unable to provide satisfactory dynamic performance when the robot is moving at a higher speed.

The efforts to improve robot performance inevitably put forward challenges to current industrial robot controller systems as the corresponding outcomes require open access capabilities and sufficient computational power for real-time implementations. To take full advantage of these advanced techniques, open architecture robot controllers need to be employed.

1.5 Outline of study

This research work addresses a number of issues related to sensor-robot integration and robot control for improving the performance of industrial robots in a flexible manufacturing environment. It concentrates on: a) the design and development of an open architecture robot controller with sufficient computational power to support advanced robotic applications; b) the development of a simple and effective numerical algorithm that solves the inverse kinematics problem when a calibrated kinematic model is used; c) the development of a robust adaptive robot tracking control algorithm that promises high control performance as well as a relatively modest computational complexity; d) devising an on-line trajectory adaptation scheme to support robot workcell feedback; e) integrating the robot controller with a stereo vision sensor capable of being mounted on the end-
effector of a robot to achieve robot visual guidance. The work is primarily intended to establish an advanced robot control platform and, by integrating it with a robot vision sensor, to demonstrate the concept that an industrial robot system equipped with external sensors can 'intelligently' react to its environment. Such 'intelligence' is expected to significantly enhance the performance of robot systems in, for example, a flexible manufacturing environment as both uncertainties and unexpected events in the workcell can be dealt with.

The design and development work on the open architecture robot controller accounts for a significant portion of the undertaken research. It involves a considerable amount of hardware design, in-house making, interface boards assembly as well as testing. Another substantial task involved is the design and development of the robot controller software. These efforts have resulted in a robot controller featuring high computational power, user accessibility at various robot control levels to support the open architecture philosophy, and external sensor integration capability. The work has also led to a robot control platform that consists of the open architecture robot controller and a Puma 560 industrial robot, providing a much needed facility for conducting and validating various advanced researches and applications in robotics.

A significant effort has been made in this research work to achieve integration of external sensors with robot systems. Interfaces at three robot control levels, i.e. the servo control level, the trajectory control level and the command programming level, have been implemented in the open architecture robot controller to facilitate sensor-robot interactions. Additionally, an on-line trajectory adaptation scheme has been devised and implemented, which enables a robot system to rapidly alter its motion trajectory in response to external sensory feedback. As an example of such a
sensor-robot application case, the integration of a robot vision sensor with the developed robot control platform to achieve robot visual guidance is discussed in depth and the experimental results are described and evaluated.

Aspects of improving both robot kinematic and dynamic performance have also been investigated in this research work. Efforts have been made to establish a numerical algorithm that provides an on-line solution to the inverse kinematics problem of a calibrated kinematic model, and to develop a robust adaptive robot tracking control algorithm that provides higher dynamic performance without requiring practically prohibitive computational power as some other adaptive algorithms. Experiments on both the algorithms have been conducted to demonstrate the potential performance improvement on static positioning accuracy and dynamic control quality.

Throughout the course of this research, the emphasis has been placed upon finding practical solutions to various complex problems and upon the experimental validation of the proposed analytical techniques. This has resulted in an advanced real-time robot control platform being built which has enabled a range of experiments to be carried out. These experiments effectively demonstrate and validate the research work, which culminates in an experimental demonstrator that simulates an industrial robot workcell capable of tracking and intercepting randomly moving objects.

1.6 Organisation of the thesis

This thesis is organised as follows. In Chapter 2, robot controller architectures and various issues involved in realising robot control are reviewed. The review covers controller hardware configurations, bus structure, system expandability, robot kinematic models as well as their inverse kinematics solution problem, robot dynamics, servo control
Chapter 1: Introduction

techniques, and trajectory generation techniques. The emphasis has been placed on identifying the limitations of existing techniques and the areas in which further improvements can be made.

In Chapter 3 the design and development work of an open architecture robot controller has been detailed with particular emphasis placed upon the underlying design philosophy, the controller hardware configuration and the structure of the controller system software.

Chapter 4 presents an iterative numerical technique to resolve the inverse kinematics problem that is associated with a calibrated robot kinematic model. It also describes a model conversion technique that converts an S-model based calibration results of a Puma 560 robot to a suitable form which makes the proposed inverse kinematics algorithm directly applicable.

Chapter 5 addresses the issue of robot dynamic control. It proposes an adaptive robust tracking control algorithm which combines PD (Proportional and Differential) feedback control with adaptive compensation. The derived adaptive control algorithm is computationally inexpensive compared to other complicated adaptive control schemes. Numerical simulation results are included to demonstrate its performance.

Chapter 6 describes the techniques for robot motion trajectory adaptation and the integration of a vision sensor with the robot control platform to achieve robot visual guidance. It also discusses the interface and synchronisation issues with particular emphasis placed on the sensor-robot dynamic interactions in the dynamic guidance case.

A number of experiments has been conducted which are reported in Chapter 7. Each of the experiments is conducted to validate or demonstrate
one or more issues addressed in this thesis, in particular the work described
in Chapter 4, 5 and 6.

The conclusions of this research are presented in Chapter 8 along with
suggestions for future research.
2.1 Introduction

Industrial robots came onto the scene in the late 1960's. Their advent marks an important trend in the development of automation for the manufacturing processes. An industrial robot system generally comprises three basic components: the robotic manipulator (robot arm) itself, the power drive unit including the power supply, and the robot controller. It furnishes a flexible manufacturing environment with a flexible and general purpose computer controlled mechanical device.

A robotic manipulator consists of mechanical devices that provide the articulation, powered by pneumatic, hydraulic or electrical drives. In its most common form, it has several rigid links connected in series by revolute or prismatic joints. One end of the mechanical chain is attached to a supporting base while the other end is free and equipped with a tool to manipulate objects or perform assembly tasks. The controller in a robot system accepts user commands or operational programs. It converts these commands or operational programs to a set of co-ordinated sub-operations for each joint and drives the joint actuators to complete these sub-operations via power amplifiers. This is where the flexibility of a robot system originates: by changing the operational software of a robot controller, the manipulator can be converted to do a variety of tasks.

The control of a robot arm is a complicated matter. It involves issues of handling robot kinematics, inverse kinematics, robot dynamics, robot
trajectory planning and motion control (Paul, 1981; Ranky and Ho, 1985). A further more advanced topic involved is the incorporation of external sensors (Fu, Gonzalez & Lee, 1987). All these tasks are complex in terms of the computational requirement, and they need to be co-ordinated within the controller to perform a coherent operation.

Due to the complexities involved in the control of a manipulator, the computational power required by modern robotics applications is very high. It is expected that future generations of robots will be considerably more agile and autonomous than present day products. To achieve this, sensing, planning, and control have been proposed by Sharir (1989) as the three main areas of research to improve the capabilities of robots. A common feature of these three areas is that they require considerable computational power to support real-time operations. This has led towards a trend of design and development of special architecture robot controllers (Graham, 1989), since the computational requirement of modern robotics is much beyond the capacity of a single state of the art microprocessor.

This chapter reviews the research and development effort in the design of advanced robot controller architectures and some of the central issues involved in achieving high performance robot control.

2.2 Robot controller architectures

Current industrial practice treats each joint of the robot arm as a simple servomechanism. This simplified approach enables conventional industrial robot controllers to be built on computing machines with simple inter-connections. An illustrative example is given in figure 2.2.1, where the architecture of a Unimation Mark III robot controller is shown.
The Unimation Mark III robot controller consists of a DEC LSI-11/73 computer and six Rockwell 6503 single chip microprocessors, each with a joint encoder, a digital-to-analog converter (DAC), and a pulse-width-modulated (PWM) current amplifier to drive the associated joint DC motor. The control structure of the Unimation controller is hierarchically arranged. At the top level, the LSI-11/73 computer system supervises all the activities. It provides a user run-time interface and performs the tasks that includes parsing, interpreting, and decoding a manipulator-level robot operation language, VAL-II. The forward and inverse kinematic calculations and the trajectory generation are all handled by the LSI-11/73 computer which yields a trajectory control cycle time of 28 milliseconds. At the lower level of the control system hierarchy are the Rockwell 6503 based digital servo control units. These low level digital controllers execute commands (such as position setpoint settings, reading current encoder values, and miscellaneous parameter settings) distributed by the supervisor and implement independent servo control loop based on a proportional-integral-derivative (PID) control algorithm for each joint.

Intercommunications between the LSI-11/73 machine and the 6503 CPUs are through a dedicated interface which appears as a number of 8-bit
input/output registers on both the Q-bus and the joint communication bus J-bus in the controller.

The architecture of the Unimation controller is quite typical of industrial robot controllers. In general, these controllers feature limited computational power with low bandwidth intercommunication support. They are designed as self-contained controller systems and provide virtually no flexibility and resources for further external sensor integration, dynamic performance improvement and adoption of calibration results.

Attempts to improve the performance of industrial robots require the adoption of more sophisticated robot controller architectures. Many advanced robot control schemes require the real-time evaluation of robot kinematics, dynamics, Jacobian matrices and their corresponding inverse. These kinematics and dynamics computations feature intensive arithmetic operations with a high level of data dependency. Additional requirements for external sensor fusion further complement the driving force for improvement in computational resources. This has become a key feature in modern robotics applications (Graham, 1989). Despite the impressive speed of development of microelectronics technology, no single state of the art microprocessor can offer the computational power that modern robotics needs. The gap is ever increasing since the computational requirement in robotics applications, like in many other scientific fields, grows much faster than the speed improvement on a single microprocessor. This situation has motivated intensive research activities in developing special architecture computing machines for robot controllers, which, in general, exploit parallel processing techniques to fill this gap (Fijany and Bejczy, 1992).

In an effort to achieve efficient evaluation of the inverse dynamics problem, Nigam and Lee (1985) proposed a special computing machine
architecture for robot control, see figure 2.2.2. Their proposed architecture, called the attached processor controller (APC), consists of several microprocessors interconnected through a global broadcasting bus and a pipelined local bus structure. By organising the microprocessors in a way in which they conform to a computational decomposition form of the Newton-Euler (NE) description, the proposed computing system can fully exploit the parallel and sequential nature of the Newton-Euler dynamic equations of a manipulator. Therefore it is expected to be very efficient when used to implement control algorithms where the real-time evaluation of the robot dynamics is required, such as the computed torque control techniques. The major drawbacks of the proposed scheme seem to be lack of available software support, substantial development cost and lack of flexibility in the hardware structure.

Figure 2.2.2 Controller architecture proposed by Nigam and Lee
Some more practical application oriented special architecture robot controllers began to emerge in literature in late 1980's. Figure 2.2.3 shows the SIERA system (System for Implementing and Evaluation Robotic Algorithms) reported by Kazanzides et al. (1987). The SIERA system is based on the standard Multibus and one or more MC68000 based single board computers can be used to form the real-time servo system. The system, under development at that time, consisted of a tightly coupled bus-based real-time servo control system and a loosely coupled general purpose point-to-point network. The latter is named the Armstrong multiprocessor system and contains a number of MC68010 microprocessors for potential multi-robots coordination and vision systems. The SIERA system has user level support on both C and assembly languages, and operating systems are provided for both the servo control system and the Armstrong network. The coding of an application is also facilitated by various library routines. However, the SIERA system seems to lack sufficient computational power in its real-time servo system to support some computationally intensive...
control algorithms, such as those based on computed torque control techniques. The system also suffers from poor flexibility and the computational power expansion may be limited by the bus bandwidth.

![Diagram of RAL Hand and Manipulator Control Architecture](image)

**Figure 2.2.4** The architecture for RAL hand and manipulator control

A similar Multibus based multiprocessor robot controller is also reported by Kim et al. (1987). Their computer system consists of a number of National Semiconductor 32000 based single board computers working in a master-slave configuration (figure 2.2.4). Each of these boards contains a 10MHz NS32000 chip-set with a 32016 CPU and a floating point support unit. The controller system was developed for the control of a four fingered RAL (Robotics and Automation Laboratory, University of Toronto) hand and for replacing the kinematics and trajectory control parts of an existing PUMA 560 robot controller. This scheme makes it possible to achieve better coordinated operations between the dextrous hand and the PUMA manipulator on which the RAL hand is mounted. The work mainly concentrated on the kinematic control issue as the combination of a 6 degrees of freedom manipulator and a 15 degrees of freedom hand leads to a
complicated kinematically redundant system. Dynamic control of the PUMA manipulator, however, was not covered.

Another high performance bus standard—VMEbus is also very popular in the development of multiprocessor based robot controllers. Narasimhan et al. (1989) reported a VMEbus based controller system, the CONDOR system, for the control of the Utah-MIT dextrous hand developed at the MIT Artificial Intelligence Laboratory, Cambridge, USA. Essentially, the CONDOR system consists of a development environment host connected to a real-time controller. The development environment employs a Sun-3/160 machine to offer the facilities of program development, and the compiled programs for the real-time controller can be downloaded to the target processors through a VME-VME bus repeater. The real-time controller of the CONDOR system consists of a number of MC68020 based computing units, each with a floating point co-processor to speed up arithmetic operations (see figure 2.2.5). These computing units are tightly coupled through the VMEbus connection. The peripheral boards shown in figure 2.2.5 are also directly attached to the VMEbus which forms the backbone of the integrated real-time control system.

![Architecture of the CONDOR system](image)

**Figure 2.2.5 Architecture of the CONDOR system**
Chapter 2: A Review of Robot Controller Architectures and Robot Control

The design of the system software of the CONDOR system adopted the approach of writing device drivers to provide structured software interfaces for various input/output devices. The low level details of a device, including its register formats and interrupt mechanisms, are abstracted from user code and hidden within the device driver. This provides a neat and convenient way for user application programming. To achieve efficient real-time control task scheduling, a simple servo control scheduling mechanism is designed for each M68020 microprocessor in the CONDOR system. This is achieved by restricting the number of tasks to virtually two per microprocessor: one runs as the normal (background) task, and the other as driven by interrupt signals.

Despite the higher computational power of the microprocessors and the higher specifications of the VMEbus, the CONDOR system is virtually the same as the Multibus based controllers in structure. Its major disadvantages are the non-flexible structure, no directly supported intercommunication mechanism in multiprocessor programming, and limited expansion capacity due to the bus bandwidth.

Special microprocessors have also been employed to construct high performance robot controllers. Shalom and Kazanzides (1989) reported the SPARTA (signal processor architecture for real-time applications of robot control) system that used a number of the specially designed signal processor subsystem units—the PIEs (personal instrument enterprises). As shown in figure 2.2.6, the SPARTA system consists an IBM VM/CMS mainframe computer for program development, an IBM PC-AT for user runtime support and the PIEs for real-time control processing. Peripheral interface boards are attached to the PC-bus and are managed by the PC host. In figure 2.2.6, each signal processor subsystem has four boards connected by a local bus. These boards are: the signal processor (SP), the PC host
attachment board (PCHAB), the general purpose I/O board (GPIOB), and the main signal processor memory (MSPM). This four board architecture for each processing unit provides the flexibility and the relative efficiency for the communication mechanisms required in the design specifications of the SPARTA system.

![Diagram of the SPARTA system hardware structure](image)

**Figure 2.2.6 Hardware structure of the SPARTA system**

The SPARTA uses the IBM PC-AT to provide the runtime and real-time file service operations. The user interface is implemented through the IBM PC-AT terminal. Since the operating system of the IBM PC-AT, the MSDOS or PC DOS, is a single user and single task system, real-time support for the multi-PIEs, their coordination and other fundamental operations has to be designed as interrupt driven modules to achieve pseudo-parallel executions. In each of the PIEs, a signal processor kernel is designed to handle the low-level processor details, such as servicing interrupts and
executing user programs. Real-time control software is organised into modules and allocated to different PIEs. Each of these modules is a separate task. Communication between tasks is accomplished via shared memory. If the tasks are on different processors, a PC-resident interrupt service routine is used to transfer the data from one processor to another.

The SPARTA system strongly relies on the communication via the PC-bus despite that the use of the two 16-bit parallel ports implemented on the PCHAB in a PIE may partially off-load the communication burden. From this perspective, the system still resembles the previously described Multibus or VMEbus based architectures. Programming in the SPARTA system seems not to be very convenient. Detailed operations of the PC-bus communication between any two PIEs have to be accounted for in the user software. The system also features intensive interrupting activities over the PC-bus, since any intercommunications between the PIEs via the PC-bus are relayed by the PC-AT host through interrupt driven modules. Therefore, the system’s efficiency will degrade if the real-time control algorithm requires frequent information interchanges.

It is worth pointing out that, the effort in design and development of special architecture robot controllers seems can be divided into two areas. In one area, a particular structure is proposed with the aim of fully exploiting some features demonstrated by the robot dynamics or kinematics at the hardware level of the controller architecture. The proposed structure, either a special design of the computing machine or a special VLSI (very large scale integration) chip, represents the most favourable architecture that a robot controller should have in order to benefit most from the particular features addressed. Examples of this category can further be found in the works of Lathrop (1985), Wang & Butner (1987), Seshadri (1987), Chang & Lee (1988) and Ling et al. (1988).
In the other area, standard bus based multiprocessor systems are adopted due to their low cost, commercial availability, various peripheral boards, easy integration and rich software support. Most of the reported developed or developing robot controllers, such as those briefly described above, belong to this category. And further examples can be found in the works of Cox et al. (1988), Bejczy & Szakaly (1987), and Lumia et al. (1990). These standard bus based multiprocessor systems simplify the construction tasks, but leave the exploitation of the dynamics and kinematics features of robot control problems entirely up to the software design. Within the limits of the intercommunication mechanism that can be realised or supported on a particular bus standard, the exploitation of the dynamics and kinematics features may only partially be achievable. Nonetheless, the bus based architecture has been the only type practically adopted for developing advanced robot controllers in many universities and research institutes. This situation seems that it will not change in the near future.

2.3 Kinematic models and inverse kinematics algorithms

Robot kinematics deals with the analytical study of the geometry of motion of a robot arm with respect to a fixed reference coordinate system without regard to the force/moments that cause the motion. Thus, kinematics describes the spatial arrangement, according to sequences and structure, of the axes of movement (degree of freedom) in relation to each other. The task of kinematics is to enable arbitrary spatial points in a work area to be approached and to create the desired spatial relationship between the end-effector or tool and this point.

There are two fundamental questions in dealing with robot kinematics. The first one is that, given a joint angle vector of a robot, what is the position and orientation of the robot's end-effector with respect to its
Chapter 2: A Review of Robot Controller Architectures and Robot Control

base coordinate system. This question is concerned with the mapping from the joint space to Cartesian space of a robot, and is commonly termed the forward kinematics problem. The other question addresses the reverse of the first one, i.e. given a desired position and orientation of a robot, what value of the joint angle vector should be so that it will lead the robot's end-effector to be positioned at the demanded pose. The second question is known as the robot inverse kinematics problem.

The inverse kinematics problem is a fundamental issue in achieving robot control. Although industrial robots have been designed to have a special mechanical structure that reduces the difficulty in finding the solutions to the inverse kinematics problem, this simplicity is generally not preserved after a calibrated kinematic model is adopted. Extra efforts must be made to achieve the performance improvement through the implementation of calibrated results, which requires accurate and stable real-time inverse kinematics algorithms.

2.3.1 Robot Kinematic Models

The most popular mathematical tool used for kinematic modelling is the homogeneous transformation matrix approach. As for the kinematic model itself, the convention developed by Denavit and Hartenburg (1955), known as the DH model, is often adopted for its simplicity. The DH model uses 4 parameters, the minimum number required, to describe the relationship between consecutive links in a serial chain. This description enables simple models to be generated of 'ideal' joint kinematic behaviour i.e. pure translation and pure rotation.

According to the DH model, each link of a serial mechanical chain is attached with a link coordinate frame as shown in Figure 2.3.1. A
homogeneous transformation matrix $A_i$ can be derived which relates the coordinate frame of link $i$ with respect to coordinate frame of link $i-1$.

$$A_i = \text{Rot}(Z,q_i) \ \text{Trans}(Z,d_i) \ \text{Trans}(X,a_i) \ \text{Rot}(X,\alpha_i)$$

$$= \begin{bmatrix}
    c_q i & -s_q i & c_\alpha i & a_i c_q i \\
    s_q i & c_q i & s_\alpha i & a_i s_q i \\
    0 & s_\alpha i & c_\alpha i & d_i \\
    0 & 0 & 0 & 1
\end{bmatrix} \quad (2.3.1)$$

where $c_x$ stands for $\cos(x)$, and $s_x$ for $\sin(x)$.

The four DH model parameters are:

$q_i$ Angle measured in the plane perpendicular to the $Z_{i-1}$ axis, from the $X_{i-1}$ axis to the common normal. This is the joint variable for a revolute joint.

$\alpha_i$ Angle between $Z_{i-1}$ and $Z_i$ measured in the plane perpendicular to the common normal, known as the link twist angle.
Chapter 2: A Review of Robot Controller Architectures and Robot Control

\(a_i\) Length of the common normal between \(Z_{i-1}\) and \(Z_i\).

\(d_i\) Displacement between the intersection of the \(X_{i-1}\) axis and the \(X_i\) axis along the joint axis \(Z_i-1\). This is the joint variable for a prismatic joint.

Having established each link’s homogeneous transformation matrix, the overall kinematic model of a robot which relates the frame attached at the end-effector, known as the frame \(N\), to the robot base coordinate frame, labelled frame 0, can be given by combining successive \(A_i\) matrices:

\[
T_N = A_1A_2 \ldots A_n = \prod_{i=1}^{N} A_i \tag{2.3.2}
\]

The matrix \(T_N\) specifies the position and orientation of the endpoint of the manipulator with respect to the base coordinate system. This \(T_N\) matrix is frequently referred as the arm matrix. It has a descriptive form:

\[
T_N = \begin{bmatrix}
    n & o & a & p \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
    n_x & o_x & a_x & p_x \\
    n_y & o_y & a_y & p_y \\
    n_z & o_z & a_z & p_z \\
    0 & 0 & 0 & 1
\end{bmatrix} \tag{2.3.3}
\]

Where (reference figure 2.3.2)

\(n\) the normal vector of the end-effector. It is orthogonal to the fingers of the robot arm.

\(o\) the orientation vector of the end-effector. It is pointing in the direction of the finger motion as the gripper opens and closes.

\(a\) the approach vector of the end-effector. It is the cross product of the vector \(n\) and \(o\).
the position vector of the end-effector. It describes the origin of the end-effector coordinate system in the robot base coordinate system.

The kinematic model is an important internal description of the underlying mechanical structure and the geometry relationships of the robot arm in a robot controller. Its accuracy directly affects the precision of the end-effector pose. In order to achieve better pose accuracy, it is often needed to identify the 'true' kinematic parameters in a complex test procedure. This process is known as the robot kinematic calibration.

It should be pointed out that, although the DH model has been popular for modelling manipulator kinematics, several problems arise when using this model in a calibration procedure. These problems are related to the issues of arbitrary location of joint coordinate frame and base frame, and proportionality of model parameters (Mooring et. al, 1991). The DH convention specifies the link coordinate frames to be located at the
intersection of the joint axis and the common normal. This implies that the location of these coordinate frames is a function of the manipulator geometry and causes the model parameters vary by large amounts for revolute joints with nearly parallel axes. Such singularity in model parameters has severe consequences during a parameter identification scheme, resulting in computational difficulties in a calibration process. In order to avoid one or several aspects of the inappropriateness of the DH model in kinematic calibration, a number of more complex kinematic models have been proposed during the last decade. Examples are the models proposed by Hayati (1983), Hsu and Everett (1985), Stone (1986), etc. In general, these models use more than 4 parameters to describe the coordinate frame relationship between consecutive links. For example, the modified S-model, developed initially by Stone et al (1986) and then improved by Stanton (1991), has 6 parameters. It allows arbitrary positioning and orientation of the link frame on the joint axis which enables the robot link parameters to be identified by a set of simple decoupled identification problems. Discussions about various robot kinematic modelling techniques and their advantages as well as disadvantages during calibration can be found in Hollerbach (1988), Stanton (1991), Mooring (1991), and Bernhardt and Albright (1993).

### 2.3.2 Inverse kinematics algorithms

Control of a robot manipulator is carried out in joint space. This basic feature of robot control requires solving the transformation of the position and orientation of a manipulator from Cartesian coordinates to joint coordinates. The basic description of the inverse kinematics problem is that given a desired robot position and orientation $T_d$:

$$T_d = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.3.4)$$
find the corresponding joint angles $\theta_1, \theta_2, ..., \theta_n$ so that

$$T_N = A_1(\theta_1)A_2(\theta_2) ... A_n(\theta_n) = T_d \quad (2.3.5)$$

In general, equation (2.3.5) constitutes a set of highly non-linear equations for joint angles $\theta_1, \theta_2, ..., \theta_n$. There is no single explicit systematic procedure to derive the solutions. Fortunately, special mechanical structures have been adopted in the design of industrial robots. This has significantly reduced the difficulty in the search for solutions for the inverse kinematics problem when a nominal kinematic model is used.

**Closed form inverse kinematics solutions**

Closed form solutions to the inverse kinematics problem are available only for certain classes of robots with a simple mechanical structure. These simple structure robots have consecutive joints that are either parallel or perpendicular to each other. With these special configurations, the link twist angle $\alpha$ is either 0 degrees or 90 degrees. This leads to many elements of the $A_i$ matrices, defined in equation (2.3.1), having a value of zero or unity. Utilising the properties of simple structure manipulators, Paul (1981) developed a convenient method for determining an algebraic expression for joint variables of each joint for a given end-effector pose. The method works in a sequential manner and isolates a single joint variable to solve each step. When all the joint variables have been determined, a solution is found. Although the method is general and simple, it suffers from the fact that no clear indication is given on how to select an appropriate solution from the several possible solutions for a particular arm configuration.

In the geometric approach proposed by Lee et al. (1984), some arm configuration indicators are directly placed into the analytical expressions to uniquely determine the inverse kinematics solution for a given robot end-
Chapter 2: A Review of Robot Controller Architectures and Robot Control

effector pose. Figure 2.3.3 illustrates the definition of the arm and elbow configuration indicators for a PUMA manipulator. A further configuration indicator on the wrist can be defined which specifies the flip operation of the wrist. These arm configuration indicators are pre-specified by a user to define the desired arm configuration for finding the inverse kinematics solution. The main advantages of this method are that it provides more insight into solving simple manipulators with rotary joints and the inverse kinematics solution is unique with the arm configuration indicators properly given.

![Definition of arm and elbow configuration indicators](image)

Figure 2.3.3 Definition of arm and elbow configuration indicators

A closed form inverse kinematics solution is desirable due to its simplicity and computational efficiency. But unfortunately manipulators
with a general structure do not have a closed form inverse kinematics solution. The following are two sufficient conditions which make the closed form inverse kinematics solution possible (Fu, Gonzalez & Lee, 1987):

1). Three adjacent joint axes intersecting;
2). Three adjacent joint axes parallel to one another.

It should be pointed out that most of the commercial robots satisfy one of the sufficient conditions in their nominal configurations. For this reason, they are frequently referred to as simple structure manipulators.

**Numerical inverse kinematics solutions**

With closed form inverse kinematics solutions only available to some simple structure manipulators, numerical approaches become more essential when general cases are considered. In fact, even a nominally simple structured industrial robot will in general no longer have a closed form inverse kinematics solution after its calibrated kinematic model (which is regarded as the 'true' kinematic behaviour of the manipulator) is in use (Judd and Knasinski, 1987; Broderic and Cipra, 1988). Thus it is not surprising that numerical approaches are of wide interest.

A number of general numerical methods for solving the inverse kinematics problem have been reported in literature. They can basically be divided into two types. The first type uses either the Newton-Raphson method to iteratively approach the inverse kinematic solutions or predictor-corrector-type algorithms to integrate the differential kinematic equations. Frequently, the Jacobian matrix, which is a differential description between the joint coordinate space and the Cartesian space, is
used to relate the iterative error to the joint variable changes. The method stops the iteration procedure when the iterative error has reached within a pre-specified error tolerance and the corresponding joint coordinate vector is regarded as the inverse kinematics solution. Examples of this type include the methods proposed by Angeles (1985), Goldenberg et al. (1987), Tsai & Orin (1987), Tucker & Ferreira (1987). The major difficulty with these methods is that, when the Jacobian matrix is singular (or ill-conditioned), they do not provide satisfactory inverse kinematics solutions. In addition, the computational burden is not a trivial one as each iteration contains, as part of the computational procedure, an evaluation of the Jacobian matrix and a numerical inverse or pseudo-inverse of it. Furthermore, since there are several possible inverse kinematics solutions, which one of them will be converged to depends on the initial joint vector selection and no clear configuration indication is available. It should also be noted that the performance of these algorithms strongly relies on the accuracy of the initial inverse kinematic solution approximations. Both the stability and accuracy of the algorithms may be affected by an insufficient accurate initial estimate of the intended inverse kinematic solutions.

The second type uses optimisation techniques to find the inverse kinematics solution. By formulating a scalar cost function, which reflects the discrepancy between the desired robot pose and the pose defined by a joint variable vector through the kinematic model, the inverse kinematics problem is converted into an equivalent minimisation problem. The inverse kinematics solution is regarded found once the cost function has reached its minimum. This enables the well developed various optimisation techniques to be used in solving the inverse kinematics problem. Wang and Chen (1991) proposed a two step optimisation algorithm to obtain the inverse kinematics solution. At the first step, a procedure called cyclic coordinate descent optimisation technique is
iteratively applied to bring the joint variable vector to the vicinity of a inverse kinematics solution. Then the second step utilises the \textit{Broyden-Fletcher-Shanno} optimisation method to fine tune the obtained coarse solution. The method is general and numerically stable. However, the approach is rather complex and computationally expensive. Therefore it seems not suitable for real-time applications.

Another useful technique in solving the inverse kinematics problem is to utilise a closed form inverse kinematics solution of a simple structure robot kinematic model to approach the solution of a robot kinematic model that does not have a closed form inverse solution. The precondition that enables the approach is that the differences between the two kinematic models can be attributed to some tiny differential changes of a set of parameters. This is of particular interest when a calibrated kinematic model is used for an industrial robot that has a nominal simple structure. Vuskovic (1989) proposed two methods in approximating the inverse kinematics solution of a calibrated industrial robot through its nominal kinematic model. Both the methods are based on a parameter sensitivity function. The first method uses the Jacobian matrix to relate the compensation differential changes of the inverse kinematic solution with the differential pose shifts calculated by the parameter sensitivity function. The inversion of the Jacobian matrix is employed to find the inverse kinematic solutions. Since the Jacobian matrix is involved, the method is similar to the Newton-Raphson method based approaches. Hence it suffers from the same difficulty when the robot pose is near its singular points. The second method uses the sensitivity function to obtain a pose shift. This pose shift is aimed at making the modified pose lead to an inverse kinematic solution of the nominal model that 'coincides' with the solution of the intended inverse kinematics problem. The second method does not involve the troublesome inversion of the Jacobian matrix, and is claimed,
by Vuskovic (1989), to be more efficient. However, it should be pointed out that, the computational complexity of the sensitive function evaluations is proportional to the involved parameter numbers, and since a considerable part of the sensitivity function evaluation formulations cannot be shared by the nominal model inverse calculations, the method becomes inefficient when the parameter number increases.

Finding real-time inverse kinematics solutions to general structure manipulators is still an unresolved problem. Fortunately, most of the industrial robots are simple structured and hence have a closed form inverse kinematics solution to their nominal kinematic models. This makes it possible to derive real-time applicable numerical inverse kinematics algorithms to solve the inverse kinematics problem of calibrated industrial robots by adopting methods similar to but more efficient than those suggested by Vuskovic (1989). It should be pointed out that, depending on the particular models adopted in the calibration procedures, some of the calibrated models will not directly provide a parameter set that is a small shift from the parameter set in the nominal kinematic models. Therefore, model conversion techniques must be established to make the derived algorithms viable.

2.4 Dynamic models and control algorithms

Robot dynamics deals with the dynamic behaviour of a robot arm. It is intimately coupled with the control problem. In fact, the purpose of robot control is to maintain the dynamic response of a robot arm in accordance with some pre-specified system performance and desired goals. In general, the dynamic performance of a robot arm directly depends on the efficiency of the control algorithms and the dynamic model of the robot.
2.4.1 Dynamic models of a robot

The actual dynamic model of a robot arm can be obtained from known physical laws such as the laws of Newtonian mechanics and Lagrangian mechanics. This leads to the development of the dynamic equations of motion for various articulated joints of the robot arm in terms of specified geometric and inertial parameters of the links. There are a number of ways to systematically develop the actual robot motion equations. All the resultant sets of equations are 'equivalent' to each other in the sense that they describe the dynamic behaviour of the same physical robot arm system. However, the structure and the parameter definitions of these equations may differ significantly as they are obtained from different perspectives for various reasons and purposes.

The most common approaches in dynamic modelling of a robot arm are the Lagrange-Euler (LE) and Newton-Euler (NE) formulations. The derivation of the dynamic model based on the LE formulation is simple and systematic. Its resultant equations of motion are a set of second order, coupled, nonlinear differential equations and are in explicit state variable form (Graig, 1986) when only the rigid arm links are considered. These explicit state variable equations for robot dynamics can conveniently be utilised to analyse and design advanced joint-variable space control strategies. The drawbacks of the LE approach is that it is inefficient for computational purposes, and hence hinders its application for real-time control purposes (Fu, Gonzalez & Lee, 1987).

Another popular approach in dynamic modelling of a robot arm is the NE formulation. It is often adopted as an alternative to LE equations when one's interests are in real-time control applications. The NE approach is a complex derivation and the resultant equations involve vector cross-
Product terms. The derived dynamic equations are a set of forward and backward recursive equations. This set of recursive equations can be applied to the robot links sequentially. The most significant result of the NE formulation is its computational efficiency. In fact, the computation of joint torque from NE equations of motion is the most efficient at present time and has been shown to possess a time lower bound proportional to the number of robot joints when the computation is performed in a uniprocessor machine (Lee & Chang, 1986).

Using LE approach, the dynamics equations of a rigid robot arm of n joints can be expressed in a matrix form as

\[ M(q)\dot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \quad (2.4.1) \]

Where the \( M(q) \) is the \( n \) by \( n \) dimensional generalised inertial matrix, \( G(q) \) is the \( n \) dimensional gravity vector, \( q \) is the \( n \) dimensional joint coordinates vector and \( \dot{q}, \ddot{q} \) its first and second order time derivative vectors, \( \tau \) is the generalised force/torque vector, and \( C(q, \dot{q})\dot{q} \) is the centrifugal and Coriolis torque vector with the \( i, j \)-th element of the matrix \( C(q, \dot{q}) \) defined as

\[
c_{ij} = \sum_{k=1}^{n} \frac{1}{2} \left( \frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \dot{q}_k \quad (2.4.2)
\]

where \( m_{ij} \) is the \( ij \)-th element of the inertial matrix \( M(q) \).

There are some interesting properties about the matrices in (2.4.1). The most useful two of them are:

1). \( M(q) \) is always symmetric and positive definite, i.e.
Chapter 2: A Review of Robot Controller Architectures and Robot Control

\[ M(q) = M(q)^T > 0 \quad (2.4.3) \]

2). The matrix \( M(q) - 2C(q,q) \) is skew symmetric, and

\[ x^T(M(q) - 2C(q,q))x = 0 \quad (2.4.4) \]

where \( x \) is an arbitrary real vector of dimension \( n \).

The above two properties are widely used in the synthesis of adaptive robot control algorithms.

2.4.2 Control algorithms

Manipulators are highly nonlinear, internally coupled systems. Their high performance control has been one of the areas which has attracted a substantial research effort within the robotics and control community. There have been a large number of papers proposing various control algorithms during the last decade. Typical categories covered by them are:

1). Computed torque technique;
2). Variable structure control;
3). Adaptive control;
4). Fuzzy logic control.

The motivation behind the above efforts is the fact that conventional industrial robot controllers make use of simple linear control algorithms which ignore the coupling and nonlinear nature of the dynamics of the manipulator and thus treat each link as a separate decoupled system. This inevitably brings about poor control performance when the manipulator is demanded to move at a high speed. All the more advanced control algorithms are aimed at overcoming some uncertainties demonstrated in applications; rejecting of the internal and external disturbances, reducing the computational burden while still keeping the control performances...
Chapter 2: A Review of Robot Controller Architectures and Robot Control

within an acceptable range, etc. In short, they are proposed as an effort to improve the control performance in one or several aspects.

Computed torque method

The computed torque method is a forward compensation control technique based on the robot dynamic model. Its basic formulation is

\[ \tau = M_a(q)(\dot{q}^d + K_v[\dot{q}^d - q] + K_p[q^d - q]) + C_a(q, \dot{q})\dot{q} + G_a(q). \tag{2.4.5} \]

where \( \dot{q}^d, q^d \) and \( q^d \) are the desired trajectory information. Assuming the robot model is fully known, i.e. \( M_a, C_a \) and \( G_a \) equal to their counterparts \( M, C \) and \( G \) in (2.4.1), the substitution of (2.4.5) into (2.4.1) leads to an error equation

\[ M_a(q)[\ddot{e} + K_v\dot{e} + K_p\dot{e}] = 0, \tag{2.4.6} \]

where \( e = q^d - q \) is the error vector, and \( \dot{e} \) and \( \ddot{e} \) are its first and second time derivatives. Since \( M_a(q) \) is always non-singular, (2.4.6) can be rewritten as

\[ [\ddot{e} + K_v\dot{e} + K_p\dot{e}] = 0. \tag{2.4.7} \]

Thus by carefully choosing the controller gain matrices \( K_v, K_p \) so that all the eigenvalues of the matrix \( (I+K_v+K_p) \) have negative real parts, the position error vector \( e \) will approach zero asymptotically.

The computed torque method relies on the availability of the dynamic model. Its performance is sensitive to the model discrepancies and may result in an unstable system if the model mismatch is significant enough (Spong & Vidyasagar, 1989). In addition, its implementation in control...
Variable structure control

Variable structure control (VSC) is a robust control approach that has been widely used in industry. The main feature of a variable structure system is that it has a so-called sliding mode on the switching surface. Within the sliding mode, the system remains insensitive to parameter variations and disturbances.

Variable structure control of a manipulator defines a switch function for each joint error variable $e$

$$s = c e + \dot{e} \quad (2.4.8)$$

where $c > 0$. The joint system is in sliding mode when $s = 0$. In order to drive the joint system into the sliding mode and keep it constrained there, the variable structure control system needs to satisfy the 'sliding condition',

$$ss < 0 \quad (2.4.9)$$

To generate such a motion, the simplest control scheme is

$$\tau = -k \text{sgn}(s) \quad (2.4.10)$$

with $k$ a sufficient large positive constant.

Variable structure control eliminates the nonlinear interactions among the joints by forcing the system into the sliding mode. However, the control outputs of the controller are high frequency discontinuous signals and cause the manipulator to have a chattering behaviour. Recent developments on VSC incorporate a continuous term in the control signal.
to reduce the chattering (Asada & Slotine, 1986; Richards & Reay, 1992; Zhu et al., 1992). However, this seems to be departing from the original principle of VSC by partially losing some of its attractive features, such as the system's insensitivity to the system dynamics.

**Adaptive Control**

Adaptive control has been the most active research area of robot control techniques during the last decade. The main motivation of adopting the adaptive control algorithms in robot controllers originated from the fact that most of the simple linear feedback control algorithms are unable to cope with the time-varying complex nonlinearity, strong couplings and uncertainties demonstrated by the robot dynamics.

Early adaptive control approaches make some special assumptions such as slowly varying parameters, linearization of dynamics along the nominal path, etc. (Dubosky and Desforges, 1979; Lee and Chung, 1984; 1985). By these assumptions, the error dynamics derived appear as a stable, linear, time invariant system. Thus a relatively simple adaptive solution can be established which generally works well under low speed situations.

By reparameterizing the manipulator dynamics equations as a product of a regressor matrix and an unknown constant vector of parameters, Craig et al. (1986) proposed an asymptotically stable adaptive controller. This marks a turning-point in the development of adaptive robot control techniques, since the special assumptions made in earlier stages are no longer needed. With the reparameterization of the robot dynamics equations, (2.4.1) can be re-written as

\[
M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Y(q, \dot{q}, \ddot{q})\theta = \tau \quad (2.4.11)
\]
where $\theta$ is a constant parameter vector, $Y$ is the so-called regressor matrix which only depends on the values of $q$, $\dot{q}$ and $\ddot{q}$. The adaptive control is achieved by calculating the control torque

$$\tau = Y(q, \dot{q}, \ddot{q})\theta'$$

(2.4.12)

$$\ddot{\theta} = q^d + K_v[q^d - q] + K_p[q^d - q]$$

(2.4.13)

where $\theta'$ is the estimation of the parameter vector $\theta$, and is on-line updated according to an adaptive law derived from the Lyapunov stability theory.

Following Craig et al's work, further development on adaptive control of robot manipulators is widely reported (Middleton and Goodwin, 1990; Hsu et al., 1987; Kelly et al., 1988; Slotine and Li, 1987; 1989; Sadegh and Horowitz, 1990a; 1990b; Ortega and Spong, 1989; Song et al., 1992). The fundamental feature of these works is to reparameterise the robot dynamics equations so that an asymptotically stable adaptive control algorithm or an exponential stable adaptive control algorithm can be established. By reparameterizing, the time-varying effects of the nonlinear dynamics are all attributed to the regressor matrix, which is supposed to be available to the controller. This, however, puts a very heavy burden on the computational requirement, as commented by Sadegh and Horowitz (1990a), since most of the items in the regressor matrix are highly nonlinear functions of joint positions and their higher derivatives. The dimensions of the regressor matrix increase very rapidly with the increase of the degree of freedom of the robot manipulator. Therefore the applications of these algorithms seem to be limited by the exceptional computational requirements.

Parallel to the regressor matrix approach, Yuan and Stepanenko (1992) proposed a $\|q\|$ modified adaptation law to cope with the time-varying...
Chapter 2: A Review of Robot Controller Architectures and Robot Control

behaviour of the robot dynamics equations. The \( \Theta \) modified adaptation law is a further modification to the well-known scheme proposed by Ioannou and Kokotovic (1983) in which a leaking coefficient \( \sigma \) (the \( \sigma \)-modification) is used in the parameter adaptive updating differential equation to deal with the non-trivial external disturbances. By introducing the \( \Theta \) item as the leaking coefficient into the parameter updating differential equation, an adaptive nonlinear feed-back component, composed of an item based on the joint’s speeds and an item based on the error integration, is formulated in the control algorithm in Yuan and Stepanenko’s work. Instead of obtaining an asymptotic stable system, the closed-loop system is proven to be uniformly bounded. The major benefit of this approach is the much simpler computational complexity. However, the proposed scheme does not incorporate any sort of feed-forward dynamic compensation, thus it may not be able to provide satisfactory control results when reference signals vary rapidly.

**Fuzzy logic control**

Control of a robot manipulator may be based on two different strategies: model based control, and knowledge based control. The algorithms reviewed before are all robot model based.

Fuzzy logic control is a knowledge based approach. It derives its actions through a linguistic description operation (fuzzy logic operations). In a knowledge based robot control system, the description of the robot dynamics needs not to be an analytical model. This feature makes the fuzzy logic control a rather attractive option, since the accurate modelling of a robot arm is both difficult and costly. Although the direct use of a knowledge based control approach within a low-level control loop may not be suitable (de Silva and MacFarlane, 1989), due to the degraded control bandwidth etc.,
its adoption into a higher level to enable the robot controller being insensitive to internal/external disturbances seems to be promising (de Silva, 1990). An illustrative scheme is shown in figure 2.4.1 where a fuzzy adaptation mechanism is employed to tune the PID controller parameters.

![Diagram](image)

**Figure 2.4.1 Linear control with fuzzy logic adaptation**

Applying fuzzy logic control to solve the robot control problem has now generated increasing interests among the robot control community. Some preliminary but encouraging results have been reported in the literature (Wakileh and Gill, 1988; Lim and Hiyama, 1991).

### 2.5 Trajectory generation

In robot manipulator control systems, the role of the trajectory generator is to convert commands specified by a programming level into a stream of setpoints suitable for tracking by a feedback controller. A typical programming command will specify constraints for the manipulator to satisfy, such as target positions, velocities, path shape, and temporal requirements along the path. The task of trajectory generation is then to produce setpoint paths that meets these constraints as closely as possible.
Trajectory generating schemes generally "interpolate" or "approximate" the desired path by a class of polynomial functions. The desired path can be defined either in the Cartesian coordinate system or in the robot joint coordinate system. Similarly, the trajectory generation can also be carried out in Cartesian space or in joint space. In general, the operator would like to specify the desired path in the Cartesian coordinate system in which he can easily visualise the correct end-effector configurations. However, since robot control is normally carried out based on the joint coordinates information. The trajectory functions generated in Cartesian space must be converted to the joint coordinate space through the inverse kinematics solution routine. This puts an extra burden on the computational requirements compared to the trajectory functions generated in the robot joint coordinate system.

The joint space oriented trajectory generating schemes normally use low order polynomials to provide the interpolation points between the specified path knot points. Commonly used polynomials are linear functions with parabolic blends (Craig, 1986) and cubic polynomials (Lin et al., 1983; Fu, Gonzalez & Lee, 1987). Most of the joint space oriented trajectory generating schemes provide smooth and continuous joint trajectories up to the second time derivatives. Some even include the minimisation of the Jerk magnitude (Kyriakopoulos, 1988). However, the exact Cartesian path on these interpolated points may not follow the desired Cartesian path. Therefore, the interpolation knots must be closely laid out so that the resultant path error is within an acceptable range.

Cartesian space based approaches are comparatively complex. Paul (1979) discussed the straight line path generation in Cartesian space, using the homogeneous transformation matrix to represent target positions. Movement between two consecutive target positions is accomplished by
two sequential operations: a translation and a rotation to align the approach vector of the manipulator hand and a final rotation about the tool axis to align the gripper orientation. Blending techniques are then employed to guarantee smooth transition between two connected path segments. Taylor (1979) further extended Paul's method, where he adopted the quaternion representation technique. Later Cartesian space generating schemes take into consideration both the path continuity and some physical constraints, such as torque constraints or speed, acceleration and jerk bounds (Luh and Lin, 1984). Recent development has involved the trajectory generation on any Cartesian curve by using an algorithm which generates enough proper intermediate path knots for coordinate space interpolations (Chang et al., 1988).

Many proposed trajectory generation algorithms have formulated the problem as to find an optimum solution which minimises a given cost function under various constraints (Yamamoto et al., 1988; Park & Lee, 1992). These optimisation approaches are rather time consuming and are in general used as off-line trajectory generating techniques. They allow both tasks and manipulator constraints to be addressed in some optimal fashion. The problem with such an approach is that all the detailed motions of the robot need to be specified in advance, and once the trajectories are computed off-line, they are generally difficult to modify in response to real-time sensor information. This is unacceptable to applications where external sensor guidance are involved. For such applications, a real-time algorithm for generating trajectories is a necessity (Koivo and Houshangi, 1991).

On-line trajectory generation, by contrast, establishes the setpoints in real-time, usually at some known sample rate, at the same time they are sent to the robot motion controller. This maximises the opportunity to
respond to sensor driven events, at the expense of that only very localised constraints can be included into the trajectory generation. It requires considerable computational power to provide the support for real-time implementation, in particular when the trajectory generation is undertaken in the Cartesian space. A common technique in on-line trajectory generation is to compute path segments that satisfy each robot motion command, and then join these together using blending techniques that provide smooth transitions between the path segments.

On-line trajectory generation in Cartesian space is particularly important in external sensor based applications. This is because most of the workcell feedback is formulated in the Cartesian space in which the relationship between the robot system and the environment is straightforward. Despite the fact that, very frequently, the paths specified in the Cartesian space are simple straight lines, the implementation of on-line path blending and alteration, required by the external sensor based robot applications, is rather complex due to the difficulty of handling orientations, and is rarely addressed in the literature. In general, an effective strategy for handling the real-time trajectory adaptation issue is required to achieve external sensor guided robot dynamic tracking and intercepting operations.

2.6 Summary

Various aspects involved in robot control have been reviewed in this section. The emphasis have been placed on issues concerning robot controller architectures, inverse kinematics solutions, robot dynamic control and the on-line trajectory adaptation based on external sensory information.
Chapter 2: A Review of Robot Controller Architectures and Robot Control

Of the previously developed robot controllers, standard bus based multiprocessor architectures are generally adopted. The major advantages are the low cost, short development period, wide range of commercially available peripheral boards, easy system integration and rich software support. However, as the bus standards normally do not provide user level inter-processor communication and coordination means, the multiprocessor programming issue becomes a potential problem. Furthermore, due to the limited bus bandwidth, the computational capacities of these systems are constrained by the bus communication bottleneck.

Research into kinematic and dynamic control of industrial robots has been widely conducted during the last decade. Efforts to improve the industrial robot accuracy have yielded calibration techniques that aim at establishing a more accurate kinematic model by identifying the 'true' geometry or 'signature' of the manipulator. Although an industrial robot may have a closed-form solution to its inverse kinematics problem when the nominal model is concerned, the inversion of the calibrated model is much more problematic. To take full advantage of the calibrated results, the real-time implementation issue must be addressed. This generally requires the numerical solution methods that may be used for on-line kinematics inversion as well as the methods of modifying the robot kinematic control software.

Dynamic control of manipulators is an area which has been viewed as having potential for further performance improvement. Various control algorithms that address the dynamic control problem of manipulators have been proposed during the last decade. Because of the complexities involved in the robot control problem, very few of the proposed algorithms have conducted experimental evaluations. Indeed, of all the published papers,
only a few has ever tried to address the algorithm implementation issue. In general, the development of efficient algorithms that are easy for implementation and provide satisfactory performance are required. To achieve the real-time control evaluation of these algorithms, user accessible high performance robot controllers need to be employed.

On-line trajectory adaptation is essential for advanced external sensor guided operations. It provides the basis for a robot system to achieve flexible interactions with the workcell environment. As most of the existing trajectory generating schemes appear to be convenient only for generating fixed paths and for off-line trajectory generations, an effective strategy for handling the real-time trajectory adaptation issue is required to achieve external sensor guided robot dynamic tracking and intercepting operations.
CHAPTER 3
DESIGN AND DEVELOPMENT OF AN OPEN ARCHITECTURE ROBOT CONTROLLER

3.1 Introduction

Attempts to improve the performance of robot systems have commonly generated the requirement for improving robot controller systems. It has been observed that modern robotics profoundly relies on computational capabilities: robot task planning, trajectory generation, sensory data processing and robot control all involving computation-intensive algorithms. It is not surprising that many schemes for developing sophisticated robot controllers are based on multi-processor configurations, since the computational requirement of modern robotics, especially when some external sensors and complicated dynamic control algorithms are involved, is much beyond the capacity of a single state of the art microprocessor. To take full advantage of the techniques of robot calibration, external sensor integration and advanced dynamic control algorithms, practical robot controller architectures with substantial computing power, easy external sensor integration and convenient user control algorithm implementation are required.

In this chapter, the design and development of an open architecture robot controller is described. The main objective of the controller design is to build a robot control platform that provides a powerful control algorithm test-bed and enables the integration of external sensors so that advanced robotics applications can be realised. The controller hardware interface has been designed to have a general form. It assumes that the robot joint position sensors are incremental encoders or any other type of sensors that...
output linear voltage signals, and that the robot joint actuators are driven by analog voltage signals buffered by appropriate external power amplifiers. This hardware interface implementation enables the controller to be used for a number of manipulator types. To take advantage of an existing PUMA 560 robot system, special interface boards have been designed, enabling the open architecture robot controller to drive the PUMA manipulator through the motor drive amplifiers and their associated hardware of the existing PUMA manipulator controller. This has successfully avoided the costly work of design and development of motor drive amplifiers and their power supply units in establishing the robot control platform that consists of the open architecture robot controller and the PUMA 560 manipulator.

3.2 System design considerations

Robot control is a coordinated process that requires the controller to perform various data manipulations and hardware related operations concurrently at a high speed. With modern robotics claiming a considerable computational requirement in its applications, the proper design of a controller architecture plays an important role in successfully building a robot controller.

The application environment of the open architecture robot controller requires the controller system to be able to provide substantial computational power for the fusion of external sensors and the implementation of advanced robot control algorithms. Additional requirements include that the controller hardware interface should be general enough to be used for a number of robot manipulator types although the current application is based on a PUMA 560 manipulator. In order to achieve this, a number of key issues in the controller design are discussed in this section.
3.2.1 Computational requirement

All robot controllers are built upon specially tailored computing machines. These computing machines frequently comprise of a number of microprocessors to offer the computational power required by the controller operations. An ideal robot controller system should provide sufficient real-time computational power for advanced control algorithms and external sensor based applications. It is obvious that the computational requirement may vary from case to case, depending on the applications and the algorithms employed.

A good indication of the representative computational requirement for current inverse dynamics based control algorithms can be found in the work of Lee and Chang (1986). The estimated computational requirement for each dynamic evaluation of a six degrees of freedom manipulator is about 1500 floating-point operations. If the dynamic control cycle time is set to be 1 millisecond, the resultant computational requirement for the dynamics evaluation is 1.5 million floating-point operations per second (MFLOPS). Taking into consideration in each control cycle time the share of other activities, such as joint sensor samplings, conversions, scaling and coordinate transformations, the computational requirement can easily be doubled or even tripled. Based on this estimation, it is considered that the open architecture robot controller should provide a computational power of more than 15 MFLOPS to cover a wide range of applications.

The precise assessment of the computational requirement of a robot application is extremely difficult. An equally difficult task is to state an upper bound on the required computational power, since innovative control algorithms and new sensing technology are still being developed. Thus it is desirable to have a system that can be expanded to meet far-
reaching computational demands. This observation suggests the use of an expandable multiprocessor architecture is advantageous.

3.2.2 Joint actuator position measurement

The control of the robot manipulator is equivalent to the control of the joint actuators. To form a basic feedback control loop, the position of the joint (or the joint actuator) needs to be measured. The most commonly used joint (or joint actuator) sensors in robotics are incremental encoders and potentiometers although the actuators themselves can be of different types. Therefore, the open architecture robot controller’s joint position measurement interface assumes the sensor feedback comprises of incremental encoder signals or linear voltage signals or the combination of both.

![Figure 3.2.1 A servo motor based joint configuration](image)

In order to discuss some special requirement involved in the measurement of a robot’s joint position and the sensor interfaces, an illustrative example of a servo motor based robot joint configuration is
shown in figure 3.2.1. The joint configuration has a potentiometer sensor and an incremental encoder sensor. Its servo motor is connected to the joint through a gear train. If the gear train is ideal in some features, such as having a rigid contact and no backlash, the joint position displacement can be accurately calculated through the measurement of the joint actuator displacement as the two have a fixed relationship of the gear ratio. Most general purpose robot systems have had special designs, if gears are used, to ensure the joints and joint actuators have a near ideal fixed transmission, thus it is very often that the two have been viewed as equivalent for position measurement purpose and have been addressed indistinguishably in many cases.

**Potentiometers**

Precision potentiometers are simple devices for obtaining rotary or translational joint position information. Basically, a precision potentiometer consists of a resistive element with a movable arm, or slider, in contact with the element. As the slider moves, the resistance varies between the end of the resistive element and the slider, indicating the position change. Potentiometers are absolute position sensors. They can be excited with alternating or direct current and the resultant outputs can easily be converted to linear voltage signals. The commonly used single-turn precision potentiometers have a rotation that is usually limited to 350 degrees. It provides a typical resolution at the scale of 0.05%, which is around 0.175 degrees for a rotary span of 350 degrees.

Potentiometers can only provide limited resolution for robot joint position measurement. Additional problems associated with them are noise and the slider wearing out. Thus they are rarely used as the sole joint sensors in robot systems.
Incremental encoders

Optical encoders are widely used in machine tools and robots to provide the position measurement of both linear and angular movement. There are two types of optical encoders: one type provides the absolute position information, hence called the absolute encoder, while the other provides solely incremental signals for recording the position displacement, called the incremental encoder. Incremental encoders are used far more commonly than absolute encoders, as they can be manufactured with much higher resolutions at a lower cost. The only advantage of an absolute encoder over an incremental encoder is that it directly provides the position information without any need for an external reference or start-up resetting. This, however, is frequently outweighed by the consideration of cost, resolution and interface issues for applications.

Figure 3.2.2 Incremental encoder signals
Incremental encoders generally provide two signals which have a phase difference of 90 degrees of electrical angle between them (see figure 3.2.2). The phase difference provides directional information of the underlying motion, that is, when signal A leading signal B stands for the sensed translation or rotation being in one direction, then signal B leading signal A reflects the reverse. For many rotary incremental encoders, an additional index signal C is frequently provided to mark the start (end) of a revolution.

Both signal A and signal B can be directly fed to an up/down counter to record the position displacement with the counting direction correctly controlled by a directional signal that is derived from the phase difference between them. The absolute position can then be determined by adding the recorded displacement on the initial start-up position. To better use the two signal channels, higher resolution can be achieved through the quadrature technique. By employing the 90 degrees phase difference in electrical angles of the two signals A and B, the quadrature technique generates four counting pulses every 360 degrees electrical angle of signal A or B, thus providing signal D (see figure 3.2.2) with a resolution as high as four times of the original channel signals. If a rotary incremental encoder outputs 1000 pulses per revolution, for example, the effective resolution of the shaft angle obtainable via the quadrature technique is 0.09 degrees.

The features of high resolution, low cost and relative simple interface requirement are the major advantages of using incremental encoders for robot joint position measurement. The most common problem associated with the use of incremental encoders in robot systems is that they need extra means to establish the start-up position so that the recorded joint position displacement can be correctly converted to the absolute joint position measurement. As incremental encoders are often used in places
where high resolutions are required, the techniques used to establish the start-up joint position must provide the same level measurement resolution to meet the performance requirement of many robot systems. A detailed discussion about some techniques and procedures of achieving appropriate incremental encoder start-up resetting is presented in Appendix A of this thesis.

3.2.3 Driving the joint actuator

The types of commonly used robot joint actuators are diverse, from various electrical motors to pneumatic and hydraulic drives requiring different power driving signals for controlling them. Even for actuators of the same type, say DC servo motors, the drive current and voltage needed are dependent on the size and the particular products; no one solution is appropriate for all of them. Therefore, it is considered impractical to include the actuator power amplifiers as part of the actuator driving interface of a robot controller that is designated for a class of robots. The important requirement then becomes how to drive these power amplifiers and to specify the interface signal form before these power amplifiers.

In essence, power signals for driving various actuators can be categorised into two basic types: the logic control signal type and the linear control signal type. Logic control signal type actuators only require the power signals to be in two different states. This can easily be achieved by using power relays which can be driven by a simple digital signal interface.

Linear control signal type actuators require the power signals to be able to vary from one extreme to the other with infinite states (or a very large number of states) between the two extremes. In general, two standard techniques for supplying such power signals are commonly used: linear amplifiers and pulse-width-modulated (PWM) amplifiers. Each has
advantages, but both are controlled by a simple analog voltage signal. To provide the analog voltage control signal, a digital-to-analog converter (DAC) interface must be employed in a computer-based digital controller.

### 3.2.4 Interface to a Unimation Mark III controller

In order to provide a robot operation platform, an existing PUMA 560 robot manipulator is used, together with the open architecture robot controller, to form a robot system for the sensor fusion work outlined in this thesis. The PUMA 560 manipulator is a six degrees of freedom (DOF) robot arm. It is representative of a large and popular class of modern industrial manipulators.

The PUMA 560 manipulator uses six geared DC servo motors with both encoder and potentiometer position feedback elements to drive the six joints. The actuators are integral packages that contain four basic components: a DC servo motor, an electric brake, an optical incremental encoder, and a geared-down potentiometer. Each of the joints has a similar configuration as the one illustrated in figure 3.2.1. The control input signals to the manipulator are the currents that activate the motors and the electric brakes; while the encoder and the potentiometer signals are the manipulator’s position feedback output.

To enable the open architecture robot controller to control the PUMA 560 manipulator, appropriate power amplifiers and their associated power supply unit are required to fill in the gap between the controller control output signals and the manipulator power signals that activate the motors and the brakes. There are two options for the work described in this chapter, i.e. either to design and develop a power amplifier sub-system for the manipulator, or to make use of the power amplifier sub-system of an existing Unimation Mark III controller that has been supplied as part of the
original PUMA robot system. Because of the adverse impact on the building up time and cost that are involved in the first option, it is considered that employing the power amplifier sub-system of the existing Unimation controller is a more appropriate approach.

The desirable situation, where the power amplifier sub-system of the existing Unimation controller is considered to be most sensibly used, is to 'share' the power amplifiers between the open architecture robot controller and the Unimation controller. This can be achieved by design a special switch-interface that operates under the control of a selection control signal. By inserting the switch-interface into a proper signal connection point within the Unimation controller, the actual control signals to the power amplifiers, which in turn output power signals to the manipulator, can be chosen as the DAC (digital-to-analog-converter) output signals of the robot controller specified by the selection control signal. If the switch-interface can further handle the manipulator position feedback signals properly, then the internal manipulator control loop of the Unimation controller can effectively be masked off, freeing the power amplifiers to be used by the open architecture robot controller to drive the manipulator.

To avoid making any damage to the existing Unimation controller system, it is considered desirable to design the switch-interface, or simply the interface, in a way that requires no hardware modification to the Unimation controller. An additional functionality desired is that the interface should retain the Unimation controller in a ready to run status when the PUMA 560 manipulator is controlled by the open architecture robot controller via the former's power amplifiers, enabling that the control of the manipulator can be switched between the two controllers in a safe and effortless approach without powering down the system. This will provide convenience for the open architecture robot controller's
development work, especially at the stages of hardware test and initial software development, as the Unimation controller can be employed to offer whatever manipulator motions that are required by the work in hand.

3.2.5 Summary

The application environment of the open architecture robot controller requires the controller system to be able to provide substantial computational power for the fusion of external sensors and the implementation of advanced robot control algorithms. To achieve this, and also to leave room for future expansion, the underlying computer system should be based on a high performance multi-processor architecture featuring easy programming and flexible hardware configuration.

In order to cover commonly used different robot products, the controller should have a position sensor signal interface capable of accommodating incremental encoders, potentiometers and other linear voltage signal sensors as well as their combinations. To make better use of the encoder sensors, the quadrature technique should be employed and the special requirement of the start-up resetting of incremental encoders should be supported at the interface. The controller output should provide both digital I/O interface and DAC signals.

To take advantage of the existing PUMA 560 robot system, a special interface to the Unimation Mark III robot controller needs to be designed and built, enabling the open architecture robot controller to make use of the former's power amplifiers and their associated hardware for the control of the PUMA 560 robot manipulator.
3.3 Hardware system

3.3.1 Architecture of the controller

In order to meet the application requirement of the open architecture robot controller, it is considered vital that the controller has flexibility and scalability in addition to the support of parallelism. This has resulted in selection of the hardware configuration for the controller as shown in figure 3.3.1.

![Figure 3.3.1 Configuration of the open architecture robot controller](image)

The underlying hardware of the controller consists of a MC68030 single board computer (SBC), a high performance Intel-i860 vector processor transputer (Inmos transputer module), a transputer array of eight T805 transputer transm and a T801 transputer based VME board (IMS-B016) that has four transputer links for network communication and provides a dual-ported memory (DPM) interface between the transputer network and the VMEbus data connection. Since the i860 processor transm communicates with other
processors through its accompanying I/O support transputer (a T800 processor) integrated within the tram, it can virtually be viewed and handled as a much more powerful transputer unit in the transputer network. The controller combines the MC68030 SBC and the transputer network to provide the computational power required by advanced robotics applications. The Sun Workstation in figure 3.3.1 is merely used as a software development environment.

The controller architecture has adopted a standard bus structure, the VMEbus, as its system integration backbone. However, unlike other bus-based tightly coupled multi-processor robot controller architectures, the use of the VMEbus in the open architecture robot controller is not primarily aimed at providing the medium for implementing the intercommunications among the multiple-processors; instead, it is adopted because of the advantages that a VMEbus based system can offer. These advantages include the relative high ratio of performance against the cost, short system development time, wide range of commercially available peripheral boards and easy system integration. As the activities over the VMEbus in the open architecture robot controller only cover the accesses to the peripheral boards and the dual ported memory that supports the intercommunications between the MC68030 SBC and the transputer network interface, the bus contention problem is very unlikely to happen.

The main reason for the combination of a conventional microprocessor based SBC with a transputer network within the hardware configuration is to separate the user accessible open computational resources (the transputers), which are required for implementing user control algorithms and for integrating external sensors, from an operating and safety monitoring core supported by the MC68030 SBC. This is advantageous due to the fact that, in general, a tram based transputer system
is hooked onto a host system (in the case here the Sun workstation), and does not provide asynchronous single node resetting and code loading functionality to enable a user to reload parts of his/her code without resetting the network. In addition, the separation of the operating and safety monitoring core from user accessible resources provides much higher system safety credibility. Furthermore, transputers in architecture are rather limited in their capacity for handling multi-external interrupts. Therefore the incorporation of a MC68030 SBC into the hardware configuration provides the extra convenience for system I/O programming.

Control of the robot manipulator is achieved through the sensor interface and the actuator interface. These interfaces comprise of a number of peripheral boards that are managed by the MC68030 single board computer. By adopting the device driver concept in the software design, general users are isolated from the particular hardware handling of the input/output interface boards.

The transputers and the i860 vector processor in figure 3.3.1 provide the high computational power for implementing sophisticated robot control algorithms and sensor fusion requirement. Transputers are high performance microprocessors. They also provide the added advantages that their interconnections are directly supported at both the hardware and software levels. With the four on chip independent high speed serial communication channels, called the transputer links, transputers can be conveniently and flexibly interconnected to form a much more powerful computing machine. On top of the computational power offered by the transputers, the i860 vector processor in the open architecture robot controller provides an impressive data processing enhancement. Roughly speaking, the i860 processor used in the open architecture robot controller is 20 times as fast as a T805 transputer in terms of the peak rate of floating
point data processing. If judging simply by summing up the peak rate of the data processing power, the number of the controller's total data processing capability is around 110 MFLOPS. This, however, should be treated very conservatively as it is extremely rare that a problem solving code, when written in a high level language, can satisfy the peak rate performance condition. A more sensible figure for judging the computational power might be the manufacturer's specification on the sustainable rate. Even so, the practically achievable figure is still hard to establish as it depends on the ram memory speed, compiler used, programming skill, computational burden distribution, etc. In order to give a rough idea of the available computational power, it is considered reasonable to assume that the application code programmed by a C programmer of average skill can reach 20%–30% of the peak rate of the underlying multi-processor system. This gives a computational power of 20–30 MFLOPS, well enough to meet the requirement of the intended applications of the controller. If in any case an even higher computational power is required, the current configuration can either be expanded by adding more transputers or be exploited by experienced programmers to increase the achievable percentage figure.

Apart from the high performance and easy expansion, an additional advantage of adopting the transputer network in the open architecture robot controller is that it provides system reconfigurability. This is an important feature since various control algorithms may favour different interconnection topologies between the computing units. Furthermore, the multiprocessor programming issue in a transputer network is much easier than its standard bus-based counterparts. Therefore, the adoption of the transputer network in the open architecture robot controller will generally ease the job of coding applications.
3.3.2 Interface design and implementation

It is considered beneficial that the open architecture robot controller should have its manipulator interface hardware to be able to cover a number of general purpose robot products, i.e. any robot type that has

a) an incremental encoder position sensor sub-system, or a linear voltage position sensor sub-system, or the combination of the both;

b) linear manipulator driving signals, or digital driving signals, or the combination of the both.

With this requirement in mind, the controller manipulator interface has been designed and implemented with an in-house made encoder interface board, a fast ADC input board, a high speed DAC output board and a general digital I/O interface board. An additional switch-interface has been designed and built to enable the open architecture robot controller to drive the PUMA 560 manipulator via its accompanying Unimation controller's power amplifiers.

**The encoder interface board**

The special requirement of quadrature detection and the start-up resetting in robot systems has made it hard to find a commercially available product to form the incremental encoder interface. This has generated the requirement of design and development of an incremental encoder interface board for the open architecture robot controller. Figure 3.3.2 gives a functional description of the in-house made encoder interface board.

The encoder interface board consists of three major parts: the interface control and on-board device control logic circuitry; encoder counter array and their associated auxiliary circuitry; and the digital input/output...
The interface control logic circuitry decodes the address and the control signals that are presented at the VMEbus connector. If the address matches with the on-board device address space and the bus operation is of a 16-bit data read or write, a hit condition is met and the interface control logic circuitry generates a board selection signal and the on-board device control logic circuitry is activated, which in turn enables the corresponding data read/write operation to be undertaken on the particular device selected by the address of the VMEbus operation. After the read/write operation has been accomplished, a timing component within the interface control logic circuitry activates the DACK* (data acknowledge) signal, informing the bus master to terminate the present VMEbus read/write operation.

Figure 3.3.2 Incremental encoder interface board block diagram.
The encoder counter array shown in Figure 3.3.2 primarily consists of six HCTL-2016 quadrature decoder and counter interface ICs and their auxiliary circuitry. A HCTL-2016 has a 16 bit binary up/down counter and features high operation speed, high noise immunity, quadrature detection and 8 bit tristate interface. In order to improve the data transfer efficiency, the on-board device control logic circuitry has been designed to be able to autonomously generate two internal read actions on a single VMEbus read request, if the read operation is designated to the HCTL-2016 array, so that the 16-bit contents of a HCTL-2016 is fetched via only one 16-bit VMEbus read operation. The auxiliary circuitry of the encoder counter array comprises of a start-up resetting control array and a status register. They provide the required functionality for incremental encoder start-up resetting procedures.

The encoder interface board has an 8-bit digital output port and an 8-bit digital input port. Additional two digital output signals, one reflecting the status of a keyed-switch and the other being the output signal of an operation watchdog, are also provided. All these digital signals can be used for general purpose.

In achieving the control of the PUMA 560 manipulator, the digital input and output lines of the encoder interface board have been employed to interact with the specially designed switch-interface. The output signal of the keyed-switch is used as the selection control signal to the switch-interface. The watchdog, which is implemented by using a software retrigergerable monostable register, has been employed to monitor the software running in the open architecture robot controller. This is achieved by connecting the watchdog output signal to the PUMA manipulator's arm power switch. Thus in the event of a controller software crash, the robot arm power will be automatically switched off.

PhD. 1994 Nongji Chen
The encoder interface board is a high speed channel for robot joint position feedback information. It can accept incremental channel signals up to the frequency of 3 MHz (12 MHz if measured in quadrature signal). The VME access time is 275ns to 400ns (best case and worst case respectively), calculated from the signal DSA* (data selection active) becomes active to the DACK* (data acknowledge active) becomes active. Figure 3.3.3 shows a picture of the encoder interface board.

The ADC input board

The ADC board is chosen as the MS-AD12H from the Matrix Corporation. The board features:

- 12-bit resolution with 0–10V, ±5V, and ±10V input range;
- 16 signal input lines, configurable as 16 single-ended or 8 differential inputs;
- Software programmable gains;
Chapter 3: Design and Development of an Open Architecture Robot Controller

- High conversion speed (12µs converter) with VMEbus interrupt capability.

The board uses an analog multiplexer to choose one out of the 8 channels (or 16 if single ended mode is used) as the active channel for analog signal conversion. It is fast enough to offer the control of a six DOF robot with a sampling rate of 1KHz if the linear voltage sensor signals are used as the primary position feedback. In applications, the differential mode of the ADC board is recommended as it provides high common-mode noise rejection. The board has a VMEbus access time of 430ns (maximum) measured from the VMEbus signal DSA* (data selection active) becomes active to the board signal DACK* (data acknowledge active) becomes active.

The DAC output board

To drive robot actuators that require linear power signals, DAC outputs must be provided by a digital controller system. This digital to analog interface in the open architecture robot controller is implemented by an AVME9210 analog output board from Acromag Inc.

The AVME9210 board is a VMEbus slave device supporting both 16-bit data and 8-bit data operations. The board has 8 independent output channels, each with 12-bit resolution. The output voltage signals of the board can be specified as between one of the range pairs of ±10V, ±5V, ±2.5V, 0-5V, and 0-10V. This provides flexibility to cover the amplifier's input signal range in applications.

The AVME9210 board has adopted an individual DAC per channel design approach. Consequently it is able to exploit the maximum signal accuracy attainable for the underlying DAC converter ICs and to minimise the output signal transits. The board has a settling-up time for the analog
signal generation of 1.2\mu s for a 5V-step change, and 6\mu s for a 20V-step change. Its VME access time is 390ns (Maximum, measured from DSA* to DACK*). Therefore it can be used to achieve very high bandwidth control in applications.

**The digital input/output board**

The digital input/output board is included in the open architecture robot controller as a general purpose digital signal interface. Its usage in the system is not pre-specified. The board is a MS-PIM parallel interface module manufactured by Matrix Corporation. It features:

- Four 8-bit programmable input/output ports with 48mA drive capability;
- Two independent 24-bit timer/counter units;
- VMEbus interrupt capability.

**The Unimation controller switch-interface**

The open architecture robot controller makes use of the motor amplifiers and their associated power supply unit of the existing Unimation Mark III robot controller in its control of the PUMA 560 manipulator. This is achieved by employing a specially designed switch-interface that is able to mask off the internal control loop of the Unimation controller, freeing the amplifier sub-system to be used by the open architecture robot controller. Figure 3.3.4 uses a simplified single joint control case to illustrate the fundamental role of the switch-interface.

In figure 3.3.4, the upper half of the drawing illustrates the servo control signal flow loop of the Unimation Mark III robot controller. As the figure shows, the DAC signal of the digital servo controller is directly forwarded to the power amplifier through a connector and the joint
position feedback signals are the inputs to the arm cable board. When the
switch-interface is placed (see the lower half of the drawing), both the two
connections have been replaced by signals coming from the switch-
interface, enabling the joint motor control loop to be closed via the external
ccontroller.

Figure 3.3.4 Functional description of the switch-interface.

Apart from the switching of the DAC signals and the manipulator
position feedback signals, the switch-interface has been designed to have
control over the amplifier power supply, the Unimation controller's main
supply, the manipulator break, etc. These functionalities put the open
architecture robot controller (or an external controller) in full charge of
driving the PUMA 560 manipulator if the selection control input line of the
switch-interface is set as external controller active. The selection control line
is a special digital control signal to the switch-interface. It determines the working mode of the interface, i.e. external controller active or internal controller (Unimation controller) active. When the selection control signal is set as internal controller active, the Unimation controller system works normally.

Special care has been taken in the design of the switch-interface so that no hardware modification to the Unimation controller is required for its operation and installation. The interface actually consists of two in-house made electronic boards and both are mounted in the Unimation controller chassis with standard spacers. Figure 3.3.5 shows a picture of the switch-interface installed in the Unimation Mark III controller chassis, the top one layer are the two boards which forms the switch-interface.

Figure 3.3.5 The switch-interface in the Unimation controller chassis.
3.3.3 System integration

By employing the switch-interface, which enables the open architecture robot controller to drive the PUMA 560 manipulator through the power amplifier sub-system of the Unimation Mark III controller, the integration of the robot control platform is straightforward. This is attributed to the adoption of the standard VMEbus 'backbone' and the controller interface implementation. Figure 3.3.6 illustrates the integrated robot control platform.

Figure 3.3.6. Illustration of the robot control platform.

The selection control signal to the switch-interface plays an important role in the integrated platform, as it determines the working mode of the switch-interface circuitry. Consequently it has been treated specially in the
system integration and the signal is wired to the output of the keyed-switch unit of the encoder interface board (reference figure 3.3.4 & 3.3.5). This provides the open architecture robot controller with the overall system control capability. In fact, the open architecture robot controller has two working modes, i.e. the controller mode and the monitor mode, in the integrated platform. If the keyed-switch is turned to the controller mode, the open architecture robot controller controls the operation of the PUMA manipulator. On the other mode, it can only monitor the manipulator's operations, leaving the manipulator to be controlled by the Unimation controller.

3.4 Software system

The proper operation of the open architecture robot controller requires that the underlying controller system software can support a number of concurrent operations. This multi-tasking requirement reflects the fact that a complete robot control system must include all of the aspects involved in moving a robot, not simply the algorithms found in the classic robot control literature. An important issue in the controller software is the efficiency in dealing with devices and externally generated events. In general, the events handling must be undertaken within certain rigid timing constraints in order to achieve an acceptable system performance. This real-time nature imposes constraints on both the underlying operating system and the structure of the controller system software.

The structure of the open architecture robot controller system software is also intimately linked to the intended usage of the controller system and the architecture of the underlying hardware. As the main objectives in the development of the open architecture robot controller are to provide a convenient control algorithm test-bed and to enable the integration of
external sensors so that advanced robotics applications can be realised, user accessibility to the internal kinematic and dynamic control software modules is considered as a basic requirement. The justification for such an approach lies in the fact that sensor based robotics is still in its infancy and no standards on sensor-controller interaction levels and protocols have been established. Indeed, as many novel sensors and innovative control algorithms are being developed, it is wise not to constrain too much on the interaction levels, leaving sufficient user implementation freedom for accommodating advanced algorithms.

3.4.1 Software hierarchy in a robot controller

Viewing from a functional perspective, the system software of a robot controller generally consists of three levels. This simplified description of the software hierarchy is illustrated in Figure 3.4.1. In general, the top task/motion planning level runs asynchronously with regard to the other two levels. It provides the pure spatial information (path) about the intended robot motions of an application to the trajectory control level. The trajectory control level calculates the time needed for the path in accordance to a given speed profile and combines the obtained temporal information with the path to form a motion trajectory. This trajectory information is then forwarded by the trajectory control level to the servo control level in a coordinated way to affect a robot motion.

Trajectory control usually runs at a cycle time that is determined by the tradeoff between the path resolution and the computational burden. In order to achieve a smooth and continuous path movement, the trajectory control level should be synchronised, with a proper gearing down ratio, to the servo control cycle. Such a synchronisation provides the means for the servo control level to move the robot to the required position at the correct
time in accordance to the trajectory. The servo control level exerts the most stringent time constraints on the robot controller software. The general requirement of the servo control cycle time is that the smaller the value the better. In practice, however, the servo control cycle time is inevitably limited by the computer speed, the bandwidth of the sensor and the power amplifiers. It is common to see that the trajectory control cycle time is in the order of ten times higher than its servo control counterpart with the latter at the scale of around a few milliseconds in many current industrial robot systems.

Figure 3.4.1 Software hierarchy of a robot controller.

3.4.2 Overview of the controller software implementation

The open architecture robot controller system software is based on two different approaches in dealing with the three functional levels. The major criterion considered is the time constraints imposed on the different levels' activities by the operational requirement. As the design setup for the default trajectory control cycle time is 10 milliseconds, it is possible for the trajectory control level to be handled by some operating system based processes. For convenience, the software modules within the servo control level are referred to as the lower level modules. Any modules that belong to the top two levels are referenced as the higher level modules.

PhD. 1994 Nongji Chen
Chapter 3: Design and Development of an Open Architecture Robot Controller

The open architecture robot controller system software is built upon a real-time core which uses a servo control invocation thread implemented in an OS9 device driver to complement the real-time kernel of the OS9 operating system, see figure 3.4.2. The software consists of several prioritised concurrent processes supported by the OS9 operating system, an extended robot device driver that runs in the MC68030 system state and some low level control modules that run on the transputer network. In figure 3.4.2, the user state OS9 processes are used to support the higher level modules. These processes are assigned with different priorities according to the time constraints imposed on them by the overall requirement, and are time-sliced by the OS9 kernel according to a pre-emptive scheme (Dibble, 1992). Intercommunications between these processes employ both the shared memory scheme and the OS9 system supported mechanisms. The latter includes pipe, event (multi-valued semaphore) and signal.

![Figure 3.4.2 Description of the controller software](image)

The controller software uses a user state process to provide a run-time user interface at the OS9 terminal. Users can run a compiled robot motion

PhD. 1994 Nongji Chen
control program, in a ‘move to position’ style, to fulfil a task through this interface. It also enables the user to issue single robot action commands or auxiliary commands, such as clear (clear one or all the named robot end effector positions), calibrate (carry out a robot encoder start-up resetting procedure), move (move to a named position), arm power on/off (erasing or applying joint brake system), list (list one or all the named end-effector positions), here (define a named end-effector position at current robot arm pose), speed (define robot motion speed), etc. By adopting the process approach, the run-time user interface can be run via a terminal or a terminal simulator at the local (RS232C connection) or a remote (Ethernet connection) site.

To achieve sensor fusion and the implementation of advanced control algorithms, it is considered important to support user definable kinematic or dynamic control functional modules. A special functional module, the application organiser, is designed to manage the mapping of motion planning and trajectory control modules to their user implemented counterparts that run on the transputers. The application organiser keeps a record of the current motion control pipe-line configuration and is responsible for the initialisation of the pipe-line if a new configuration is specified (i.e. a particular module has been mapped to the user defined one or mapped back to the default one). All the data exchange between the application organiser and the modules on the transputer side are via the dual-ported memory (DPM) and the handshake is accomplished through the intercommunication module switch registers (ICMS) of the IMS-B016 transputer board.

The controller software uses a dedicated robot device driver to handle and coordinate the real-time low level control activities. The main consideration for this approach is that the servo control level has much
more severe time constraints compared to the higher level modules. A typical servo control cycle time for the open architecture robot controller is considered as 500 micro-seconds. At this time constraint level, a process based implementation will simply waste a considerable portion of the processor time in activating the operating system kernel. Therefore a low invocation overhead alternative to the process based scheme is attractive.

3.4.3 The device driver

The dedicated robot device driver forms part of the core of the real-time system software. It runs in the system state of the MC68030 microprocessor and is responsible for providing a means of invoking time-critical servo control level modules and managing all the interface hardware activities involved in the control of the robot. The device driver uses a timer interrupt handler to invoke the servo control activities, figure 3.4.3. To guarantee the minimum interrupt latency, the timer interrupt request level is assigned at level 6 (the highest level of maskable interrupt request) and the interrupt handler is directly invoked without activating the kernel. This means the servo control interrupt handler can not use any system support of the OS9, a price worth paying since the servo control level normally does not need to directly signal anything to the higher level modules. If in any case an event needs to be signalled to a higher level module, a flag is set by the handler at the device driver static data section and the task will be fulfilled by a house-keeping module in the driver.

Several interface threads to modules that need to be run at the servo control rate have been provided by the servo control interrupt handler. They include threads for data acquisition, safety protection and servo control. In order to support the open architecture robot controller philosophy, the servo control thread is able to link to either the default PID
control module within the device driver or a user servo control module implemented over the transputer network.

![Diagram of extended device driver](image)

Figure 3.4.3 The extended device driver

The device driver provides a module that monitors the safety of the controlled robot. This is of importance since the Puma 560 robot does not have hardware safety protection mechanisms. Implemented measures include joint limit checks (by both the encoder recorded position value and the potentiometer output signal), joint control envelope violation checks, cable connection failure checks, and control software crash checks. The software crash check is achieved by employing the watchdog signal in the encoder interface board. To keep the watchdog signal alive, the safety module must reference a triggering address within a certain time interval. In the event of a controller software crash, the robot arm power will be switched off and the brake applied.
Servo control synchronisation

As previously mentioned, the real-time servo control module can be chosen as the default PID control module in the device driver, or, as a module implemented on the transputer network. When the latter is chosen, the synchronisation issue arises.

The synchronisation issue becomes prominent because of the stringent time constraints on the servo control module, the trajectory control and servo control timing relationship, and the architecture differences between the MC68030 micro-processor and the transputers. Unlike the transputers, the MC68030 does not have a link hardware and link communication instructions to synchronise with other processor units. To overcome this difficulty, some interrupt driven interactions between the OS9 device driver and the transputer network have been devised. They are established on the VMEbus interrupt capability supported by both the MC68030 SBC and the IMS-B016 board.

![Diagram](image_url)

Figure 3.4.4 Servo control synchronisation interaction.
Chapter 3: Design and Development of an Open Architecture Robot Controller

The interactions take place between the transputer servo control interface module in the robot device driver (see figure 3.4.3) and a high-priority process server of the T801 transputer on the B016 board. As is illustrated in figure 3.4.4, the transputer servo control interface module first places the robot joint angles and other servo control relevant information into the dual-ported memory provided by the B016 board. Then, it generates an interrupt signal to trigger the transputer servo control server which in turn invokes a user implemented servo control module over the transputer network. When the servo control evaluation is finished, a VMEbus interrupt is generated by the servo control server to signal back to the device driver. These interactions synchronise the servo control calculation operations over the transputer network to the control sampling rate determined by the servo control timer of MC68030 SBC. If in any case that the servo control evaluation fails to finish before the start of the next servo control cycle, an error is signalled to the run-time user interface at the OS9 terminal and the robot arm power is switched off.

3.5 Summary

The open architecture robot controller is a multiprocessor based advanced robot controller. It employs a high performance transputer network (including an i860 vector processor based transputer) and an MC68030 SBC to provide the computational power required by modern robotics. The architecture of the controller is a departure from previously developed robot controller systems described in Chapter 2. This has resulted in the controller featuring easy expansion in both computational power and peripheral hardware. Therefore it provides convenience for sensory hardware inclusion as well as in achieving some special system configuration required by advanced robotics applications.
Chapter 3: Design and Development of an Open Architecture Robot Controller

The implementation of the controller interface has considered the applicability to a wider range of robot manipulators. This is reflected in the sensor and robot manipulator driving signal forms and the special handling of the encoder start-up resetting in the in-house made encoder interface board. A special switch-interface has been designed and developed, enabling the open architecture robot controller to drive a PUMA 560 manipulator via an existing Unimation controller’s power amplifiers. This has successfully avoided the costly work of developing an appropriate power amplifier subsystem for the controller-manipulator platform.

The controller system software provides a default operational environment for general users of the open architecture robot controller. User applications can be programmed in a 'move to position' style and the compiled C program can be launched at a run-time user interface. For research users, the system software has the capability of linking to user defined kinematic control or dynamic control modules, enabling the user to modify the default robot control loop for the fusion of a wide range of sensors and the implementation of advanced robot control algorithms such as adaptive controllers, variable structure controllers, fuzzy logic controllers, etc.
CHAPTER 4

INVERSE KINEMATICS SOLUTION TO A CALIBRATED PUMA 560 ROBOT

4.1 Introduction

Establishing the inverse kinematics solution to a calibrated industrial robot is an important implementation issue in achieving performance improvement via calibration techniques. The benefits promised by a calibrated robot kinematic model only become a reality when the calibrated results are implemented in applications. Although current industrial robots have a simple nominal kinematic model that renders an analytical solution to the inverse kinematics problem, their calibrated counterparts, however, frequently fail to retain such simplicity as many nominally zero valued elements re-appear with a non-zero value in the calibrated kinematics equations. In general, numerical approaches have to be adopted to provide a solution to the inverse kinematics problem under such circumstances.

This chapter details the procedures solving the inverse kinematics problem of a Puma 560 manipulator employed to form the research platform of this work. After establishing an analytical inverse kinematics solution to the nominal kinematic model, a calibrated kinematic model of the employed Puma 560 manipulator identified by Stanton (1991) is described and a numerical algorithm that solves its inverse problem is derived. The proposed numerical method is simple, effective and suitable for real-time applications. It can be generalised to cover other calibrated industrial robots that have an analytical inverse solution to their corresponding nominal kinematic models.
4.2 Nominal kinematic model and its inverse solution

The Puma 560 manipulator is a six-axis industrial robot. It has been widely studied in the literature due to its wide availability. The Puma manipulator has a simple nominal structure which enables closed form inverse kinematics solution for the nominal kinematic model. Figure 4.2.1 shows a DH representation of the arm and its nominal parameters (Fu et al., 1987).

![Image of Puma robot arm with DH representation and parameters]

<table>
<thead>
<tr>
<th>Joint</th>
<th>$q_i$</th>
<th>$\alpha_i$</th>
<th>$d_i$</th>
<th>Joint range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>-160 to +160</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>451.5 mm</td>
<td>-225 to 45</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>90</td>
<td>-20.32 mm</td>
<td>-45 to 225</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-90</td>
<td>433.07 mm</td>
<td>-110 to 170</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>-100 to 100</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>56.25 mm</td>
<td>-266 to 266</td>
</tr>
</tbody>
</table>

Figure 4.2.1 Link coordinate system and parameters of a Puma robot

The most commonly used tool for describing the robot kinematic model is homogenous transformations. The DH convention of attaching a coordinate system to a link and its corresponding homogenous transformation formula is reviewed in chapter two section 2.3.1. Substitutes the parameters shown in figure 4.2.1 into each $A_i$ matrices defined in (2.3.1), the kinematic model of a Puma 560 manipulator is written as

$$T = A_1A_2...A_6 = \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = A_1A_2...A_6$$
where

\[ n_x = C_1[C_23(C_4C_5C_6 - S_4S_6) - S_23S_5C_6] - S_1(S_4C_5C_6 + C_4S_6); \]
\[ n_y = S_1[C_23(C_4C_5C_6 - S_4S_6) - S_23S_5C_6] + C_1(S_4C_5C_6 + C_4S_6); \]
\[ n_z = -S_23(C_4C_5C_6 - S_4S_6) - C_23S_5C_6; \]
\[ s_x = C_1[-C_23(C_4C_5C_6 + S_4C_6) + S_23S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6); \]
\[ s_y = S_1[-C_23(C_4C_5S_6 + S_4C_6) + S_23S_5S_6] + C_1(-S_4C_5S_6 + C_4C_6); \]
\[ s_z = S_23(C_4C_5S_6 + S_4C_6) + C_23S_5S_6; \]
\[ a_x = C_1(C_23C_5S_5 + S_23C_5) - S_1S_4S_5; \]
\[ a_y = S_1(C_23C_4S_5 + C_23C_5) + C_1S_4S_5; \]
\[ a_z = -S_23C_4S_5 + C_23C_5; \]
\[ p_x = C_1[d_6(C_23C_5S_5 + S_23C_5) + S_23d_4 + a_3C_23 + a_2C_2] - S_1(d_6S_4S_5 + d_2); \]
\[ p_y = S_1[d_6(C_23C_4S_5 + S_23C_5) + S_23d_4 + a_3C_23 + a_2C_2] + C_1(d_6S_4S_5 + d_2); \]
\[ p_z = d_6(C_23C_5 - S_23C_4S_5) + C_23d_4 - a_3S_23 - a_2S_2, \]

with the notation that

\[ S_i = \sin(q_i); \quad S_{ij} = \sin(q_i + q_j); \quad C_i = \cos(q_i); \quad C_{ij} = \cos(q_i + q_j). \]

The closed form inverse kinematics solution to the nominal Puma model can be derived by several approaches. A popular method is to isolate an unknown joint angle one at a time (Paul, 1981). This provides a chance of solving the inverse kinematics problem on a joint-by-joint basis. As there are multiple possible solutions corresponding to a single robot pose, it is convenient to use some arm configuration indicators to uniquely specify which one is the desired arm configuration of the Puma 560 manipulator.

To start with, we define three arm configuration indicators, i.e. ARM, ELBOW and WRIST, according to human arm geometry as the Puma 560 manipulator is anthropological in structure. The indicators bear a value defined as:
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

\[
\text{ARM} = \begin{cases} 
+1 & \text{RIGHT ARM} \\
-1 & \text{LEFT ARM} 
\end{cases} \quad (4.2.2)
\]

\[
\text{ELBOW} = \begin{cases} 
+1 & \text{ABOVE ARM} \\
-1 & \text{BELOW ARM} 
\end{cases} \quad (4.2.3)
\]

\[
\text{WRIST} = \begin{cases} 
+1 & \text{WRIST PLUS} \\
-1 & \text{WRIST MINUS} 
\end{cases} \quad (4.2.4)
\]

with

RIGHT ARM: Positive \( q_2 \) moves the wrist in the positive \( z_0 \) direction while joint 3 is not activated;

LEFT ARM: Positive \( q_2 \) moves the wrist in the negative \( z_0 \) direction while joint 3 is not activated;

ABOVE ARM: Position of the wrist of the RIGHT/LEFT arm with respect to the shoulder coordinate system has negative/positive value along the \( y_2 \) axis;

BELOW ARM: Position of the wrist of the RIGHT/LEFT arm with respect to the shoulder coordinate system has positive/negative value along the \( y_2 \) axis;

WRIST PLUS: Joint 5 value > 0;

WRIST MINUS: Joint 5 value < 0.

To facilitate the derivation of the inverse kinematics solution, the contribution of \( d_6 \) in figure 4.2.1 towards equation (4.2.1) can be firstly excluded without any side-effect. This can be done by minus a position vector \( d_{6a} \) from \( T \) which leads to \( T_c \):

\[
T_c = \begin{bmatrix} 
    n_x & s_x & a_x & (p_x - d_{6a}x) \\
    n_y & s_y & a_y & (p_y - d_{6a}y) \\
    n_z & s_z & a_z & (p_z - d_{6a}z) \\
0 & 0 & 0 & 1 
\end{bmatrix} = A_1A_2 \ldots A_5 A_6 = \begin{bmatrix} 
    n_x & s_x & a_x & p_{cx} \\
    n_y & s_y & a_y & p_{cy} \\
    n_z & s_z & a_z & p_{cz} \\
0 & 0 & 0 & 1 
\end{bmatrix} \quad (4.2.5)
\]

where

PhD. 1994 Nongji Chen

102
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

\[
A_6' = \begin{bmatrix}
C_6 & -S_6 & 0 & 0 \\
S_6 & C_6 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix};
\]

\[
P_{cx} = p_x - d_6 a_x = C_1 (S_2 S_3 d_4 + a_3 C_2 + a_2 C_1) - S_1 d_2;
\]

\[
P_{cy} = p_y - d_6 a_y = S_1 (S_2 S_3 d_4 + a_3 C_2 + a_2 C_1) + C_1 d_2;
\]

\[
P_{cz} = p_z - d_6 a_z = C_2 d_4 - a_3 S_{23} - a_2 S_2.
\]

Note that the position vector \( \mathbf{P_c} = (p_{cx}, p_{cy}, p_{cz})^T \) above points from the origin of the robot world coordinate system \((x_0, y_0, z_0)\) to the wrist centre where the last three joint axes of the Puma 560 manipulator intersect, and \(A_6'\) is \(A_6\) with the offset value \(d_6\) being replaced with a zero one.

From equation (4.2.5), it is easy to obtain

\[
(A_1)^{-1} \mathbf{T_c} = \begin{bmatrix}
C_1 & S_1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-S_1 & C_1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
n_x & s_x & a_x & p_{cx} \\
n_y & s_y & a_y & p_{cy} \\
n_z & s_z & a_z & p_{cz}
\end{bmatrix} = A_2 \ldots A_5 A_6'. \tag{4.2.6}
\]

Equating the (3,4) elements from both sides of (4.2.6), there is

\[
C_1 p_{cy} - S_1 p_{cx} = d_2. \tag{4.2.7}
\]

By squaring both sides of (4.2.7) and re-organising the elements, one can obtain

\[
(C_1 p_{cx} + S_1 p_{cy})^2 = (p_{cx})^2 + (p_{cy})^2 - (d_2)^2. \tag{4.2.8}
\]

Hence

\[
C_1 p_{cx} + S_1 p_{cy} = \pm \sqrt{(p_{cx})^2 + (p_{cy})^2 - (d_2)^2}. \tag{4.2.9}
\]

From (4.2.7) and (4.2.9), there are solutions for \(S_1\) and \(C_1\):

\[
S_1 = (\pm p_{cy} \sqrt{(p_{cx})^2 + (p_{cy})^2 - (d_2)^2} - d_2 p_{cx})/((p_{cx})^2 + (p_{cy})^2) \tag{4.2.10}
\]

\[
C_1 = (\pm p_{cx} \sqrt{(p_{cx})^2 + (p_{cy})^2 - (d_2)^2} + d_2 p_{cy}))/((p_{cx})^2 + (p_{cy})^2) \tag{4.2.11}
\]

Equation (4.2.10) and (4.2.11) lead to two possible solutions of \(q_1\) due to the different sign of the square-root in (4.2.9). They correspond to the two
different arm (left or right) configurations respectively. By combining the ARM indicator, $q_1$ is obtained as

$$q_1 = \arctan2\left(-\frac{p_{cy}}{p_{cx}} \sqrt{(p_{cx})^2 + (p_{cy})^2 - (d_2)^2 - d_2^2}, \frac{p_{cx}}{p_{cy}} \sqrt{(p_{cx})^2 + (p_{cy})^2 - (d_2)^2 + d_2^2}\right). \quad (4.2.12)$$

It should be noted that the inverse trigonometric function $\arctan2(y,x)$ is a variant of $\arctan(z)$ and usually is written with the arguments to be separated by a comma as the signs of $y$ and $x$ become important in determining the resultant angle in the range of $-\pi$ to $\pi$ (for instance, $\arctan2(y,x) \neq \arctan2(-y,-x)$). In order to achieve formula compactness in this section, the form of $\arctan2(y,x)$ has been replaced by $\arctan2(y/x)$ and it should still be understood as $\arctan2(y,x)$.

Having solved the value of $q_1$, the next step is to look at joint three. Equating the $(1,4)$ elements from both sides of (4.2.6) and doing the same to the $(2,4)$ elements, there are

$$C_1 p_{cx} + S_1 p_{cy} = d_4 s_{23} + a_3 c_{23} + a_2 c_2, \quad (4.2.14)$$
$$-p_{cz} = -d_4 c_{23} + a_3 s_{23} + a_2 s_2. \quad (4.2.15)$$

Squaring both sides of (4.2.14) and (4.2.15), adding the resulting equations and using (4.2.8) lead to

$$P = d_4(s_{23}c_{23} - c_{23}s_{23}) + a_3(c_{23}c_2 + s_{23}s_2) = -(d_4)s_3 + a_3 c_3, \quad (4.2.16)$$

where $P$ is defined as

$$P = \frac{(p_{cx})^2 + (p_{cy})^2 - (a_2)^2 - (a_3)^2 - (d_2)^2 - (d_4)^2}{2a_2}. \quad (4.2.17)$$

Noting that equation (4.2.16) is of the same form as (4.2.7), consequently there exist two solutions for $q_3$ and the procedures leading to them are exactly the same as those for $q_1$. Combining the arm configuration indicators into the solution expression, $q_3$ is obtained as
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

\[
q_3 = \tan^{-2} \left( \frac{-\text{ARM ELBOW} \sqrt{(d_4)^2 + (a_3)^2 - (P)^2} + d_4 P}{\text{ARM ELBOW} \ d_4 \sqrt{(d_4)^2 + (a_3)^2 - (P)^2} + a_3 P} \right). \quad (4.2.18)
\]

In order to isolate \(q_2\) from the unresolved \(q_4\), \(q_5\) and \(q_6\), both sides of (4.2.5) is left-multiplied with \([A_1 A_2 A_3]^{-1}\):

\[
\begin{bmatrix}
C_1 C_23 & S_1 C_23 & -S_23 & -a_2 C_3 - a_3 \\
-S_1 & C_1 & 0 & -d_2 \\
C_1 S_23 & S_1 S_23 & C_23 & -a_2 S_3 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\mathbf{n}_x & \mathbf{s}_x & a_x & p_{cx} \\
\mathbf{n}_y & \mathbf{s}_y & a_y & p_{cy} \\
\mathbf{n}_z & \mathbf{s}_z & a_z & p_{cz} \\
0 & 0 & 0 & 1
\end{bmatrix} = A_4 A_5 A_6. \quad (4.2.19)
\]

Equating the (1,4) elements from both sides of (4.2.9), as well as the (3,4) elements, there exist

\[
C_1 C_{23} p_{cx} + S_1 C_{23} p_{cy} - S_{23} p_{cz} - a_2 C_3 - a_3 = 0. \quad (4.2.20)
\]

\[
C_1 S_{23} p_{cx} + S_1 S_{23} p_{cy} + C_{23} p_{cz} - a_2 S_3 = d_4. \quad (4.2.21)
\]

Equations (4.2.20) and (4.2.21) have two unknowns \(S_{23}\) and \(C_{23}\). They can be directly solved, resulting \(q_{23}\) as

\[
q_{23} = \tan^{-2}\left(\frac{(a_3 - a_2 C_3) p_{cz} + (C_1 p_{cx} + S_1 p_{cy}) (a_2 S_3 + d_4)}{(a_2 C_3 + d_4) p_{cz} + (-a_3 - a_2 C_3) (C_1 p_{cx} + S_1 p_{cy})}\right). \quad (4.2.22)
\]

In turn the solution for joint 2 is found

\[
q_2 = q_{23} - q_3. \quad (4.2.23)
\]

Now the entire left side of (4.2.19) is known. Equating the (1,3) elements, together with the (2,3) elements as well, from both sides of (4.2.19), there are

\[
a_x C_1 C_{23} + a_y S_1 C_{23} - a_2 S_{23} = C_4 S_5. \quad (4.2.24)
\]

\[
- a_x S_1 + a_y C_1 = S_4 S_5. \quad (4.2.25)
\]

Provided \(S_5\) is not 0, \(q_4\) can be calculated as

\[
q_4 = \tan^{-2}\left(\frac{\text{WRIST}(-a_x S_1 + a_y C_1)}{\text{WRIST}(a_x C_1 C_{23} + a_y S_1 C_{23} - a_2 S_{23})}\right). \quad (4.2.26)
\]
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

The WRIST indicator in (4.2.24) specify whether the joint angle of \( q_5 \) is required to be within the range of 0 to \( \pi \) ( \( S_5 > 0 \)) or to be within the range of \(-\pi\) to 0 ( \( S_5 < 0 \)). When \( S_5 \) is zero, the Puma robot is at its singular place where the joint \( 4 \) and joint \( 6 \) are aligned with each other. Under such circumstances, \( q_4 \) can be set as any convenient value (0, for example) as the required orientation can always be reached by properly rotating joint 6.

By left-multiplying equation (4.2.5) with \([A_1A_2A_3A_4]^{-1}\) and equating the elements of (1,3) and (2,3) from both sides of the resulting equation, one can obtain

\[
(C_1C_23C_4 - S_1S_4) a_x + (S_1C_23C_4 + C_1S_4) a_y - C_4S_{23} a_z = S_5. \tag{4.2.27}
\]
\[
-C_1C_23 a_x - S_1S_{23} a_y - C_{23} a_z = -C_5. \tag{4.2.28}
\]

Hence

\[
q_5 = \text{atan2}(\frac{(C_1C_23C_4 - S_1S_4) a_x + (S_1C_23C_4 + C_1S_4) a_y - C_4S_{23} a_z}{C_1C_23 a_x + S_1S_{23} a_y + C_{23} a_z}). \tag{4.2.29}
\]

Further more, if one equates the elements of (3,1) and (3,2) from the resulting equation of (4.2.5) left-multiplied with \([A_1A_2A_3A_4]^{-1}\), there are

\[
(-C_1C_23S_4 - S_1C_4)n_x + (C_1C_4 - S_1C_23S_4)n_y + S_4S_{23} n_z = S_6. \tag{4.2.30}
\]
\[
(-C_1C_23S_4 - S_1C_4)s_x + (C_1C_4 - S_1C_23S_4)s_y + S_4S_{23} s_z = C_6. \tag{4.2.31}
\]

The solution for joint 6 is obtained as

\[
q_6 = \text{atan2}(\frac{(-C_1C_23S_4 - S_1C_4)n_x + (C_1C_4 - S_1C_23S_4)n_y + S_4S_{23} n_z}{(-C_1C_23S_4 - S_1C_4)s_x + (C_1C_4 - S_1C_23S_4)s_y + S_4S_{23} s_z}). \tag{4.2.32}
\]

As all the six joint values are now established in corresponding to a given pose, the inverse kinematics problem to the nominal kinematic model of the Puma 560 robot is resolved. The three arm configuration indicators ARM, ELBOW and WRIST have been used to choose a specified configuration from the eight possible solutions.
4.3 Inverse kinematics solution to a calibrated Puma 560 robot

Experience has shown that many current industrial robots have better repeatability compared to their accuracy (Mooring et al, 1991). This makes it possible to improve the robot accuracy through calibration techniques. Unfortunately, the simple nominal configuration structure of an industrial robot will generally not be kept when the calibrated kinematic model is derived. This invalidates its nominal closed-form inverse kinematic solution and a numerical approach has to be adopted to solve the inverse kinematics problem.

4.3.1 The calibrated kinematic model

<table>
<thead>
<tr>
<th>Link i</th>
<th>$\theta_i$ (deg)</th>
<th>$\hat{\theta}_i$ (m)</th>
<th>$b_i$ (m)</th>
<th>$\hat{b}_i$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>-90.09</td>
<td>0.2 e-03</td>
<td>0.187</td>
<td>-0.0493</td>
</tr>
<tr>
<td>2</td>
<td>-0.15</td>
<td>0.119</td>
<td>-156.709</td>
<td>156.709</td>
</tr>
<tr>
<td>3</td>
<td>90.21</td>
<td>-21.5 e-03</td>
<td>0.491</td>
<td>-37.1 e-03</td>
</tr>
<tr>
<td>4</td>
<td>-90.28</td>
<td>0.4 e-03</td>
<td>41.2 e-03</td>
<td>-58.1 e-03</td>
</tr>
<tr>
<td>5</td>
<td>90.23</td>
<td>-0.2 e-03</td>
<td>60.8 e-03</td>
<td>-41.3 e-03</td>
</tr>
<tr>
<td>6*</td>
<td>0.0</td>
<td>0.0</td>
<td>55.9 e-03</td>
<td>-60.8 e-03</td>
</tr>
</tbody>
</table>

* are nominal offsets; ** Original value is 0.493, it is caused by base definition.

Calibration of a robot involves the processes of robot position measuring, data processing and model parameter identification. The Puma 560 manipulator used in this work has previously been investigated for calibration purposes. The calibration work were conducted by using a laser triangulation measurement system OPTOTRAC (Mayer & Parker, 1988) developed at University of Surrey. In Table 4.3.1, the calibrated kinematic result based on an S-model identification process is listed (Stanton, 1991). The S-model is a 6 parameter kinematic description proposed by Stone et al (1986). It allows arbitrary positioning and orientation of the link frame on a
joint axis which enables the robot link parameters to be identified accurately by a set of simple decoupled identification problems. The S-model is inherently related to the commonly used Denavit-Hartenberg (DH) convention. Its parameters can easily be mapped into the DH counterparts. Table 4.3.2 lists the equivalent DH model parameters extracted from the calibrated result by using the inherent relationship between the S-model and the DH model (Stanton, 1991).

Table 4.3.2 Extracted DH model link parameters

<table>
<thead>
<tr>
<th>Link i</th>
<th>$\alpha_i$ (deg)</th>
<th>$a_i$ (m)</th>
<th>$d_i$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-90.09</td>
<td>0.2 e-03</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-0.15</td>
<td>0.119</td>
<td>156.89</td>
</tr>
<tr>
<td>3</td>
<td>90.21</td>
<td>-21.5 e-03</td>
<td>-156.74</td>
</tr>
<tr>
<td>4</td>
<td>-90.28</td>
<td>0.4 e-03</td>
<td>0.4331</td>
</tr>
<tr>
<td>5</td>
<td>90.23</td>
<td>-0.2 e-03</td>
<td>-0.1 e-03</td>
</tr>
<tr>
<td>6*</td>
<td>0.0</td>
<td>0.0</td>
<td>55.9 e-03</td>
</tr>
</tbody>
</table>

As can be seen in Table 4.3.2, the calibrated result has a much more complicated configuration structure than a nominal one under which the twist angles (as shown in column 2), for example, would be either 0 or ±90 degrees. With this increased complexity in the configuration structure, the inverse kinematics problem becomes difficult to resolve. In general, numerical approaches have to be adopted to tackle the inverse kinematic solution difficulties for calibrated kinematic models.

4.3.2 The inversion algorithm

In this section, an iterative numerical method solving the inverse kinematics problem of a calibrated industrial robot is derived. The basic idea of the method is to obtain a shifted pose for any given pose so that the former leads to an solution, based on the inversion of the nominal...
kinematic model, that 'coincides' with the inverse solution of the calibrated
kinematic model with regard to the given pose. The derivation of the
iterative formula is undertaken in a generic form.

Let \( f(c, q) \) be a homogeneous transformation function that represents
the nominal kinematic model of an industrial robot as \( f(c_0, q) \), and the
calibrated kinematic model as \( f(c_1, q) \), where \( c \in \mathbb{R}^n \) is the parameter variable
vector, \( q \in \mathbb{R}^n \) is the joint variable vector, and \( c \) and \( q \) are independent
variables. Since all the elements of a robot kinematic model comprise the
linear and sinusoidal form functions of the parameters and the joint
variables, it can be proven that \( f(c, q) \) is a continuous homogeneous
transformation function and has continuous partial derivatives up to an
infinite order with respect to both \( c \) and \( q \). The uniform description of the
nominal model and the calibrated kinematic model by \( f(c, q) \) implies that
the two models have the same zero point definition of the joint variable
vector \( q \), and that changes in the parameter vector \( c \) do not bring any
changes to the zero point definition of \( q \).

In the following discussions, the nominal kinematic model function
\( f(c_0, q) \) is assumed to have a closed-form inverse kinematic solution. Let the
inversion function is denoted by \( b(T) \), for a given pose \( T \), therefore

\[
T = f(c_0, q) \iff q = b(T), \quad (4.3.1)
\]

where \( b(T) \) is assumed to have had some arm configuration indicators
properly set so that it gives a unique inverse kinematic solution. Another
assumption about the two kinematic models \( f(c_0, q) \) and \( f(c_1, q) \) is that the
values of \( c_0 \) and \( c_1 \) can be related as \( c_1 = c_0 + \Delta c \), where \( \Delta c \) is a small
differential change of the parameter vector.

With the above assumptions, the pose shifting strategy for solving the
inverse kinematics problem of the calibrated industrial robot can now be
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

formulated. Given a desired robot pose $T_d$, find a shifted pose $T_s$ such that the nominal inverse kinematic solution obtained by

$$q_d = b(T_s) \quad (4.3.2a)$$

leads to the calibrated kinematic model satisfying

$$T_d = f(c, q_d). \quad (4.3.2b)$$

Therefore, $q_d$ is the intended inverse kinematic solution for the calibrated model $f(c, q)$.

The pose shifting strategy addresses the inverse kinematics problem of the calibrated kinematic model as an equivalent pose finding problem. As long as the accuracy of the obtained shifted pose $T_s$ is satisfactory, the intended inverse kinematics solution problem is virtually solved. Thus finding the corresponding $T_s$ of a desired robot pose $T_d$ forms the key to the solution of the inverse kinematics problem under discussion.

Consider the equation (4.3.2b). By neglecting the higher order items of the small parameter change $\Delta c$, the first order Taylor expansion of equation (4.3.2b) can be obtained as

$$T_d = f(c, q_d) = f(c_0, q_d) + \frac{\partial f(c, q_d)}{\partial c} \bigg|_{c=c_0} \Delta c = T_s + \frac{\partial f(c, q_d)}{\partial c} \bigg|_{c=c_0} \Delta c. \quad (4.3.3)$$

Rearranging equation (3), the intended $T_s$ can be approximated as

$$T_s = T_d - \frac{\partial f(c, q_d)}{\partial c} \bigg|_{c=c_0} \Delta c. \quad (4.3.4)$$

Expanding the last item of equation (4) by another first order approximation with respect to the change of the joint variable vector $q$, gives

$$T_s = T_d - \frac{\partial f(c, q_0)}{\partial c} \bigg|_{c=c_0} \Delta c - \frac{\partial}{\partial q} \left( \frac{\partial f(c, q)}{\partial c} \bigg|_{c=c_0} \Delta c \right) \bigg|_{q=q_0} \Delta q, \quad (4.3.5)$$

PhD. 1994 Nongji Chen
where \( q_o = b(T_o) = b(f(c_o, q_o)) \), and \( \Delta q = q_d - q_o \). Defining \( T_1 = f(c_1, q_o) \), and by considering the first order approximation, there exists

\[
T_1 - T_d = f(c_1, q_o) - f(c_o, q_o) = \left. \frac{\partial f(c, q)}{\partial c} \right|_{c=c_o} \Delta c.
\]

(4.3.6)

Introducing (4.3.6) into (4.3.5), the latter can be rewritten as

\[
T_s = 2T_d - T_1 - \left. \frac{\partial f(c, q)}{\partial q} \right|_{c=c_o} \Delta c \bigg|_{q=q_o} \Delta q.
\]

(4.3.7)

The third item in equation (4.3.7) contains the intended solution vector \( q_d \) in \( \Delta q \) and is not directly available. To avoid this difficulty, an initial estimation of the \( T_s \) is evaluated through (4.3.7) by omitting the third item (which is virtually of a second-order nature):

\[
T_s = T_s^1 = 2T_d - T_1.
\]

(4.3.8)

Then an estimated inverse kinematic solution \( q_1 \) can be obtained by the nominal inverse kinematics function and a pose corresponding to this estimated solution of the calibrated model can be evaluated:

\[
q_1 = b(T_s^1), \quad T_2 = f(c_1, q_1).
\]

(4.3.9)

Since the difference between \( T_2 \) and \( T_s^1 \) is caused by the differential change \( \Delta c \), hence

\[
T_2 - T_s^1 = f(c_1, q_1) - f(c_o, q_1) = \left. \frac{\partial f(c, q)}{\partial c} \right|_{c=c_o} \Delta c.
\]

(4.3.10)

Noting that the vector \( q_1 \) should be reasonably close to \( q_d \), then the value of the last item in (4.3.7) can approximately be treated as:

\[
\frac{\partial}{\partial q} \left( \left. \frac{\partial f(c, q)}{\partial c} \right|_{c=c_o} \Delta c \right) \bigg|_{q=q_o} \Delta q = \left. \frac{\partial f(c, q)}{\partial c} \right|_{c=c_o} \Delta c - \left. \frac{\partial f(c, q_o)}{\partial c} \right|_{c=c_o} \Delta c.
\]
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

Thus a better approximation of the $T_s$ is obtained by using this approximation value in (4.3.7), which, in conjunction with (4.3.6) and (4.3.10), gives

$$T_s = T_s^2 = T_s^1 - T_2 - T_d = 3T_d - T_1 - T_2. \quad (4.3.11)$$

The above approximation of the $T_s$ can be further iteratively carried out as

$$T_s = T_s^n = T_s^{n-1} - (T_n - T_d) = (n+1)T_d - \sum_{i=1}^{n} T_i$$

$$q_n = b(T_s^n)$$

$$T_{n+1} = f(c_1, q_n) \quad (4.3.12)$$

until some criterion of pose error $(T_{n+1} - T_d)$ is satisfied and the vector $q_n$ is regarded as the intended inverse kinematic solution. In general, the number of iterations will be small unless the calibrated model is significantly different from the nominal configuration. In fact, for the calibrated Puma robot discussed in this chapter, $q_2$, or even $q_1$, is accurate enough for most pose definitions.

4.3.3. Model conversion

The numerical method derived in the previous section is not directly applicable to the S-model calibrated Puma robot under discussion. The main difficulty is due to the near-parallel structure of joint two and joint three, which leads to some link parameters changing dramatically from their nominal values. Therefore, a model description change must be undertaken to make the derived method viable.

In order to avoid the use of the common norm, the homogeneous transform matrix $A_2$, which relates the co-ordinate system $o_2x_2y_2z_2$ assigned
on link 2 to the $o_1x_1y_1z_1$ assigned on the previous link (see Figure 4.3.1), should have a form other than the DH model convention. Here $A_2$ is chosen as

$$A_2 = R(Z,q_2)Tr(Z,d_2)Tr(X,a_2)R(Y,\beta)R(X,\alpha), \quad (4.3.13)$$

where $a_2$ and $d_2$ are defined as shown in Figure 1; $q_2$ is the joint variable measured between the axis $x_1$ and the line denoted by $a_2$ in a plane perpendicular to the axis of joint 2; $\beta$ and $\alpha_2$ are the two rotation angles needed to align the $z_2$ with the axis of joint 3. The introduction of an extra rotation, $R(Y,\beta)$, enables the two co-ordinate frames to be related without referring to the common norm $a_2$. Note that, when $\alpha_2 = 0$ and $\beta = 0$, which corresponds to the case that the two axes are parallel, equation (4.3.13) is equivalent to the nominal DH definition.

Figure 4.3.1. Illustration of $a_2, d_2, q_2$

The parameters with a "-" cap shown in Figure 4.3.1 are the relevant calibrated DH model parameters drawn from Table 4.3.2 and the frame $o_nx_ny_nz_n$ is the extracted DH convention counterpart of the frame $o_2x_2y_2z_2$. The twist angle $\alpha_2$ is the rotation angle around the axis $x_n$ and is measured
from the axis $z_n$ to the dashed line parallel to the axis $z_1$ in a plane perpendicular to $x_n$. Having chosen the model form for the homogeneous transformation matrix $A_2$, the remaining task is to derive the parameters required by equation (4.3.13).

For the parameter set in Table 4.3.2, the conversion from the S-model (or extracted DH model) to equation (4.3.13) for link 2 is not unique. This can be clearly seen in Figure 4.3.1 since the change of $d_2$ will cause the link 2 co-ordinate frame to slide along the joint 3 axis. The extra constraint chosen is to select the value of $d_2$ so that $d_3$ will be its nominal value, 0. This is satisfied by setting

$$d_2 = d_2^r - |d_3^r| \cos \alpha_2.$$  \hspace{1cm} (4.3.14)

![Figure 4.3.2. Geometric relationship between frames](image)

In Figure 4.3.2, an auxiliary variable $\theta$ is illustrated. It is defined as the rotation angle around the $z_1$ axis and is measured from the line $p o_2$ to the line $p p'$. Since the dashed line $o_2 p'$ is parallel to the axis of joint 2 and passes the intersection between the axis of joint 3 and the common norm $\vec{a}_2$, it must be orthogonal to the common norm and coplanar with the axis.
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

of joint 3. Therefore the intersection point between the dashed line and the plane determined by the line \( p_0z \) rotating around the axis of joint 2 can be used to establish a clear triangular relationship between the DH convention parameters and the parameters required by equation (4.3.13). From figure 4.3.2, the following equations are obtained

\[
\begin{align*}
 r &= d_z^2 \\
 \theta &= - \arctan \left( \frac{d_z^2 \sin \alpha_2}{r} \right) \\
 a_2 &= \frac{r}{\cos \theta}
\end{align*}
\]  

(4.3.15)

where the minus sign in equation (4.3.15) accounts for the fact that a negative twist angle \( \alpha_2 \) generates a positive angle \( \theta \) and vice versa.

Having obtained the parameters \( d_2, a_2 \) and the auxiliary variable \( \theta \), the remaining unresolved parameters are \( \alpha_2 \) and \( \beta \). Consider the two coordinate frames \( O_x'x'y'z' \) and \( O_x'y'z' \) in Figure 4.3.2, where the \( O_x'y'z' \) frame is related to the frame \( O_x'y'z' \) by \( R(Z, q_2)T(Z, d_2)T(X, a_2) \). The effect of the transformation \( R(Y, \beta)R(X, \alpha_2) \) in equation (13) is to align the \( O_x'y'z' \) frame with the frame of \( O_x'y'z' \). Hence the vector \( u=[0, 0, 1]^T \) in the frame \( O_x'y'z' \) can be described in the frame \( O_x'y'z' \) as

\[
R(Y, \beta)R(X, \alpha_2) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sin \beta \cos \alpha_2 \\ -\sin \alpha_2 \\ \cos \beta \cos \alpha_2 \end{bmatrix}.
\]  

(4.3.16)

On the other hand, from the geometric relationship shown in Figure 4.3.2, the projections of the same unit vector on the axes of the frame \( O_x'y'z' \) can be determined. Denoting the projections as \( [v_x, v_y, v_z]^T \), the value of \( v_z \) is straight-forward as the axis \( z' \) is parallel to the dashed line \( p'o_n \) in Figure 4.3.2:

\[
v_z = \cos \alpha_2.
\]  

(4.3.17a)
Since the axis $x'$ is in alignment with the line $po'$, axis $y'$ is vertical to the line $po'$ and the $x'y'$ plane is coplanar with the triangle $pop'$, the magnitude of the projections $v_x$ and $v_y$ can be easily calculated following the triangular relationships shown in Figure 4.3.2. A slightly complicated matter is the determination of the signs of the two projection values.

The projection $v_x$ is always a non-positive value no matter what sign the twist angle $\alpha_2$ is. This is because the length of $a_2$ is always not less than $r$ under the condition that $d_2 < d_2'$, thus the axis $z_2$ always tilts to the negative direction of $x'$. Combining the fact that the value of $\theta$ has a different sign with respect to that of $\alpha_2$ and the magnitude of both is less than 90 degrees, the value of $v_x$ is obtained as:

$$v_x = -|\sin\theta\sin\alpha_2| = \sin\theta\sin\alpha_2.$$  \hspace{1cm} (4.3.17b)

The sign of $v_y$ depends on the sign of the twist angle $\alpha_2$: positive when $\alpha_2 < 0$ and negative when $\alpha_2 > 0$. Therefore,

$$v_y = -\text{sign}(\alpha_2) |\cos\theta\sin\alpha_2| = -\cos\theta\sin\alpha_2$$  \hspace{1cm} (4.3.17c)

where \text{sign}(x) returns the sign of its argument $x$. Relating equations (4.3.16) and (4.3.17) leads to

$$\begin{bmatrix} \sin\beta\cos\alpha_2 \\ -\sin\alpha_2 \\ \cos\beta\cos\alpha_2 \end{bmatrix} = \begin{bmatrix} \sin\theta\sin\alpha_2 \\ -\cos\theta\sin\alpha_2 \\ \cos\alpha_2 \end{bmatrix}.$$  \hspace{1cm} (4.3.18)

Thus $\beta$ and $\alpha_2$ can be obtained as

$$\begin{aligned} \beta &= \text{atan}(\sin\theta \tan\alpha_2) \\ \alpha_2 &= \text{atan}(\cos\theta \tan\alpha_2 / \cos\beta) \end{aligned}.$$  \hspace{1cm} (4.3.19)

For the parameter set given in table 4.3.2, by using equations (4.3.14), (4.3.15) and (4.3.19) to convert the parameters of link 2, the newly derived calibrated parameter set is obtained as shown in Table 4.3.3.
Chapter 4: Inverse Kinematics Solution to a Calibrated Puma 560 Robot

Table 4.3.3 The derived link parameters

<table>
<thead>
<tr>
<th>Link i</th>
<th>θ_i (deg)</th>
<th>a_i (mm)</th>
<th>d_i (mm)</th>
<th>β (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-90.09</td>
<td>0.22</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>-0.15</td>
<td>427.2</td>
<td>149.4</td>
<td>-0.14</td>
</tr>
<tr>
<td>3</td>
<td>90.21</td>
<td>-21.5</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>-90.28</td>
<td>0.4</td>
<td>433.1</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>90.23</td>
<td>-0.2</td>
<td>-0.1</td>
<td>N/A</td>
</tr>
<tr>
<td>6*</td>
<td>0.0</td>
<td>0.0</td>
<td>55.9</td>
<td>N/A</td>
</tr>
</tbody>
</table>

It is worth pointing out that a small variation of some nominal Puma kinematic model parameters (those $a_i$ and $d_i$ with non-zero values) will not affect the validity of a closed-form inverse kinematic solution. Table 4.3.4 has reflected this consideration where a nominally configured Puma robot model has had several parameters set to the calibrated values given in Table 4.3.3. The remaining ten unequal parameters in Table 4.3.3 and Table 4.3.4 form a parameter vector which relates the two models with a small change in parameter vector. It should be noted that the model pair described by Table 4.3.3 and Table 4.3.4 have the same joint variable zero point definition and this feature does not change when the value of the parameter vector changes. Therefore, the closed-form inverse kinematic solutions based on the parameter set in Table 4.3.4 can be used to iteratively approach the inverse kinematic solutions of the calibrated model described in Table 4.3.3 by employing the algorithm established in previous section.

It should be noted that, by the adoption of equation (4.3.13), the definition of the joint angle of joint 2, $q_2$, is different from its counterpart $\tilde{q}_2$ of the extracted DH model given in table 4.3.2 (see figure 4.3.2). The same
is also true of joint 3 as the origin of frame 2 slides from $o_1$ to $o_2$ along the axis of joint 3. The corresponding relationship is

$$q_2 = q'_2 + \theta, \quad \text{and} \quad q_3 = q'_3 + \theta$$

(4.3.20)

where $\theta$ is calculated by (4.3.15).

**Table 4.3.4 'Nominal' link parameters**

<table>
<thead>
<tr>
<th>Link</th>
<th>$a_1$ (deg)</th>
<th>$a_i$ (mm)</th>
<th>$d_i$ (mm)</th>
<th>$\beta$ (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-90.00</td>
<td>0.0</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>427.2</td>
<td>149.4</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>90.00</td>
<td>-21.5</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>-90.00</td>
<td>0.0</td>
<td>433.1</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>90.00</td>
<td>0.0</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>6*</td>
<td>0.0</td>
<td>0.0</td>
<td>55.9</td>
<td>N/A</td>
</tr>
</tbody>
</table>

4.3.4 Numerical examples

Three examples of the numerical calculation of the inverse kinematics solution to the calibrated Puma 560 robot are listed in this section to illustrate the effectiveness of the derived algorithm. The Angle line of the tables in this section is the exact inverse kinematics solution for reference purpose. It has been used to generate the desired pose through the extracted DH model given in Table 4.3.2. The Sol. 1 line is the solution based on the $T_1$ approximation, and the Sol. 2 line the $T_2$ approximation (see equations (4.3.8)-(4.3.12)). They correspond to the result of the first iteration and the second iteration of equation (4.3.12).

The numerical examples in this section have shown that the derived inverse kinematics algorithm is simple and effective. For many given pose definitions, the first iteration result of the corresponding inverse
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

The computation time needed to perform the first iteration is roughly three times that required for the nominal inverse kinematics calculation. For the second iteration approximation, the time factor becomes roughly five. Hence the derived numerical approach for solving the inverse kinematics problem of a calibrated industrial robot is highly suitable for real-time applications. In fact an on-line implementation of the calibrated kinematic model in controlling the Puma 560 manipulator has revealed an average calculation time of less than 1.8 ms for the inverse kinematics calculation. The implementation employs one T805 transputer in the open architecture robot controller described in this work to undertake the calculation and the termination condition is set as (reference (4.3.12))

\[
\text{Trace}\left(\begin{pmatrix} T_{n+1} - T_d \end{pmatrix}^T \begin{pmatrix} T_{n+1} - T_d \end{pmatrix}\right) < 10^{-10}, \tag{4.3.21}
\]

or the iteration number \(n\) reaches 4, whichever comes first. It has been noted that the iteration number rarely exceeds 2 unless the given pose \(T_d\) is very close to the singular configuration case.

### Table 4.3.5 Numerical example 1

<table>
<thead>
<tr>
<th>Angle</th>
<th>J1 (deg)</th>
<th>J2 (deg)</th>
<th>J3 (deg)</th>
<th>J4 (deg)</th>
<th>J5 (deg)</th>
<th>J6 (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sol. 1</td>
<td>10.0000</td>
<td>19.9999</td>
<td>30.0002</td>
<td>40.0001</td>
<td>49.9999</td>
<td>59.9999</td>
</tr>
<tr>
<td>Sol. 2</td>
<td>10.0000</td>
<td>20.0000</td>
<td>30.0000</td>
<td>40.0000</td>
<td>50.0000</td>
<td>60.0000</td>
</tr>
</tbody>
</table>

### Table 4.3.6 Numerical example 2

<table>
<thead>
<tr>
<th>Angle</th>
<th>Joint 1</th>
<th>Joint 2</th>
<th>Joint 3</th>
<th>Joint 4</th>
<th>Joint 5</th>
<th>Joint 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sol. 1</td>
<td>-6.927</td>
<td>0.0</td>
<td>81.579</td>
<td>9.481</td>
<td>7.668</td>
<td>5.55</td>
</tr>
<tr>
<td>Sol. 2</td>
<td>-6.9270</td>
<td>-0.0010</td>
<td>81.5808</td>
<td>9.4822</td>
<td>7.6674</td>
<td>5.5489</td>
</tr>
</tbody>
</table>

PhD. 1994 Nongji Chen
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

Table 4.3.7 Numerical example 3

<table>
<thead>
<tr>
<th></th>
<th>Joint 1</th>
<th>Joint 2</th>
<th>Joint 3</th>
<th>Joint 4</th>
<th>Joint 5</th>
<th>Joint 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>100.0</td>
<td>-14.0</td>
<td>74.883</td>
<td>35.0</td>
<td>-72.0</td>
<td>156.0</td>
</tr>
<tr>
<td>Sol. 1</td>
<td>100.0000</td>
<td>-14.0001</td>
<td>74.8833</td>
<td>34.9997</td>
<td>-72.0001</td>
<td>156.0000</td>
</tr>
<tr>
<td>Sol. 2</td>
<td>100.0000</td>
<td>-14.0000</td>
<td>74.8830</td>
<td>35.0000</td>
<td>-72.0000</td>
<td>156.0000</td>
</tr>
</tbody>
</table>

4.4 Summary

Solving the inverse kinematics problem is essential to controlling a manipulator. It is common that the tasks for a manipulator to perform are frequently specified in Cartesian space since this is the way a human operator can most easily perceive the operations and is also very often required by the tasks themselves. To actually drive the manipulator to the specified Cartesian position, however, requires the coordinated joint motion in joint space. Inverse kinematics calculation is often undertaken at a very high execution rate in a robot controller to attain the required Cartesian path.

In general, current industrial robots have a simple nominal structure that enables the inverse kinematics problem to be solved through an analytical formula. This simplicity, however, is lost in dealing with a calibrated kinematic model of a nominally simple structured robot. This results in the inverse kinematics problem becoming difficult to solve.

A simple and effective method of calculating the inverse kinematics solution for calibrated industrial robots has been described in this chapter. The method exploits the nominal simple structure of an industrial manipulator and uses differential changes in the model parameter vector to iteratively approach a shifted pose which leads to an acceptable inverse
Chapter 4: Inverse Kinematics Solution to A Calibrated Puma 560 Robot

The kinematics solution to the calibrated robot model. It is particularly suitable for being implemented in on-line operations.

Model conversion techniques have been established to describe a pair of near parallel links in a formulation that can avoid the problem of discontinuity in the model description experienced in DH conversion when one of the axes is perturbed away from the parallel case. This enables the derived inverse kinematics method to be used for an S-model calibrated Puma robot that forms part of the research platform for this work.
CHAPTER 5
ROBUST ADAPTIVE TRACKING CONTROL OF ROBOT MANIPULATORS

5.1 Introduction

To cause an actual change in a robot's position or orientation requires the robot's joint actuators to drive their corresponding joints and produce the necessary joint position displacements obligatory to the demanded change. Very few robots nowadays use stepper motors or other actuators that can be controlled in an open loop fashion. Usually, manipulators are powered by actuators which output a torque or a force at each joint. Under such circumstances, some kind of closed-loop control scheme is required to produce the appropriate actuator torque/force that will realise the desired robot motion. Figure 5.1.1 presents a block diagram illustrating closed-loop control of a robot.

Figure 5.1.1 Closed-loop robot control layout.

PhD. 1994 Nongji Chen
Chapter 5: Robust Adaptive Tracking Control of Robot Manipulators

The problem of computing the appropriate actuator commands for a demanded motion is called the robot motion control problem. Numerous techniques have been proposed for its solution. The most commonly seen control algorithm employed by current industrial robot controllers is a proportional and differential (PD) feedback control scheme together with, in many cases, an integral term (PID) to combat the gravitational force. The reason for the popularity of the PD feed-back control scheme is that it is simple and computationally inexpensive. Additionally, when a rigid manipulator dynamic model does not have gravity effects (or their effects have been fully compensated for), a properly chosen PD controller can give asymptotic stable control of the manipulator over the entire work space (Tomel, 1991). However, it is widely recognised that, although the PID form control schemes in general provide a very economic working algorithm for solving the robot motion control problem, they are unlikely to provide a satisfactory dynamic performance at high speeds for multi-degrees-of-freedom robot manipulators. More advanced control algorithms are required in an attempt to achieve better robot dynamic responses.

In this chapter, a robust adaptive tracking control algorithm is derived. The adaptive controller combines a PD feedback control scheme with an adaptive compensation part. The controller employs a leaking parameter, or the so called σ-modification (Ioannou and Kokotovic, 1983; Spong and Ghorbel, 1990), proportional to the norm of the joint speed in its adaptation integrals. It is similar to a certain degree to the adaptive controller developed by Yuan & Stepanenko (1992) as both the schemes adopted the same leaking (σ-modification) strategy. The improvement of the proposed new control approach with regard to the one developed by Yuan & Stepanenko is that the former has included adaptive feed-forward compensation in its control formula, enabling better performance to be achieved. The new adaptive control algorithm is computationally
inexpensive compared to other complicated adaptive control schemes. Its stability analysis is established through a Lyapunov-type function and the result indicates that there exists an error bound which can be adjusted by proper controller parameter design. In this chapter, the performance of the derived algorithm is illustrated by a simulation study on controlling a two-link robot model and comparisons are made against the controller developed by Yuan & Stepanenko (1992).

5.2 Preliminaries

The robot dynamic model considered in this chapter is a rigid manipulator with n degrees-of-freedom described by

\[ M(q)\dot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau - \tau_n, \quad (5.2.1) \]

where the i, j-th element of the matrix \( C(q, \dot{q}) \) is defined as

\[ c_{ij} = \sum_{k=1}^{n} \frac{1}{2} \left( \frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{ik}}{\partial q_j} \right) q_k, \quad (5.2.2) \]

with \( M(q) \in \mathbb{R}^{n \times n}, \dot{q} \in \mathbb{R}^n, C(q, \dot{q}) \in \mathbb{R}^{n \times n}, q \in \mathbb{R}^n, G(q) \in \mathbb{R}^n, \dot{q} \in \mathbb{R}^n, \tau \in \mathbb{R}^n \) and \( \tau_n \in \mathbb{R}^n; m_{ij} \) is the i, j-th element of \( M(q) \). By convention, the \( M(q) \) matrix is called the generalised inertia matrix, \( C(q, \dot{q}) \dot{q} \) is the centrifugal and Coriolis torque vector, \( G(q) \) is the gravity vector, \( \tau \) is the torque applied to the robot joint actuators, and \( \tau_n \) the internal/external disturbances. In the following discussion, \( \| \tau_n \| \) is assumed uniformly bounded.

The matrices \( M(q), C(q, \dot{q}) \) and \( G(q) \) have some interesting properties (Song et al., 1992; Spong & Vidyasagar, 1989):

1) \( M(q) \) is always symmetric and positive definite, i.e.
\[ M(q) = M(q)^T > 0; \quad (5.2.3) \]

2) The matrix \( N(q, \dot{q}) = M(q) - 2C(q, \dot{q}) \) is skew symmetric, and
Chapter 5: Robust Adaptive Tracking Control of Robot Manipulators

\[ x^TN(q,q)x = 0 \quad \forall x \in \mathbb{R}^n \]  
(5.2.4)

when \( C_{ij} \) is defined as in equation (5.1.2);

3) \( M(q) \) and \( G(q) \) are uniformly bounded;

4) \( C(q,q) \) is uniformly bounded in \( q \) and linear in \( q \).

In addition, the following proposition exists.

Proposition: Let each entry of \( C(q,q) \) be defined as (5.2.2), \( e \in \mathbb{R}^n \) is an arbitrary vector. Then the product \( C(q,q)e \) can be expressed as

\[ C(q,q)e = D(q)[q:e] \]  
(5.2.5)

where \( D(q) \in \mathbb{R}^{nxn^2}, [q:e] = [q_1 e_1 \ q_2 e_1 \ ... \ q_n e_n]^T, [q:e] \in \mathbb{R}^{n^2} \).

Proof: Let \( W = C(q,q)e \). The \( i \)-th row of \( W \) Obviously can be written as

\[ W_i = \sum_{j=1}^{n} c_{ij}e_j = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{1}{2} \left( \frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \dot{q}_k e_j \]

\[ = \sum_{j=1}^{n} \frac{1}{2} \left[ \omega_{1j} \omega_{2j} \ ... \ \omega_{nj} \right] [\dot{q}_1 e_j \ \dot{q}_2 e_j \ ... \ \dot{q}_n e_j]^T \]

\[ = \frac{1}{2} \left[ \omega_{11} \omega_{21} \ ... \ \omega_{nn} \right] [\dot{q}_1 e_1 \ \dot{q}_2 e_1 \ ... \ \dot{q}_n e_n]^T \]

\[ = D_i(q)[q:e] \]  
(5.2.6)

where \( D_i(q) \) is the \( i \)-th row of \( D(q) \) and

\[ \omega_{kj} = \left( \frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \]  
(5.2.7)

Substitutes (5.2.6) into each row of \( D(q) \) the proof is completed. \[ \square \]

From the definition of \( D(q) \), it is easy to observe that \( D(q) \) is uniformly bounded. In discussions afterwards, the matrices \( M(q) \), \( D(q) \) and \( G(q) \) are treated unknown. Only some general structural knowledge about them is assumed, such as that they are uniformly bounded. Since the entries in
M(q), D(q) and G(q) are all smooth functions of the joint coordinates vector q, it can be shown that \( i = 1, 2, ..., n \)

\[
\left\| \frac{\partial M}{\partial q_i} \right\| < \infty, \left\| \frac{\partial D}{\partial q_i} \right\| < \infty, \left\| \frac{\partial G}{\partial q_i} \right\| < \infty \tag{5.2.8}
\]

The matrix norm is defined as (Frobenius norm)

\[
\| A \| = \sqrt{\text{Tr}(A^T A)} \tag{5.2.9}
\]

Therefore, by using differential chain, there are

\[
\| M \| \leq \xi_1 \| q \|, \quad \| D \| \leq \xi_2 \| q \|, \quad \| G \| \leq \xi_3 \| q \| \tag{5.2.10}
\]

where \( \xi_1, \xi_2 \) and \( \xi_3 \) are three positive constants with their values being dependent on the particular manipulator under considerations.

### 5.3 A robust adaptive control algorithm

For path tracking control, the planned trajectory information can be employed in a set of control torque calculation formulae to generate the required control torque so that a robot's motion follows the pre-planned trajectory with a satisfactory dynamic performance. Theoretically, if an accurate dynamic model of the robot under question is available, the synthesis of the control torque is straightforward and the dynamic control performance can be optimised through a feed-forward dynamic compensation scheme by utilising the known dynamic model and the pre-planned trajectory information. Practically, however, this may prove to be hard to achieve as: i) establishing an accurate dynamic model of a robot is difficult, costly and time consuming, and frequently only a degree of approximation is feasible; ii) the inevitable frictional effect at each joint is hard to estimate; iii) mismatches between the dynamic model employed in the synthesis of the control torque and the underlying dynamic behaviour of the robot physical system are potential sources for leading to control
instability if they are not carefully accounted for in an overall control strategy; iv) the computational requirement is very high for real-time implementations. Nevertheless, the idea of incorporating the planned trajectory information into the control torque synthesis is attractive as it promises the potential of achieving better tracking performances.

Trajectory information for dynamic control is in general specified in the form of functions of joint accelerations, joint speeds and joint positions with respect to time. Normally, the desired acceleration $\dot{q}_d \in \mathbb{R}^n$, desired speed $\dot{q}_d \in \mathbb{R}^n$ and desired position $q_d \in \mathbb{R}^n$ of a robot motion are calculated at the trajectory planning stage. They are commonly devised as continuous functions of time and their values are limited within a prescribed range to avoid jerking and roughness. In the following discussions, the desired joint accelerations, joint speeds and joint positions are assumed to be uniformly bounded.

5.3.1 The robust adaptive control algorithm

In this section, a robust adaptive control algorithm is synthesised. In order to provide convenience for presenting the control algorithm, the definitions of an error signal and some auxiliary signal vectors are firstly given. The error vector of the joint coordinates is defined as

$$e = q_d - q \quad (5.3.1)$$

Its first and second derivatives against time are derived as

$$\dot{e} = \dot{q}_d - \dot{q} \quad , \quad (5.3.2)$$

$$\ddot{e} = \ddot{q}_d - \ddot{q} \quad (5.3.3)$$

where $q, \dot{q}, \ddot{q}$ are the joint position, speed and acceleration of the robot with their corresponding desired value marked with a subscript $d$ (stands
for *(d)esired*). On the basis of the error signal definition, two auxiliary signals are defined as

\[ \dot{y} = q_d + \alpha \dot{e} \quad (5.3.4a) \]

\[ \ddot{y} = \ddot{q}_d + \alpha e \quad (5.3.4b) \]

with \( \alpha > 0 \). Since the auxiliary signals are the desired trajectory speed and acceleration plus an error or an error speed modification item, they can be referred to as the modified desired joint speed and modified desired joint acceleration respectively.

The adaptive control algorithm formulation is shown as

\[ \tau = K_v \dot{e} + K_p e + \bar{M} \dot{y} + \bar{D}[qy] + \bar{G} \quad (5.3.5) \]

where and \( K_p, R^{n \times n} \) and \( K_v, R^{n \times n} \) are two symmetric positive definite matrices; \( \bar{M}, R^{n \times n}; \bar{D}, R^{n \times n^2}; \) and \( \bar{G}, R^n \) are three adaptive dynamic compensation gain matrices; \([q y] \) is defined as \([q_{e1} q_{e2} \ldots q_{en}] \) \( R^{n^2} \). It should be noted that the item \( K_v \dot{e} + K_p e \) in (5.3.5) forms a constant PD feedback controller. Additionally, the control formula (5.3.5) contains both feed-forward compensation and feedback compensation since \( \dot{y} \) and \( y \) are combinations of trajectory information and tracking error signals. The adaptation equations for \( \bar{M}, \bar{D} \) and \( \bar{G} \) are given later in (5.3.34), (5.3.35), (5.3.36).

Applying (5.3.5) to the robot dynamic model (5.2.1) leads to an error dynamic equation

\[ -M \ddot{e} = (K_v + \alpha C) e + (K_p + C + \alpha M) \dot{e} + (\bar{M} - M) \ddot{y} + \bar{D}[qy] - C \dot{y} + \]

\[ + \bar{G} - G - \tau_n \quad (5.3.6a) \]
where the arguments of $M$, $C$ and $G$ have been dropped for notation simplicity. Using (5.2.5), the error dynamic equation can be rewritten as

$$-M{\ddot{e}} = (K_p + \alpha C)e + (K_v + C + \alpha M){\dot{e}} + \tilde{M}\tilde{y} + \tilde{D}[\dot{q}\ddot{q}] + \tilde{G}, \quad (5.3.6b)$$

where

$$\tilde{M} = M - M \quad (5.3.7)$$

$$\tilde{D} = D - D \quad (5.3.8)$$

$$\tilde{G} = G - G - \tau_n \quad (5.3.9)$$

Since $M$ is symmetric and positive, its inverse always exists. Thus by introducing a state vector $\mathbf{x} = [e^T, \dot{e}^T]^T$, equation (3.6b) can be expressed in the state space as

$$\dot{\mathbf{x}} = A\mathbf{x} + B\Phi\nu \quad (5.3.10)$$

where

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}(K_p + \alpha C) & -M^{-1}(K_v + C + \alpha M) \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -M^{-1} \end{bmatrix}$$

$$\Phi = [\tilde{M}, \tilde{D}, \tilde{G}] = [\tilde{M}, \tilde{D}, \tilde{G}] - [M, D, G + \tau_n] = \Phi_c - \Phi^*,$$

$$\nu = [\tilde{y}, [\dot{q}\ddot{q}]^T, [1]^T]^T.$$

The matrix $\Phi^*$ shown above is uniformly bounded since all its components (i.e. $M$, $D$ and $G + \tau_n$) are uniformly bounded. It is the 'target' that the adaptive gain matrices are aiming at to compensate for.

With regard to the $A$ matrix shown in the error dynamic system (5.3.10), the following proposition exists.
Proposition: Let $k_p$ and $k_v$ be the smallest eigenvalues of symmetric positive definite matrices $K_p$ and $K_v$ respectively, and let $m$ be the largest eigenvalue of the inertia matrix $M(q)$, i.e.

$$m = \sup_{\|q\| = 1} \sup \frac{x^TM(q)x}{x^Tx} > 0.$$ 

If the matrices $K_p$ and $K_v$ are designed with $k_p$ and $k_v$ satisfying

$$k_p + \alpha k_v > (1 - \alpha)m,$$  \hspace{1cm} (5.3.11)

then the symmetric matrix $P$ defined below is positive definite

$$P = \frac{1}{2} \begin{bmatrix} K_p + \alpha K_v + \alpha^2 M & \alpha M \\ \alpha M & M \end{bmatrix}.$$  \hspace{1cm} (5.3.12)

Furthermore, the matrix $A^TP + PA + \dot{P}$ can be expressed as

$$A^TP + PA + \dot{P} = -Q + S,$$

where $Q = Q^T > 0$ is a symmetric positive definite matrix with the smallest eigenvalue being

$$\lambda_1 = \min(k_v, \alpha k_p),$$  \hspace{1cm} (5.3.13)

and $S$ satisfies

$$x^TSx = 0 \hspace{1cm} \forall x \in \mathbb{R}^n.$$  \hspace{1cm} (5.3.14)

Proof: Since $M(q)$ is positive definite, there always exist some full rank transform matrix $T_1$ and its inverse $T_2$, leading to

$$\Delta = T_1MT_2, \hspace{1cm} T_1T_2 = I,$$

where $\Delta = \text{diag}[\eta_1, \eta_2, \ldots, \eta_n]$, and $\eta_1, \ldots, \eta_n$ are the $n$ eigenvalues of the inertia matrix $M(q)$ with $0 < \eta_1 \leq m$. Manipulating $P$ through a similarity transformation, one obtains

$$H = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ T_2 & 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} T_1(K_p + \alpha K_v)T_2 + \alpha^2 \Delta & \alpha \Delta \\ \alpha \Delta & \Delta \end{bmatrix}.$$  \hspace{1cm} (5.3.15)

PhD. 1994 Nongji Chen 130
Matrix $H$ has the same eigenvalues as those of $P$ since the two are similar matrices. Therefore to prove $P$ is positive definite is equivalent to proving that $H$ is positive definite. Re-organising matrix $H$, one obtains

\[
H = \frac{1}{2} \begin{bmatrix}
T_1(K_p + \alpha K_v)T_2 - (k_p + \alpha k_v)I + (k_p + \alpha k_v)I + \alpha^2 \Delta \\
\alpha \Delta \\
\Delta
\end{bmatrix}
\]

\[
= H_1 + H_2,
\]

where

\[
H_1 = \frac{1}{2} \begin{bmatrix}
T_1(K_p + \alpha K_v)T_2 - (k_p + \alpha k_v)I & 0 \\
0 & 0
\end{bmatrix},
\]

\[
H_2 = \frac{1}{2} \begin{bmatrix}
(k_p + \alpha k_v)I + \alpha^2 \Delta & \alpha \Delta \\
\alpha \Delta & \Delta
\end{bmatrix}.
\]

Obviously, matrix $H_1$ in (5.3.16) is semi-positive definite since $K_p$ and $K_v$ are positive definite matrices with $k_p$ and $k_v$ being their smallest eigenvalues respectively. Considering matrix $H_2$ in (5.3.16), according to Gershgorin theorem (Bell, 1975), any eigenvalue $\lambda$ of $H_2$ satisfies the union of the inequalities:

\[
\left| \lambda - \frac{1}{2} (k_p + \alpha k_v + \alpha^2 \eta_i) \right| \leq \frac{1}{2} \alpha \eta_i, \quad i = 1, ..., n,
\]

\[
\left| \lambda - \frac{1}{2} \eta_i \right| \leq \frac{1}{2} \alpha \eta_i, \quad i = 1, ..., n.
\]

From (5.3.17a), for $i = 1, ..., n$, there is either

\[
0 < \lambda - \frac{1}{2} (k_p + \alpha k_v + \alpha^2 \eta_i) \leq \frac{1}{2} \alpha \eta_i
\]

(5.3.18a)

or

\[
0 \geq \lambda - \frac{1}{2} (k_p + \alpha k_v + \alpha^2 \eta_i) \geq - \frac{1}{2} \alpha \eta_i.
\]

(5.3.18b)
By noting that \( m \geq \eta_1 > 0 \), and using (5.3.11) in (5.3.18b), there must be \( \lambda > 0 \). Since \( \lambda \) is arbitrarily chosen, \( \lambda > 0 \) implies all eigenvalues of \( H_2 \) are positive. Thus \( H_2 > 0 \). Combining \( H_1 \) and \( H_2 \) leads to \( H \) being a positive definite matrix, which, in turn, results in \( P \) being positive definite. Now the matrix \( A^TP+PA+\dot{P} \) can be calculated by simple arithmetic:

\[
A^TP + PA + \dot{P} = \frac{1}{2} \begin{bmatrix}
-2\alpha K_P - \alpha^2 (C^T+C-M) & \alpha (M^T-C^T-C) \\
\alpha (M-C^T-C) & -2K_V + (M-C^T-C)
\end{bmatrix}
\]

where

\[
Q = \begin{bmatrix}
\alpha K_P & 0 \\
0 & K_V
\end{bmatrix},
\]

and

\[
S = \frac{1}{2} \begin{bmatrix}
-\alpha^2 (C^T+C-M) & \alpha (M-C^T-C) \\
\alpha (M-C^T-C) & (M-C^T-C)
\end{bmatrix}.
\]

Using equation (5.2.4), it is obvious that \( S \) satisfies (5.3.14). From the special structure of matrix \( Q \), it is easy to see that the eigenvalue set of \( Q \) is the union of the eigenvalue set of \( \alpha K_P \) and that of \( K_V \). Thus all the eigenvalues of \( Q \) are positive and the smallest eigenvalue \( \lambda_1 \) is

\[
\lambda_1 = \min(k_V, \alpha K_P).
\]

Since both \( \alpha K_P \) and \( K_V \) are symmetric matrices, therefore the matrix \( Q \) is symmetric and positive definite. 

The above proposition provides a sufficient condition for devising a PD feedback controller's gain matrices \( K_P \) and \( K_V \) that lead to a negative definite matrix \( A^TP + PA + \dot{P} \) for the positive definite matrix \( P \) defined in (5.3.12). This feature is used later in the derivation of the adaptation formulae. It should be noted that, in the proposition proof, the smallest
eigenvalue $\lambda_1$ of $Q$ can be made arbitrarily large by properly increasing the feedback gain matrices $K_P$ and $K_V$ as shown by equation (5.3.20).

For the error dynamic system described by (5.3.10), assume the PD feedback control gain matrices have been properly set hence $P$ and $Q$ are positive definite matrices, consider a positive definite function $V(x)$

$$V(x) = x^T P x + \frac{\gamma}{2} \text{Tr}(\Phi^T \Phi) > 0$$

where $\gamma > 0$, $\text{Tr}(\Phi^T \Phi)$ is the trace of matrix $\Phi^T \Phi$, and $\Gamma^T \Gamma > 0$ with $\Gamma$ being a constant positive definite matrix. It follows that

$$\dot{V} = x^T (A^T P + PA + \dot{P}) x + 2x^T P B \dot{v} + \gamma \text{Tr}(\Phi^T \Phi)$$

$$= -x^T Q x + x^T R \dot{v} + \gamma \text{Tr}(\Phi^T \Phi)$$

where

$$R = 2PB = -[\alpha I_n \ I_n]^T$$

$$\dot{\Phi} = \dot{\Phi}_c - \dot{\Phi}_*$$

From (5.3.24), if $\Phi_c$ could be updated in a form so that it leads to

$$\dot{\Phi} = -\frac{1}{\gamma} \Gamma^{-1} R^T x v^T$$

or

$$\dot{\Phi} = -\frac{1}{\gamma} \Gamma^{-1} R^T x v^T - \sigma \| \dot{q} \| \Phi$$

$\dot{V}(x)$ would then be of negative value for any non-zero state vector $x$ and the dynamic system described by (5.3.10) would be asymptotically stable. However, equation (5.3.26) is not reachable since $\Phi^*$ and $\Phi_*$ are unknown.
under the assumption. Consequently an adaptation law for $\Phi_c$ is synthesised by adopting an integration leaking scheme (Ioannou and Kokotovic, 1983) with the leaking parameter being chosen as $\sigma \|q\|$ (Yuan & Stepanenko, 1992) to combat the disturbance caused by $\Phi^*$ in (5.3.26):

$$
\dot{\Phi}_c = \frac{1}{\gamma} T^{-1} R^T x v^T - \sigma \|q\| \Phi_c.
$$

(5.3.27)

Introduce (5.3.27) into (5.3.25b), $\dot{\Phi}$ can be written as

$$
\dot{\Phi} = \frac{1}{\gamma} T^{-1} R^T x v^T - \sigma \|q\| \Phi_c - \Phi^* - \sigma \|q\| \Phi^*
$$

(5.3.28)

$$
= -\frac{1}{\gamma} T^{-1} R^T x v^T - \sigma \|q\| \Phi - \Phi^* - \sigma \|q\| \Phi^*
$$

where $\Omega = \Phi^* + \sigma \|q\| \Phi^*$. Since $M, D, G$ and $\tau_d$ are uniformly bounded, by observing (5.2.10) it can be derived that

$$
\|\Omega\| \leq \xi \|q\|, 
$$

(5.3.29)

where $\xi$ is a positive constant. Substituting (5.3.28) into (5.3.24) leads to

$$
\dot{V} = -x^T Q x + \text{Tr}(\gamma \sigma \|q\| \Phi^T \Phi - \gamma \Omega^T \Phi)
$$

$$
\leq -\lambda_1 \|x\|^2 - \gamma (\sigma \xi_1 \|q\| \|\Phi\|^2 - \xi_2 \xi \|q\| \|\Phi\|)
$$

$$
\leq -\lambda_1 \|x\|^2 - \gamma \|q\| \|\Phi\| (\sigma \xi_1 \|\Phi\| - \xi_2 \xi). 
$$

(5.3.30)

Where $\lambda_1$ is the smallest eigenvalue of $Q$ as given by (5.3.21), $\xi_1$ and $\xi_2$ are the smallest and the largest eigenvalues of $\Gamma$ respectively.

Noting that (5.3.30) is negative if either

$$
\|\Phi\| \geq \rho = \frac{\xi_2 \xi}{\sigma \xi_1},
$$

(5.3.31)

or

$$
\|\Phi\| \leq \rho.
$$
Chapter 5: Robust Adaptive Tracking Control of Robot Manipulators

\[ \| \Phi \| < \rho, \quad \text{and} \quad \| x \|^2 > \frac{\rho \gamma \xi_2 \zeta \| q \|}{\lambda_1} = \rho \| q \|. \quad (5.3.32a) \]

Where \( \nu = \frac{\rho \gamma \xi_2 \zeta}{\lambda_1} \). Since \( \| q \| \leq \| q_d \| + \| x \|, \) (5.3.32a) can be further derived as

\[ \| \Phi \| \leq \rho, \quad \text{and} \quad \| x \| > \frac{1}{2} \left( \nu + \sqrt{\nu^2 + 4 \nu \| q_d \|} \right). \quad (5.3.32b) \]

It is clear that the values \( \rho \) and \( \nu \| q \| \) in (5.3.31) and (5.3.32) specify a rectangle in the two-dimensional space \( (\| x \|, \| \Phi \|) \). Consequently \( V(x) \) is bounded since no matter what initial value \( V(x) \) has, it will be attracted below a constant \( V^* \) that covers the rectangle, and once it has done so, there will always be \( V(x) < V^* \).

The fact that \( V(x) \) is bounded when adaptation law (5.3.27) is adopted implies both \( x \) and \( \Phi \) are bounded as (5.3.22) is a monotonous increase function with regard to \( \| x \| \) and \( \| \Phi \| \). The ultimate asymptotic bounds for \( x \) and \( \Phi \) are derived as not being greater than \( (\nu + \sqrt{\nu^2 + 4 \nu \| q_d \|})/2 \) and \( \rho \) respectively.

Note that \( \nu \) can be reduced by either increases \( \lambda_1 \) or reduces \( \gamma \) as (5.3.22a) suggests. The derived bound for the error vector can therefore be conveniently adjusted. Theoretically, the error bound can be designed as arbitrarily small since we can make \( \lambda_1 \) arbitrarily large according to (5.3.21).

In the above algorithm derivation, all the adaptive gain matrices have been lumped together to simplify the notation in the context. In a more flexible form, the positive definite function \( V(x) \) in (5.3.22) can be replaced with a new function \( V_1(x) \)

\[ V_1(x) = x^T P x + \frac{\gamma_1}{2} \text{Tr}(\bar{M}^T T_1 \bar{M}) + \frac{\gamma_2}{2} \text{Tr}(\bar{D}^T T_2 \bar{D}) + \frac{\gamma_3}{2} \text{Tr}(\bar{G}^T T_3 \bar{G}), \quad (5.3.33) \]
where the matrices $\tilde{M}$, $\tilde{D}$ and $\tilde{G}$ are defined in (5.3.7), (5.3.8) and (5.3.9), and similar resultant error bounds can be established when the corresponding adaptation equations for $\tilde{M}$, $\tilde{D}$ and $\tilde{G}$ are

$$\dot{\tilde{M}} = -\frac{1}{\gamma_1} \Gamma_1 \bar{J} \bar{x} \bar{y} y^T - \sigma \| \dot{q} \| \tilde{M}, \quad (5.3.34)$$

$$\dot{\tilde{D}} = -\frac{1}{\gamma_2} \Gamma_2 \bar{J} \bar{x} [\dot{q} y] y^T - \sigma \| \dot{q} \| \tilde{D}, \quad (5.3.35)$$

$$\dot{\tilde{G}} = -\frac{1}{\gamma_3} \Gamma_3 \bar{J} \bar{x} 1^T \sigma \| \dot{q} \| \tilde{G}. \quad (5.3.36)$$

These equations provide more freedom for choosing adaptive gains for different items, hence making it possible to obtain a finer tuned controller to optimise the tracking performance for a given robot dynamic system. The control torque calculation formula remains unchanged as (5.3.5).

5.3.2 Discussion

The derivation of the error bound in section 5.3.1 relies on a sufficient condition (5.3.11) that relates the PD feedback gain matrices $K_p$ and $K_v$ with the maximum eigenvalue of the inertial matrix $M(q)$. As the dynamic model of the robot is not assumed known, (5.3.11) does not provide quantitative guidance to the algorithm design but an instructive qualitative reference value. In fact, even if a robot model is known, the calculation of the maximum eigenvalue of the inertial matrix $M(q)$ in general is difficult as it is a complex functional problem over the joint space. Thus the significance of (5.3.11) is that it implies the stability of the established adaptive control algorithm though the proper values of $K_p$ and $K_v$ may have to be devised through other means.

Since many industrial robots are controlled by PID control algorithms, the use of the established adaptive control algorithm becomes convenient and advantageous. Firstly, not requiring a dynamic model means the...
established adaptive control algorithm can be employed to replace the existing PID control algorithm with little preparation work. Secondly, the PD parameters of the PID controller may well serve as the starting point of the adaptive controller's parameter tuning process given that equation (5.3.5) is, in essence, a further expansion on the PD formula. It should be noted that the established adaptive control algorithm does not include any trigonometric calculations in its evaluation. The computational requirement for real-time implementations is very moderate. This is attractive to real-time applications as a smaller evaluation time for a control algorithm means a smaller pure time-delay element existing in the feedback control loop.

5.4 Simulation Study

A two-link robot, shown in figure 5.4.1, has been used in the simulation study to evaluate the performance of the adaptive control algorithm established in section 5.3. The parameters of the model are the same as those in Yuan and Stepanenko (1992). The model dynamics equation is characterised as

\[ M(q)q + D(q)[\dot{q}q] + G(q) = \tau, \quad (5.4.1) \]

where

\[
M(q) = \begin{bmatrix}
(2l_1c_2 + l_2)l_2m_2 + l_1^2(m_1 + m_2) & l_1^2m_2 + l_1l_2c_2m_2 \\
l_2^2m_2 + l_1l_2c_2m_2 & l_1^2m_2
\end{bmatrix};
\]

\[
D(q) = \begin{bmatrix}
0 & -l_1l_2s_2m_2 & -l_1l_2s_2m_2 & -l_1l_2s_2m_2 \\
l_1l_2s_2m_2 & 0 & 0 & 0
\end{bmatrix};
\]

\[
G(q) = \begin{bmatrix}
g(m_2l_2c_1 + (m_1 + m_2)l_1c_1) & m_2l_2gc_12 \end{bmatrix}^T.
\]

In the above equations, \( s_1 = \sin(q_1), s_2 = \sin(q_2), c_1 = \cos(q_1), c_2 = \cos(q_2), \) \( c_{12} = \cos(q_1 + q_2). \) Each link of the robot is assumed to have a point mass at its
end; the inertia tensors are assumed to be zero. The parameters of the links are $l_1 = 0.7\text{m}$, $l_2 = 0.5\text{m}$, $m_1 = 9.0\text{kg}$, $m_2 = 1.2\text{kg}$.

![Figure 5.4.1 A two-link planar articulated robot.](image)

In the simulation study, the PD feedback control gain matrices of the established adaptive controller are simply chosen with diagonal forms as:

$$K_p = \begin{bmatrix} k_p & 0 \\ 0 & k_p \end{bmatrix}, \quad K_v = \begin{bmatrix} k_v & 0 \\ 0 & k_v \end{bmatrix}.$$  

The controller’s parameters are set as $k_v = 12 \text{Nm s deg}^{-1}$, $k_p=60 \text{Nm deg}^{-1}$, $\alpha=0.9$, $\sigma=0.001$, $\gamma_1 = \gamma_2 = 0.05$ and $\gamma_2 = 0.0022$. All the adaptive gain matrices used are initialised to zero. The error weighting matrices $\Gamma_1, \Gamma_2$ and $\Gamma_3$ in (3.34), (3.35) and (3.36) are all set to unit matrices. The simulation study has chosen the same $k_p$ and $\sigma$ used by Yuan and Stepanenko(1992) so that comparisons among the two can be made on a more sensible basis.

The simulation study is undertaken with two different frequency trajectory signals. In the first case, the desired trajectories for both the two joints are given as $q_1 = q_2 = 10(1-\cos(\pi t))$ degrees, the same reference trajectory signals as in the simulation study carried out by Yuan and
The tracking control result of joint one by the established adaptive controller is given in figure 5.4.2. Due to the smaller mass, joint two normally has better tracking performances. Hence the emphasis in this section will concentrate only on joint one's results. In figure 5.4.2, the dotted line is the desired trajectory for joint one while the solid line is the proposed adaptive controller's tracking control result. It is hard to see any significant differences between the two curves in figure 5.4.2, as they almost overlap each other for the given plotting scale.

![Reference signal and tracking control result](image)

**Figure 5.4.2 Joint 1 trajectory tracking results of the proposed controller**

The tracking error corresponding to figure 5.4.2 is shown in figure 5.4.3 by the solid line. As a comparison, the control result by the control algorithm developed by Yuan and Stepanenko (1992) is also drawn in the figure by the dotted line. The proposed adaptive controller has a significantly smaller maximum error and shows better damping effect over Yuan's algorithm.
The improved performance of the new adaptive controller is better illustrated by the second case in which the desired trajectories have double the frequency to $q_1 = q_2 = 10(1 - \cos(2\pi t))$ degrees. Figure 5.4.4 gives the simulated results of trajectory tracking errors obtained by using the two

Figure 5.4.3 Tracking error comparison.

Figure 5.4.4 Tracking error comparison when the signal frequency doubled.
Chapter 5: Robust Adaptive Tracking Control of Robot Manipulators

controllers. Again, the solid line represents for the tracking control error of
the proposed new adaptive controller, while the dotted line represents the
controller developed by Yuan and Stepanenko (1992). The effectiveness of
the adaptive control algorithm established in this chapter is very obvious in
figure 5.4.4.

The satisfactory result in figure 5.4.4 achieved by the proposed new
adaptive control algorithm is not hard to explain when one examines the
dynamic control torque signal profile in figure 5.4.5. Both the required
control torque signal and the adaptive control torque signal are shown in
the figure. The former (dotted line in figure 5.4.5) is obtained via the robot
dynamic model equation (5.4.1) by using the given trajectory information,
while the latter (solid line in figure 5.4.5) is generated in the control
simulation by the adaptive algorithm defined in equations (5.3.5) and
(5.3.34)–(5.3.36). There are few significant differences between the two
signals except in the initial stage.

![Torque Profiles](image)

Figure 5.4.5 Illustration of torque profiles.
A further examination of the adaptive control torque signal is illustrated in figure 5.4.6. The contribution made by the PD feedback control element in equation (5.3.5) is shown by the solid line, and the whole controller’s control torque signal is represented by the dotted one. At the initial stage, since all the adaptive gain matrices have been set to zero, the PD feedback controller plays a dominant role. This situation is quickly shifted when the adaptive compensation part becomes more and more effective. In fact, the PD feedback controller's contribution becomes so small, as indicated near the right end of the figure, that the robot could be considered as being driven by the adaptive compensation torque signal generated by the adaptive gain matrices through the planned trajectory information. Obviously the tracking error is satisfactorily reduced. Additionally, it is worth pointing out that the established adaptive controller does not have any excessive control torque in achieving the tracking performance shown in figure 5.4.4.

![Figure 5.4.6 Control torque profile and PD part contributions](image_url)
5.5 Summary

High performance control of a robot manipulator is a difficult task. The difficulty arises from the complex dynamic behaviour of a manipulator system. Although it is widely acknowledged that current industrial practice of using a simple linear PID controller is unlikely to achieve satisfactory dynamic performance, especially when a robot is moving at a relatively high speed, improvements have been slow to be implemented.

From a pragmatic point of view, many robot control algorithms proposed in the literature appear to be too complicated to be implemented in real-time control with reference to current technology and implementation cost. This has promoted the interest in developing stable, high performance robot control algorithms that have an acceptable computational complexity.

In this chapter, a robust adaptive control algorithm for robot tracking control has been derived. The proposed algorithm is stable, relatively simple and easy to use. It bears a very close link to the conventional PD feedback controllers which enables experiences obtained in PID implementations to be exploited in the realisation of the more advanced control algorithm. Simulation studies undertaken in this chapter have demonstrated the effectiveness of the proposed adaptive control algorithm in controlling a simple two-link robot model. A real-time validation of the control algorithm for a Puma 560 robot is given later in Chapter 7.
6.1 Introduction

A fundamental feature of operations performed by robots can be described as generating position/orientation changes of a robot's end-effector. Depending on a particular task, such changes can be specified by supplying point destinations in workcell or continuous curves that the robot end-effector needs to traverse. Although, in principle, it is perfectly possible for a robot controller to work on a large bank of data that directly defines the required robot motion trajectory at its finest resolution, it is common practice that a programming level is provided in a robot controller so that robot motion can be specified in a form more compact and also tractable for a human operator and the trajectory can be secured to be smooth without discontinuities. A typical programming command will specify constraints, such as the target position, velocities, path shape, timing information, etc., on the required motion of a robot operation. The controller then plans a trajectory which in turn produces a stream of setpoints suitable for tracking by a feedback controller, enabling the motion to be realised.

Trajectory planning schemes generally "interpolate" or "approximate" the desired path by a class of polynomial functions. The interpolation and approximation can be conducted either in joint space or in Cartesian space depending on the operational constraints. In general, planning in joint space is relatively easy and straightforward. But the resultant trajectory may have a complex and irregular path in Cartesian space, which is frequently
undesirable unless the travelling path is not important in the intended robot operation. If the motion path in Cartesian space is of primary concern, planning the trajectory in Cartesian space is naturally the first option. The major problem involved in Cartesian space planning is that it is much more complex compared to joint space planning due to the difficulty of handling the orientation change. Additionally, more computational power is required for both the path calculation and the mapping from Cartesian space to joint space at each trajectory updation cycle on real-time execution.

The effort of employing workcell feedback sensing mechanism to equip a robot with the capability of 'sensing and reacting' has led to the requirement for achieving on-line trajectory adaptation. There are two important implications of it on trajectory planning. Firstly, it does not allow the use of off-line trajectory planning techniques which generally can result in the time invariant planned trajectories to be optimal in accordance to some criterion but too difficult to be modified in response to real-time sensory information. Secondly, the generated trajectories by an on-line planning scheme must be in an easily retrievable form so that the on-line execution is rapid to trajectory change requests.

Trajectory planning in its widest sense may include advanced topics of obstacle avoidance, optimum path searching, etc. In many practical cases, it has been narrowly confined into the on-line generation of path segments, leaving the more advanced issues to a high level planning problem which is still open for research in many aspects.

During the last decade, there has been a strong tide of research work in using external sensors to improve the performance of robot systems. Of various external sensors investigated, machine vision is the system that promises to be the most powerful and effective sensing system because it
can recognise and accurately locate a workpiece in 3D. In general, images available to a vision system from a camera are characterised by far more data than can be analysed in detail by existing vision processing systems in real-time. The information that is important for the particular task tends to be episodic (gathered in space and time) and is often surrounded by much information that is redundant. This feature is exploited by active vision systems which aim to improve the performance by focusing attention and sparse computational resources on the critical regions, ignoring the irrelevant data. This helps to ease the computational burden and to achieve the objectives of reliable extraction of information within certain time constraints.

The advantages that active vision offer include the ability to overcome a limited field of view offered by a fixed configuration camera, to increase the spatial resolution of the vision system by being able to examine the full visual field and by reducing the computational burden by selecting portions of the scene containing potentially interesting features. Other advantages include the ability to stabilise the images, aiding motion estimation, figure-ground separation, better range estimates fused from stereo, focusing and sensor geometry, and lessening the effects of occlusions. These advantages come at the price of the requirement of an extra vision system configuration control mechanism which must be able to adjust to an appropriate configuration in response to some event or occurrence in part of the visual field.

Mounting an active vision sensor on the end-effector of an industrial robot has several distinct advantages over fixed camera configurations. The first is that it is possible to utilise the flexibility of the robot in providing the six degrees-of-freedom necessary to accomplish 3D positioning and orientation of the active device. The sensor can also be moved over the
entire workcell avoiding difficulties of obscured views to give a complete view of the workcell. The active vision sensor can also be brought closer to the various parts of the workcell to allow higher resolution images to be obtained.

In this chapter the implemented on-line trajectory planning scheme and its adaptation strategy in the open architecture robot controller is firstly outlined. It is then followed by a description of a stereo-vision system, which was developed by Pretlove (1993) and is employed to achieve robot visual guidance in this research. Afterwards, aspects of achieving static and dynamic robot visual guidance by using a robot vision system mounted on a robot end-effector are discussed. Finally, issues involved in sensor-robot system integration are considered.

6.2 Trajectory planning and On-line alteration strategy

The trajectory planner in the open architecture robot controller undertakes a linear interpolation between two given target positions in default. The planner expects that each motion command specifies a target position and, if other than the default mode is to be applied, the path type and timing information of the intended motion. When the planner receives a motion command, the newly issued target position is combined with the one specified by the last command to define a trajectory's two end points if no trajectory alteration is requested. The target position of a motion command can be specified in either Cartesian or joint coordinates. Both Cartesian space and joint space trajectories can be generated.

6.2.1 Trajectory planning

The trajectory planner produces a trajectory with a basic speed profile shown in figure 6.2.1 in default mode. The profile contains three phases for
Chapter 6: On-Line Trajectory Adaptation and vision guidance

the planned motion: acceleration period, steady speed (cruising) period and deceleration period. It provides a zero speed at both the starting point and the ending point, enabling a rational and smooth motion with continuities in both position and speed to be achieved. The ramping up and down in the speed profile effectively limits the maximum acceleration required.

![Speed-Time Path Duration](image)

Figure 6.2.1 The basic speed ramp profile in trajectory planner

In joint space planning, the speed profile is applied independently to each joint to calculate the corresponding \( \tau \) and \( T \). This is done by using a pair of preset speed and acceleration values corresponding to the joint under calculation. Among all the joints, the maximum acceleration duration and the maximum cruise duration are chosen to form the path time, and each joint’s actual acceleration and speed values are then adjusted in accordance to the determined motion duration. This brings all the joints to the same time pace and provides a co-ordinated motion.

In Cartesian space planning, the displacement from one position to another can be decomposed into translational change and rotational change in Cartesian space. Because rotations about different axes do not comply

PhD. 1994 Nongji Chen
with the rule of commutation, a rotational change in general cannot be formulated to linear consecutive angle changes with regard to the fixed Cartesian axes as a change does in translation. Equally it is impossible for the rotational change to be expressed as rotation matrices in a linear form with a scaling coefficient. To sidestep this difficulty, the planner uses a 4-tuple consisting of an axis of rotation and an angle of rotation about it to describe the orientation shift. Consequently, the translational change and the resulting rotation angle in the 4-tuple representation can be linearly scaled and the speed profile in figure 6.2.1 is applied to both of them in shaping up the trajectory.

The calculation of the acceleration time $\tau$ and the path time $T$ for a trajectory with a ramp speed profile is straightforward. To simplify the illustration, only Cartesian space path is considered here afterwards. The calculation for joint space path is very similar.

In default mode, the velocity and acceleration for Cartesian motions are specified in terms of a translational "cruising" speed magnitude $v_t$, a rotational speed magnitude $v_r$, a translational acceleration magnitude $a_t$, and a rotational acceleration magnitude $a_r$. Assume the required motion is given as a straight line starting from the target position $C_1$ (in homogeneous transformation form) and ending at the target position $C_2$. The displacement $D$ between the two targets can be computed as

$$D = C_2 C_1^{-1}. \quad (6.2.1)$$

$D$ is used to compute $u_t$ (a unit vector parallel to the translational vector), $d_t$ (the magnitude of the translational vector), $u_r$ (a unit vector describing the rotation axis), and $d_r$ (the rotation angle about $u_r$). The basis for resolving $u_r=[u_x, u_y, u_z]^T$ and $d_r$ is that the rotation sub-matrix of $D$ can be re-written in a form as (Craig, 1986):
\[
\Psi(u_r, d_r) = \begin{bmatrix}
    u_x^2 V \! d_r + C d_r & u_x u_y V \! d_r - u_z S d_r & u_x u_z V \! d_r + u_y S d_r \\
    u_x u_y V \! d_r + u_z S d_r & u_y^2 V \! d_r + C d_r & u_x u_y V \! d_r - u_y S d_r \\
    u_x u_z V \! d_r - u_z S d_r & u_y u_z V \! d_r + u_x S d_r & u_x^2 V \! d_r + C d_r
\end{bmatrix}
\]

where, \( S d_r = \sin(d_r) \), \( C d_r = \cos(d_r) \), and \( V d_r = \text{vers}(d_r) = 1 - \cos(d_r) \).

For a trajectory planning task aiming at achieving the ramp speed profile in figure 6.2.1, considering first the translational component, the trajectory planner needs to estimate the acceleration time and the cruising duration for the specified motion. Let \( s_a = v_t^2/a_t \) be the nominal translational distance covered during the acceleration and deceleration phases. If \( s_a \leq d_t \), the estimates of acceleration time and cruising duration are given by

\[
\hat{\tau}_{at} = v_t/a_t, \quad (6.2.2a)
\]

\[
\hat{\tau}_{ct} = (d_t - s_a)/v_t. \quad (6.2.2b)
\]

Otherwise, if \( s_a > d_t \), then the preset cruising velocity \( v_t \) will not be reached, and estimates of the acceleration time and the cruising duration become

\[
\hat{\tau}_{at} = \frac{1}{2} \sqrt{d_t/a_t}, \quad (6.2.3a)
\]

\[
\hat{\tau}_{ct} = 0. \quad (6.2.3b)
\]

The same computation may be repeated for the rotational component to obtain \( \hat{\tau}_{ar} \) and \( \hat{\tau}_{cr} \). Since the corresponding duration of each phases between the translational component and the rotational component will generally not be equal, the actual duration for the acceleration phase and the cruising phase are set to

\[
\tau_a = \max(\hat{\tau}_{at}, \hat{\tau}_{ar}), \quad (6.2.4a)
\]

\[
\tau_c = \max(\hat{\tau}_{ct}, \hat{\tau}_{cr}). \quad (6.2.4b)
\]
Chapter 6: On-Line Trajectory Adaptation and vision guidance

Having determined the $x_a$ and $x_t$ for the trajectory, the corresponding maximum speeds (or cruising speeds if the cruising phase exists), accelerations and path time $T$ are computed by

$$v_t = \frac{d_t}{x_a + x_c} u_r,$$  \hspace{1cm} (6.2.5a)

$$v_r = \frac{d_r}{x_a + x_c},$$  \hspace{1cm} (6.2.5b)

$$a_t = v_t / x_a,$$  \hspace{1cm} (6.2.5c)

$$a_r = v_r / x_a,$$  \hspace{1cm} (6.2.5d)

$$T = 2x_a + x_c. \hspace{1cm} (6.2.5e)$$

The planned trajectory $C(t)$ from $C_1$ to $C_2$ is then described in respect of time $t$ as

$$C(t) = \Omega(\Psi(u_r, q(t)) R_{c1}, p(t)), \quad t \in [t_0, T+t_0] \quad (6.2.6)$$

where $\Omega(\Psi(u_r, q(t)) R_{c1}, p(t))$ is a homogeneous transformation function with its position vector as $p(t)$ and its orientation sub-matrix as the result of $\Psi(u_r, q(t))$ right-multiplied with $R_{c1}$—the orientation sub-matrix of $C_1$; $t_0$ is the trajectory execution start time; and the rotation angle $q(t)$ and position vector $p(t)$ is calculated by equations:

$$q(t) = \begin{cases} 0.5a_t t^2, & t \in [t_0, x_a+t_0] \\ 0.5a_r x_a^2 + v_r t, & t \in [x_a, x_a+x_c] \\ 0.5a_r (x_a^2 + (t-x_c-x_a)^2) + v_r x_c, & t \in [T-x_a, T] \end{cases}, \quad (6.2.7a)$$

$$p(t) = \begin{cases} p_0 + 0.5a_t t^2, & t \in [t_0, x_a+t_0] \\ p_0 + 0.5a_r x_a^2 + v_r t, & t \in [x_a, x_a+x_c] \\ p_0 + 0.5a_r (x_a^2 + (t-x_c-x_a)^2) + v_r x_c, & t \in [T-x_a, T] \end{cases}, \quad (6.2.7b)$$

where $p_0$ is the position vector of $C_1$. 

PhD. 1994 Nongji Chen
Chapter 6: On-Line Trajectory Adaptation and vision guidance

The ramp profile in figure 6.2.1 can also be established in another mode which requires the acceleration duration and the cruising duration to be specified. The planner then computes the speed and the acceleration, and the corresponding magnitudes of them are checked to guarantee that no excessive acceleration or speed value is produced. In the event that a command fails to comply with the acceleration or speed boundary, the command is simply ignored by the planner and an error code is returned.

![Diagram of trajectory planning](image)

Figure 6.2.2 Illustration of trajectory planning with no path alteration.

6.2.2 On-line trajectory alteration

In its more conventional way of operation, a robot is seen to follow some sequential path segments in fulfilling a task. These path segments are defined by a series of command target positions which in turn are coded in a user application program. A fundamental feature of the relationship between any two consecutive path segments is that, in general, the next path segment will not be started until the motion of the first one is finished. Thus the starting and ending positions of any path segment are specifically defined by two consecutive motion commands. Such a feature simplifies the trajectory planning problem since it contains only one directional
information flow (see figure 6.2.2), which makes the implementation of a trajectory planner easily achievable by employing a simple pipe-line structure.

![Figure 6.2.3 On-line trajectory alteration](image)

**Figure 6.2.3 On-line trajectory alteration**

**Trajectory alteration: joint or translational element**

On-line trajectory alteration, on the other hand, requires the interaction between the trajectory planning and the trajectory execution. Consider the case illustrated in figure 6.2.3. The currently on-going motion trajectory is from C1 to C2. When the trajectory is executed to an intermediate point M, the occurrence of an event is sensed and the robot is requested to move to position C3 instead of C2. Obviously, a sensible solution to such a problem is to abandon the remaining part of the currently running trajectory and blend the motion into a new trajectory that heads to position C3. However, since M is an intermediate point of the running trajectory, the intended new trajectory’s starting point is not specified by a motion command. Consequently the trajectory planning part must interact with the execution part to retrieve the current execution point in undertaking the planning task. The information flow is therefore no longer in single direction any more as a route from the 'trajectory feeding'
block to the 'start position' block is added in comparison with figure 6.2.2. This increases complexity in practical implementations as the synchronisation issue between the two different repeating rate modules needs to be resolved.

A basic issue in achieving trajectory adaptation, as the case illustrated in figure 6.2.3, is to plan a new trajectory for the current on-going trajectory to be blended into and to generate a smooth transition phase that joins the two trajectories with at least no discontinuities in both position and velocity, and, at the same time, provide as small deviation from the trajectories as possible. The heuristic approach adopted in the open architecture robot controller uses cubic polynomials to provide such a transition in trajectory alteration. To elaborate a little bit further on the adaptation transition, a one dimensional case is considered as shown in figure 6.2.4.

Figure 6.2.4 Illustration of one dimensional trajectory alteration.
The initial running trajectory in figure 6.2.4 is from position C1 to position C2. When it is executed to position M, an alteration to position C3 is requested. In default, the planner of the controller assumes the new trajectory which leads to C3 consists of three phases with a speed profile as the one in figure 6.2.1. Its start point D is firstly estimated by

\[ D' = M + 0.5 \cdot V_m \cdot \tau' \]  \hspace{1cm} (6.2.8b)

where \( D' \) stands for the estimate of D; \( V_m \) is the velocity of the current running trajectory at M, and \( a_c \) is the preset acceleration magnitude. In essence, \( D' \) is the point at where the speed of the current running trajectory can be brought to zero by a deceleration with magnitude \( a_c \). Having obtained \( D' \), a preliminary three phase trajectory connecting \( D' \) to C3 can be established in accordance to section 6.2.1. Assuming the cruising phase of the preliminary trajectory starts from position \( N' \) with the cruising velocity being \( V_{n'} \), a half time of the duration required to change the velocity from \( V_m \) to \( V_{n'} \) is computed as

\[ \tau = 0.5(v_{n'} - V_m)/a_c \]  \hspace{1cm} (6.2.9)

The value \( \tau \) is then used to determine the starting point D of the new trajectory, which leads to

\[ D = M + 0.5 \cdot V_m \cdot \tau \]  \hspace{1cm} (6.2.10)

After the determination of position D, the new trajectory as well as its cruising phase start position N and cruising velocity \( V_n \) are computed. The coefficients of the transitional cubic polynomial are then established by

\[ \alpha_3 = 0.25(v_m + v_n)/\tau^2 - 0.25(N - M)/\tau^3 \]  \hspace{1cm} (6.2.11a)

\[ \alpha_2 = 0.75(N - M)/\tau^2 - 0.5(2v_m + v_n)/\tau \]  \hspace{1cm} (6.2.11b)

\[ \alpha_1 = v_m \]  \hspace{1cm} (6.2.11c)
Chapter 6: On-Line Trajectory Adaptation and vision guidance

\( a_0 = M \), \hspace{1cm} (6.2.11d)

and the blending phase, which joins the two trajectories with both continuities in position and velocity at M and N, is computed by

\[ \beta(t) = a_3t^3 + a_2t^2 + a_1t + a_0, \quad t \in [0, 2\pi]. \] \hspace{1cm} (6.2.12)

The adoption of cubic polynomials in achieving trajectory blending is advantageous in several aspects. Firstly, it provides the fundamental continuity in trajectory blending. Secondly, because of its low order nature, it behaves in a reasonably predictable fashion with very good closeness to the desired trajectories. Besides, the computation of the transitional phase is self-contained: nothing about the two joining trajectories is required any further beyond the coefficient determination. Additionally, it provides natural absorption to some small skews among the related positions, velocities, acceleration and transitional duration. These skews are likely to be existing due to the way by which D, N, \( v_n \) and \( \tau \) are computed.

The two step approach in obtaining D, N and \( \tau \) (i.e. firstly estimating and then re-evaluating) provides better velocity smoothness in joining the two trajectories as it can reduce the unnecessary speed fluctuation that may appear if the preliminary trajectory is directly used as the new trajectory. The reason is obvious by considering the case of C3 being very close to C2 (see figure 6.2.4) thus the current speed scarcely needs to change. If the second step of re-evaluation is not performed in the new trajectory computation, one would observe that the transition from the on-going trajectory to the newly generated trajectory experiences a speed reduction to zero and then recovers back. This unnecessary speed fluctuation is significantly reduced by the two step approach as its re-evaluation takes into consideration the speed differences of the two trajectories.
Trajectory alteration: rotational element

A more complex issue involved in achieving on-line trajectory alteration is to tackle the orientation change for Cartesian space trajectories. The requirement under such circumstances is that, not only the current running trajectory's angle of rotation, which is a time variable in a planned trajectory, needs to be joined through a transitional phase with its counterpart of the next trajectory, but also does the pair of the axis of the rotation, which themselves are constant vectors to their corresponding trajectories and in general not aligned to each other. There is no simple vector-like representation which offers a proper description of the combined rotation effect.

To simplify the handling of this complex problem, the trajectory planner in the open architecture robot controller treats the transitional change of the rotation axis and the rotation angle separately. More specifically, it employs a cubic polynomial to join the rotation angles in a way similar to the handling of a translational element described previously, and uses a linear equation to bring the transitional rotation axis from a position aligned with the rotation axis of the first trajectory to a position in alignment with the second trajectory's rotation axis. So far such an approach has been proven to be workable and no major problems have been encountered in vision guided operations in this research work.

Other trajectory alteration modes

In default, the trajectory planner generates the second trajectory with zero ending speed under trajectory adaptation circumstances. There are other operational modes which differ from the default mode in that either a specific time instant is specified for the trajectory to reach its ending target position, or the ending speed is non-zero, or both. The last one of the stated
non-default modes is intensively used in the vision guided dynamic interception described later. Its underlying principle is that it assumes the robot will reach the second target position at the given time instant and maintains the cruising velocity there. Consequently the new trajectory from M to C3 in figure 6.2.4 (illustrated in one dimension) consists of only an acceleration period and a cruising speed period. Because in this particular mode the time instant for reaching C3 is specified, say at \( t_{c3} \). The path duration of the new trajectory is defined by

\[
T = t_{c3} - t_m ,
\]

where \( T \) is the path duration and \( t_m \), which is retrieved from on-line trajectory execution, the time instant for current trajectory point M. Assume the velocity change is carried by an acceleration \( a_c \), the basic equations for the new trajectory’s calculation can be written as

\[
\begin{align*}
C_3 - M &= v_c \tau_c + 0.5 \tau_a (v_c + v_m) \\
T &= \tau_a + \tau_c \\
v_c &= v_m + \text{sign}(C_3 - M - v_m T) a_c \tau_a
\end{align*}
\]

where \( \tau_a \) is the acceleration duration; \( \tau_c \) the cruising phase duration; and \( v_c \) the cruising as well as the ending speed. Resolving these equations, \( \tau_c \) and \( \tau_a \) can be obtained as

\[
\begin{align*}
\tau_c &= \sqrt{T^2 - 2 \left| (C_3 - M - v_m T) / a_c \right|} , \\
\tau_a &= T - \tau_c
\end{align*}
\]

Obviously the existence of the above solution relies on \( T \) being a proper value to render a real number \( \tau_c \). This in turn is determined by the specified position reaching time instant \( t_{c3} \). In the event that the target position of C3 is unreachable for the specified \( t_{c3} \), the planner will report back an error and the trajectory alteration command will be ignored. Additionally, if at the time instant \( t_{c3} \), which is the end of the trajectory
execution time, no further motion command is received, the planner will automatically generate a deceleration phase to bring the robot to a smooth stop.

6.3 The Surrey stereo vision system and robot visual guidance

In order to make a robotic system function well in a much less constrained manufacturing environment, feedback from external sensors is essential. There is a wide range of sensors available that can be used for different tasks and, depending on their working principles, these sensors can be classified as contact or non-contact sensors. Machine vision is a technique that can be employed to establish a non-contact sensing mechanism for robot operations. Because of its analogy to human vision, as one might have expected, it promises to be the most powerful and effective general purpose sensing system in robot applications.

Robot vision systems supply valuable information that can be used to automate the manipulation of objects, to overcome manufacturing uncertainties, and to react to unexpected events. In broad terms, a robot vision system can be based on either 2D vision techniques or 3D vision techniques. 2D machine vision uses the analysis of single images, finding features of interest and then reasoning about these features and their relationship with the world model. It requires a sufficiently constrained environment to simplify the problem of reasoning about 3D objects from 2D data. 3D machine vision, on the other hand, can work in a much less constrained environment. It relies on correctly mapping image points from two (or more) views, the stereo correspondence problem. The advantage of 3D machine vision is the generality of the method while its disadvantage is the formidable data processing required. With the use of machine vision in a robot workcell, the position, orientation, identity and condition of each
workpiece in the scene can be obtained. This high-level information can then be used to plan robot motion such as determining how to intercept, to grasp and to manipulate a workpiece, or alternatively, to avoid collisions with obstacles. In general, to achieve such functionality, the vision sensor and the robot controller need to be integrated in a harmonic way and any interactions between them well coordinated.

6.3.1 The Surrey active stereo vision system

To achieve active vision, the sensor system must be equipped with the ability to vary imaging parameters to aid the performance of visual tasks. These include such parameters as the six degrees-of-freedom for the sensor position and orientation, optical lens parameters such as aperture, focus and zoom, and sensor parameters such as variable baseline and torsional control of the individual cameras. The choice of controllable parameters and their performance is largely made against the application although the cost of developing such systems often overrides this factor.

The Surrey active stereo vision head

The Surrey stereo vision head has been designed by Pretlove (1993) for use in a manufacturing environment, attached to the end-effector of an industrial robot and must therefore fulfill certain design constraints. The approach adopted in the design and development of the Surrey stereo head has been a pragmatic and integrated engineering approach which has resulted in a flexible solution features low cost, light weight and compactness. A picture of the stereo vision head is given in figure 6.3.1.

The stereo vision head has six controllable degrees-of-freedom. These are independent vergence, focusing control of both cameras and aperture control. As Figure 6.3.1 shows, the design consists of two lightweight CCD camera and lens sub-assemblies which are mounted in a light rigid
supporting frame. The frame also accommodates optical limit switches for the camera and lens sub-assemblies vergence mechanism, which are used for calibration and protection against over-travel. The vision head provides 579x583 pixels in resolution of each camera and the sensing range is 0.2 meters upwards (in the case here nominally 2 metres due to the robot working volume). To provide the 0.2 metres minimum sensing distance, the cameras are capable of rotating through a maximum of 60° inwards from having both optical axis parallel, converging to a point 0.2m away. Both the focus drive mechanism and the vergence drive mechanism employ geared low-cost dc servo-motors with optical shaft encoders.

Figure 6.3.1 A photograph of the Surrey Active Stereo Head.

The dc servo-motors are currently controlled by off-the-shelf dc motor driver and amplifier units housed in a free standing industrial racking system. These control units are accessed via an daisy-chained RS-232 serial link from a vision processing host machine. They are capable of implementing proportional, integral, derivative and feed forward (P.I.D.F)
Chapter 6: On-Line Trajectory Adaptation and vision guidance control, with the control parameters being configurable from the host. In normal operation each motor control unit works in response to its instructions from the vision processing system. Additionally a hand pendant has been provided allowing manual control and an emergency stop.

It has been observed that the above servo-motor control units suffer to some extent from their poor communication capability. This has posed a bottleneck in achieving dynamic vision guidance, limiting the attainable system performances. Additionally, as they are designed for general purpose single axis motion control, these control units stop short of providing the coordination and synchronisation between each of the six degrees-of-freedom of the vision head. To overcome these disadvantages, a transputer based controller for the stereo vision head is undergoing development. It is expected that this new head controller will be in operation very shortly and the system performance will be boosted.

The vision processing system

Currently the hardware of the vision processing system consists of a Sun 4/630 multi-processing UNIX workstation, ten dedicated image processing boards, and a transputer array. A high speed VME bus connects the Sun with the image processing boards and to the transputer array via a transputer link. This is used primarily for command and control of the image processing boards and the transputer array although it can transfer image data from the image processing boards to the Sun workstation.

The dedicated image processing boards use a proprietary, and now a quasi industrial standard, 10MHz bus, the Maxbus. This provides a flexible and reconfigurable data path between all of the boards. It also allows for a variety of processing paths, either straight pipelined, recirculation or...
multiple parallel paths. The full board set comprises two digimax boards for digitising and displaying images, two framestores for storing up to 3 x 512 images each, a region of interest store capable of storing up to 2Mb of image data in any format and on a pixel boundary. The image processing boards are an 8x8 convolution board, the vfir_II; a systolic neighbourhood array processor, SNAP; a general purpose board, MAX_SP; and a histogramming and feature detector board, FeatureMax. In addition a 64 point multiplexer, MAX_MUX, is used to dynamically reconfigure the image data paths on a pixel boundary. The two digimax boards and the two framestores are used to capture left and right images corresponding to the two cameras of the stereo head simultaneously. The digitised images are then available to the remaining MaxWare system, via the Maxbus, for further processing.

The parallel processing array provides further general processing capability to the dedicated image processing boards. It is designated to high level vision tasks such as stereo correspondence, three dimensional information recovery, and target determinations. The array also provides high-speed transputer links for inter-communication between the vision sensor system and the open architecture robot controller. A further transputer link communication channel will be established between the array and the vision head controller when the head controller improvement is completed.

The vision system works under the management of the Sun machine. By issuing commands to the memory mapped control registers of the dedicated image processing boards, the Sun has full control over these boards' working status. In addition, the Sun is also responsible for command and control of the currently working vision head motor driver controllers via one of its RS232 serial ports. A schematic diagram of the vision system hardware configuration is shown in figure 6.3.2.
Figure 6.3.2 Overview of the Surrey stereo vision system hardware.

6.3.2 Robot visual guidance

Robot vision system provides information about a workpiece within the robot workcell which can be used to effect a highly flexible operation. The Surrey active stereo vision system is capable of processing images from its two cameras by computer to extract information from the common field of view. This could be recognising objects for manipulation as well as inspection, or taking measurements of the position of objects and their spatial relationships so that a proper robot operation can then be planned. For tasks such as material handling, assembly and workpiece interception, it is necessary that the vision system provides 3D information which allows the robot to manipulate objects within the workcell. Such information typically includes the three dimensional position and orientation of the object and this must be further processed by a task manager which responds the sensory results by commanding one or more robot actions in accordance to a given task.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

A typical robot vision guided operation may be regarded as locating a target object within the workcell and then guiding the robot to 'pick' it up or to perform some processing on it. Precisely what robot responses will be to the sensory information are determined by the task manager which in general has a different procedure for each different task. In this research work, the discussion of vision guidance is confined in using the visual feedback to achieve a successful interception of a target, avoiding the diversity of various task managing issues. This, however, should not be regarded as a severe constraint on generality as other types of task would essentially be achieved by a replacement of the task managing procedure.

Static guidance

A simple case of vision guidance is to use the vision system to determine the position and orientation of a target and then initiate a robot move to the target position. The target itself is static and remains so for a duration long enough for the discussed robot operation to be completed. A typical scenario for such a case may be the palletising of products by a robot. The products come into the robot work cell on a conveyor system which is triggered to stop when at least one product is within the range of the robot.

In the static guidance case, the three dimensional position and orientation of the target is fixed with respect to the workcell coordinate system. Consequently, the guidance can be achieved simply by applying a 'look and move' strategy. As both the target and the robot position are assumed static under such a case, the start and end points of the motion are known priory to the beginning of the robot motion. Thus no on-line trajectory alteration will be involved. A flow chart describing the operational steps in static guidance is shown in figure 6.3.3.
Triggered by the presence of a target within the working range, the first step of the static guidance procedure is to recognise what the object is and to calculate its three dimensional position and orientation. This object position information is then passed on to the task manager module that coordinates the vision sensor and the robot controller to perform the task.

For the static guidance under discussion, the task manager is responsible for calculating the desired position of the robot end-effector or tooling, based upon the task definition, and then issues a command to the robot controller which produces a smooth trajectory and moves the end-effector to the desired position. In principle, static guidance can be achieved by a single execution of the guidance cycle outlined in figure 6.3.3. This, however, requires that the combined accuracy of the robot positioning and the vision system measurement is within the allowable tolerance of the task. If improved accuracy, noise reduction or the rejection of external disturbances...
is required, then the above guidance cycle needs to be repeated. For instance, a two stage approach would result in better final positioning accuracy but this would be at the cost of increased task operation duration. In such an approach, the vision-robot system uses the first guidance cycle to take the vision system and the end-effector to a pre-defined stand-off position, reasonably close to the target. At this point the task manager re-initiates another cycle to complete the job. This two stage approach makes use of the accuracy improvement of the vision system at a closer range.

The ultimate accuracy of a well managed static guidance with multi-cycle executions is largely determined by the vision sensor system. The effect of robot kinematic modelling error will virtually be eliminated as the guidance is doing small 'differential motion' when the end-effector and the vision sensor mounted on it is very close to the target. The advantages of the sensor on the end-effector scheme thus become prominent here due to the fact that the vision sensor system has better accuracy at closer range.

The static vision-based robot guidance model is suitable for applications where products have unknown or ill-defined position and orientation and can be stopped while the robot carries out its task. These techniques are not suitable where the object may move. This static vision-based robot guidance paradigm demonstrates the concept of interactive sensing for robot positioning but it is not a dynamic control system since each step is executed independently and in sequence.

**Dynamic guidance**

In the static guidance problem discussed previously, image grabbing in each cycle is assumed to be undertaken when the robot is not moving hence the vision sensor mounted on the end-effector is static as well. Such a case simplifies interactions between the sensor system and the robot controller.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

since very little synchronisation is required. When the target is moving, or the image grabbing is undertaken while the robot is not in stationary, or the combination of the both, the situation becomes much more complicated as the timing factor under such dynamic circumstances has an important role to play. Because of the inherent time delay in image processing, the position information of the target recovered by stereo vision is not instantly available. Consequently any motion in either target or robot leads the relative position to be different from that at the time instant the images are grabbed. The implication of this timing factor in achieving visual guidance is two fold. Firstly, in order to recover the target position in the world coordinate system at the time instant that the images are grabbed, the robot end-effector position (the vision sensor position) at that time instant must be recorded. Secondly, as the sensory information in available form is always lagging in time, some predictive scheme must be employed to extrapolate the target motion, based upon the motion history of the target, so that an interception can be successfully achieved.

A major difference between the static guidance and the dynamic guidance is that the latter is performed in a four dimensional space (Euclidean space plus the fourth dimension of time). The solution to the dynamic guidance problem hence requires the robot end-effector (tool) to be at the right place together with the condition of at the right time. In order for a manipulator to grasp an object moving along an unknown path, direct on-line interaction between the vision processing system and the robot controller is necessary. Consequently the robot’s on-going trajectory will be frequently adapted to reach new target positions. In a dynamic, visual feedback system the vision-based routines and control of the manipulator are executed in parallel, as shown in figure 6.3.4.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

The position estimate derived from the vision system via a prediction scheme is updated to the robot controller which then performs an on-line adaptation of its motion trajectory in accordance to the newly available sensory information. The process relies on the precise synchronisation of a number of events between the vision sensor system and the robot controller as both the target and the robot end-effector, which is mounted with the cameras, may be undergoing motion. This dynamic interaction between the sensory system and the robot controller is a critical issue in achieving dynamic guidance.

Dynamic manipulator guidance is advantageous in at least two respects. In the case of static objects, dynamic manipulator guidance is necessary to achieve reduced cycle times and to improve overall spatial accuracy. For the case of a moving object, dynamic manipulator guidance

Figure 6.3.4 Dynamic vision-based robot guidance procedure.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

becomes essential. This control regime relies heavily on the synchronisation between the vision system and the robot controller and on predictive control techniques to anticipate the future object motion.

Target motion prediction techniques

Essential to achieving dynamic guidance is the prediction of target motion. In general, some prior knowledge of the underlying motion of the target must be assumed in order to provide the basis for choosing a proper motion predictor. Typical examples of such assumptions include: near constant speed with marginal speed variation, no-constant speed but the speed change rate is very small, etc. The central point of these assumptions is that they must provide some sort of regularity, either in statistical description or in short period temporal feature, so that the future movement of the target can be reasonably described based on its motion history so far available via the sensory system. This requirement of describable motion is commonly met in manufacturing environment. For instance, many belt conveyor systems can be reasonably described as moving at a constant speed with white-noise-like speed variations.

The key issue in predicting the target motion is to establish a mathematical model which is updated whenever a new observation from the sensor is available. The framework for such a mathematical processing is frequently provided by either the Kalman filter theory (Brown, 1989) or the recursive least square solution approach (Willsky, 1979; Sage & Melsa, 1982). Both the two methods are very popular and the choice of which should be used seems to be dependent on both the available statistical information of the application case and the preference of the researcher.
6.4 System integration issues

In order to support a wide range of external sensors, the open architecture robot controller provides an interface at each of its three hierarchical levels, i.e. servo control, trajectory control and command programming levels. These interfaces act as servers which conform to the widely used client/server programming model. In achieving external sensor integration, the open architecture robot controller expects a sensor system to be a client of one of its interfaces in accordance to the client/server definition. The interactions between the sensor system and the controller are then undertaken in the form of responses from the controller via the corresponding server to the client's requests.

A fundamental feature of the client/server programming model is that the server never initiates any communication except to respond to a request from the client. On the client side, it will not proceed any further after sending a request to its server until the request has been serviced and a result has been sent back. This request-response relationship is maintained all the time through an established connection between a client/server pair. In a client/server implementation, the server provides a set of pre-declared services available to its clients. This service set fully defines the scope of the functionality that the server can support.

The physical medium for communicating to the open architecture robot controller's sensory interfaces can be either transputer links or the ethernet LAN. The former features high speed (20 Mbit/s on each direction) with the interfaces available at all the three functional levels. The latter is an earlier implementation based on Unix socket communication protocol and it only provides interfaces for trajectory control and command programming levels.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

The integration of the stereo vision system and the robot controller in this research work is achieved through the interface at the trajectory control level supported by transputer links. The vision system and the robot controller interacts with each other via a task manager which coordinates the operation of the two systems in accordance to the task definition. Given the generic nature of the vision sensory information, task managing is necessary as, for instance, intercepting and grasping an object requires different robot responses to that of tracking of a weld seam although the information provided by the vision system would be essentially the same.

An important issue in the above sensor-robot system integration is to synchronise the two systems. This is particularly true in performing dynamic guidance. Under such circumstances, both the dynamic target position determination (grabbing images, performing the image processing and calculating the 3D position of the target while the robot is moving) and the target interception requires the timing information so that the four dimensional task can be successfully performed. To help achieving precise timing in coordinated operations, the robot controller stamps each of its responding messages with a clock field of 32-bit in length for the client to reference or to establish a slave clock in synchronisation. This clock provides timing every 500 micro-second and the controller expects any request for its services provided by the interfaces to be referenced to this clock. For example, if a task requires the robot to move to a new position at a future time one second later than the current time instant, it can be achieved by issuing a command specifying the motion ending time at a clock number representing the current time instant plus extra 2000 clock periods (i.e. one second).
6.5 Summary

On-line trajectory adaptation is crucial in achieving external sensor-based robotic applications. It provides the basis for the sensor-robot system to react to environmental change or manufacturing uncertainties. In this chapter, the trajectory planning scheme and the on-line trajectory alteration strategy of the open architecture robot controller are outlined. They are heuristic in nature and have a reasonable computational complexity for on-line processing.

Also discussed in this chapter are the static and dynamic robot guidance by integrating the Surrey active stereo vision system, developed by Pretlove (1993), with the open architecture robot controller. In the case of static guidance, the stereo vision system determines the 3D position of the object and then instructs the robot to move. It works in a look and move style which consists of a sequence of independent steps. These can be repeated to improve the overall positioning accuracy at the cost of increased cycle time.

In dynamic guidance the vision-based target determination and the robot motion occur in parallel. The new target position estimates are updated to the robot controller as fast as they are generated and these immediately effect the motion of the robot arm through trajectory adaptation. The technical difficulties associated with this control regime are concerned with the inherent delays in the vision system processing the data, which varies with the complexity of the images, and the correct timing that must be considered in order to perform the task in a four dimensional space. To achieve smooth end-effector motion it is necessary to incorporate predictive filters to compensate for the image processing delay in dynamic guidance.
Chapter 6: On-Line Trajectory Adaptation and vision guidance

Issues involved in sensor-robot integration is another topic discussed in this chapter. In brief, the open architecture robot controller provides three level interfaces for external sensor integration. The interfaces conform to the client/server programming model and at the controller side they always act as servers. Both the transputer link based implementation and the ethernet based implementation of these interfaces are provided.

Experimental tests which demonstrate the visual guidance by employing the Surrey stereo vision head and the open architecture robot controller are presented in the following chapter.
CHAPTER 7
EXPERIMENTS

7.1 Introduction

In an effort to improve the performance of robot systems and to make robot systems more versatile in a flexible manufacturing environment, a number of issues involved in robot control and sensor-robot integration have been discussed in previous chapters. Most notably, an open architecture robot controller has been designed and developed and its integration with a stereo-vision head to yield visual guidance has been achieved. By using the developed robot control platform and the stereo-vision system, a series of experiments have been conducted. Each of the experiments demonstrates or validates some aspect of the open architecture robot controller, the integrated vision-robot system and some other issues discussed. The experiments culminate in a simulation of an industrial workcell capable of tracking and intercepting moving objects which requires dynamic interactions between the stereo-vision system and the robot controller.

The experiments were conducted in the laboratory of the Mechatronic Systems and Robotics Research Group at the University of Surrey. All of the tests use the same robot control platform which consists of a Unimation Puma 560 mark III industrial robot and the open architecture robot controller described in chapter 3. Additional equipment is used to perform the robot positioning accuracy tests (ISO standards) and to achieve visual guidance. More specifically, for the ISO position accuracy tests, a laser triangulation measurement system OPTOTRAC (Mayer & Parker, 1988) is employed to measure the robot end-effector position. For the visual
guidance tests, a motorised slideway is used and the active stereo vision system briefly described in Chapter 6 is another major component of the experimental set-up. The stereo vision head is attached to the end-effector mounting flange of the robot and the cables to the vision system's host computer are attached to the robot arm at suitable positions. The motorised slideway is placed on top of a robot work table which is positioned in the robot's working envelope so that the end-effector can reach all parts of the work table. For the visual guidance experiments, the target object used is a high-contrast cardboard shape which is fixed on the slideway. Additionally, the robot table is covered with black cardboard to enhance the contrast between the target object and the background. Figure 7.1.1 and 7.1.2 illustrate the open architecture robot controller crate and the visual guidance test workcell respectively.

This chapter reports on the experiments that have been conducted, which cover three areas, i.e. kinematic accuracy, dynamic control and visual guidance. They reflect and validate the research work reported in previous chapters.

Figure 7.1.1 A photograph of the open architecture robot controller crate.
Chapter 7: Experiments

Figure 7.1.2 A photograph of the experimental set-up.

7.2 The ISO pose accuracy tests

7.2.1 Objectives

The objectives of the experiments here were to perform on-line evaluation of the inverse kinematics algorithm presented in Chapter 4 and to demonstrate the capability of the open architecture robot controller in accommodating user defined kinematic control modules. Additionally, the resultant positioning accuracy of a calibrated robot model against its nominal counterpart was also assessed in terms of the ISO Pose Accuracy (ISO 9283, 1990).
7.2.2 Description of the experiments

The ISO test consisted of driving a manipulator to five test points which lie on one of several planes within the ISO test region. Generally the test region is the largest cube that fits within the most usable portion of the robot's working volume which has its sides parallel to the base axes of the robot. For the Puma 560 robot tested in the experiments, the five test points are shown in figure 7.2.1 with the position coordinates defined in table 7.2.1.

![Figure 7.2.1 ISO test cube and the 5 test points for Puma 560 robot.](image)

**Table 7.2.1 Positions of the five test points in robot base frame**

<table>
<thead>
<tr>
<th>Position</th>
<th>X (mm)</th>
<th>Y (mm)</th>
<th>Z (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.0</td>
<td>543.25</td>
<td>63.81</td>
</tr>
<tr>
<td>P2</td>
<td>250.0</td>
<td>793.25</td>
<td>-186.19</td>
</tr>
<tr>
<td>P3</td>
<td>-250.0</td>
<td>793.25</td>
<td>-186.19</td>
</tr>
<tr>
<td>P4</td>
<td>-250.0</td>
<td>293.25</td>
<td>313.81</td>
</tr>
<tr>
<td>P5</td>
<td>250.0</td>
<td>293.25</td>
<td>313.81</td>
</tr>
</tbody>
</table>

In order to measure the robot end-effector position, an optical, non-contact 3D motion tracking instrument—OPTOTRAC (Gilby and Parker,
Chapter 7: Experiments

1984; Mayer and Parker, 1988) developed at the University of Surrey was employed in undertaking the ISO tests. The instrument contains two optical sub-systems and it works in conjunction with a reflective cat-eye which is attached to the robot end-effector. Each of the two optical sub-systems emits a visible low-powered (class 2) laser and tracks the reflection from the cat-eye via a set of internal motorised micro-mirrors. The robot end-effector position (the cat-eye position) is then determined by triangulation based on the two laser beams which are inferred through the rotation angles of the internal micro-mirrors. The measurement accuracy of the OPTOTRAC is high enough for robot position accuracy assessment and calibration purposes over the normal operational range, despite the fact that the accuracy of the instrument degrades slightly when the distance between it and the robot end-effector increases. To give a reference figure, the static deterministic error of the instrument over a nominal working volume of one cubic meter is approximately $\pm 0.06\text{mm}$ (Mayer, 1991).

The ISO position accuracy test was conducted twice using the nominal Puma 560 kinematic model and a calibrated kinematic model respectively. Both the tests used the open architecture robot controller to drive the Puma manipulator which was part of the controller-robot platform. The kinematic calibration of the Puma 560 robot was conducted by Stanton (1991) and the resultant model parameters have been tabulated in Chapter 4. To resolve the inverse kinematics problem associated with the calibrated kinematic model, the algorithm derived in Chapter 4 was applied, enabling an on-line solution to be achieved for the kinematic control of the robot.

The experiment contained 30 repeated cycles in each test procedure. Within each cycle, the robot was commanded to move from one ISO position point to the next one and the actual position reached by the robot end-effector in Cartesian space was measured and recorded by the
Chapter 7: Experiments

OPTOTRAC system. During the test, the robot was programmed to follow a fixed sequence of P1-P2-P3-P4-P5, avoiding the problem of indeterminacy in approaching directions for the ISO test points.

7.2.3 Results

The experimental results on the ISO Pose Accuracy are shown in Table 7.2.2.

<table>
<thead>
<tr>
<th>ISO Pose Accuracy</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Model</td>
<td>839 μm</td>
<td>1525 μm</td>
<td>1038 μm</td>
<td>2089 μm</td>
<td>1089 μm</td>
</tr>
<tr>
<td>Calibrated Model</td>
<td>767 μm</td>
<td>1343 μm</td>
<td>914 μm</td>
<td>1870 μm</td>
<td>995 μm</td>
</tr>
</tbody>
</table>

As can be seen in table 7.2.2, the calibrated kinematic model only shows marginal improvement over the nominal kinematic model for the particular Puma 560 robot tested. This unexpected result suggests that the extracted kinematic model through the calibration process does not fully characterise the underlying kinematic structure of this Puma manipulator, since the measured position accuracy is less than the measured position repeatability which is in the order of 200 μm for all the position measurements. Further examination of the Puma 560 manipulator identified that joint 4 has a significant backlash. It is very likely that this backlash has affected the accuracy of the calibration result, given that the calibration procedure adopted by Stanton (1991) relies on the remaining axes being still when determining the axis direction of a rotating joint.

7.2.4 Conclusions

The experiments have demonstrated that the open architecture robot controller provides a means for users to adopt calibrated kinematic models
or to incorporate any other modifications to the kinematic control requirement. Additionally, the successful implementation of the on-line kinematic control based on the calibrated kinematic model amply validated the algorithm derived in Chapter 4 for solving the inverse kinematics problem.

The improvement in terms of the ISO Pose Accuracy of the calibrated kinematic model over its nominal counterpart was relatively small in this particular case. Yet it nonetheless does reflect the fact to an extent that a calibrated kinematic model can yield better position accuracy. The case also highlights the importance of pre-checking the robot for backlash and other nonlinear effects and making corresponding adjustments on the manipulator before a robot kinematic calibration procedure is undertaken so that the calibrated result more closely resembles the underlying kinematic structure of the manipulator.

7.3 Servo control tests

7.3.1 Objectives

The objectives of the servo control experiment tests were to perform on-line evaluation of the robust adaptive tracking control algorithm presented in Chapter 5 and to demonstrate the capability of the open architecture robot controller in forming an advanced robot control algorithm test-bed for research purposes.

7.3.2 Description of the experiments

The servo control experiments were conducted on the robot control platform that consisted of the open architecture robot controller and the Puma 560 robot. The experiments used the controller to perform the dynamic control of the Puma robot while tracking a given sinusoidal trajectory.
trajectory in joint space. Both the adaptive control algorithm derived in Chapter 5 and a PID control algorithm were used in the experiments and the corresponding results were recorded to give a performance comparison between the two control algorithms.

The Puma robot was brought to an initial position and maintained there prior to the trajectory tracking control experiment. This initial position was arbitrarily chosen at a point close to the centre of the operational volume within which the visual guidance system was operative. At the initial position, the Puma 560 manipulator was extended with the robot wrist facing downwards. The joint angles of the three major joints—joint 1 to joint 3 were -90.7470, -128.6720 and 19.2470 degrees respectively at the initial position.

The reference trajectory was given in joint space for each of the joints. More specifically, joint 1 and joint 2 were required to follow a sinusoidal change of their joint references while joint 3 to joint 6 were set to maintain their joint angles unchanged. The formulae of the reference trajectory signals for joint 1 and joint 2 were given as (in degrees):

\[
\begin{align*}
J_1 &= 10.0(1-\cos(\pi t)) - 90.7470; \\
J_2 &= 10.0(1-\cos(\pi t)) - 128.6720.
\end{align*}
\]

The PD (proportional and differential) feedback gains for both the PID control algorithm and the adaptive tracking control algorithm were identical. They were initially selected empirically to give a near critically damped response to the individual joints when in independent motion, and then re-adjusted to accommodate the coupling and non-linearity of the robot dynamic behaviour so that the PID control algorithm gave a reasonably good performance over a wide region. The underlying consideration here was that the emphasis was placed on achieving a
relative performance comparison. Therefore, no major effort was made to adjust the PD feedback gains to a better set of values that might bring about higher tracking control performance. After the PID control parameters had been tuned, the PD parameters were directly ported to the adaptive control algorithm. The remaining parameters of the adaptive control algorithm, required in equation (5.3.34) to (5.3.36) of Chapter 5, were then added and tuned empirically. Table 7.3.1 lists the control parameters used in the experiments.

Table 7.3.1 Control parameters in the servo control experiments

<table>
<thead>
<tr>
<th>Joint</th>
<th>J1</th>
<th>J2</th>
<th>J3</th>
<th>J4</th>
<th>J5</th>
<th>J6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kp (v/deg)</td>
<td>67.94</td>
<td>125.34</td>
<td>87.41</td>
<td>51.57</td>
<td>48.76</td>
<td>41.61</td>
</tr>
<tr>
<td>Kd (v•s/deg)</td>
<td>101.91</td>
<td>140.38</td>
<td>82.25</td>
<td>42.73</td>
<td>40.34</td>
<td>52.80</td>
</tr>
<tr>
<td>Ki (PID) (v/deg•s)</td>
<td>13.59</td>
<td>16.71</td>
<td>11.65</td>
<td>16.50</td>
<td>15.60</td>
<td>16.64</td>
</tr>
<tr>
<td>Symbol</td>
<td>α</td>
<td>γ1</td>
<td>γ2</td>
<td>γ3</td>
<td>σ</td>
<td>Γ</td>
</tr>
<tr>
<td>Value</td>
<td>4.65</td>
<td>0.0044</td>
<td>0.0044</td>
<td>0.000313</td>
<td>0.001</td>
<td>1</td>
</tr>
</tbody>
</table>

Unit symbols: v - volts (for power amplifier input signal, proportional to torque); s - second; deg - degree

Both the PID control and the adaptive control experiments were conducted with the computation of the control signal repeating at a sampling rate of 2 kHz. To provide the required computational power for such a high complexity data processing case, the i860 vector processor and two T805 transputers in the open architecture robot controller were employed to run the adaptive control algorithm.

The dynamic responses of the robot was recorded by the robot controller during the experiments. The position of each joint was sensed by
the joint optical encoder and the resultant output was a digitised representation of the joint angle. Consequently there inevitably existed some sensor noise which lead to one bit uncertainty in the digital readings.

### 7.3.3 Results

In this section, the dynamic control results given by both the PID control algorithm and the adaptive tracking control algorithm are presented. Because of the small impact on the wrist joints (joint 4, joint 5 and joint 6) by the tracking activities of joint 1 and joint 2, the experimental results are concentrated on the three major link axes of the Puma 560 robot.

![Figure 7.3.1 Trajectory reference signals in servo control experiments.](image)

The trajectory reference signals for joint 1 and joint 2 are illustrated in figure 7.3.1. The reference signal for joint 3 is omitted because it is simply a horizontal line. Both the PID control and the adaptive control achieved close tracking of the reference signals, and the results, if plotted, would exactly lay on the curves in figure 7.3.1 for the given scale. In order to
illustrate the experimental results more clearly, the trajectory tracking error is used in this section.

![Figure 7.3.2 Tracking errors for the three major joints in experiments.](image)

The trajectory tracking errors of the three major joints of the Puma 560 manipulator are shown in figure 7.3.2. It is evident from the figure that the adaptive control algorithm outperforms its PID counterpart. This is reflected in two aspects. Firstly, the maximum error for all the three joints is
Chapter 7: Experiments

reduced by a third when compared to the PID control results. Secondly, the coupling effect is relatively less significant as suggested by the results of joint 3, to which the coupling force is the major source of disturbance under the experimental conditions.

To give a more quantitative description of the tracking performance, a scalar index of the error signal is computed for each of the three major joints. The performance index calculation formula is defined as

\[ Q(t) = \sqrt{\frac{t - t_0}{t - t_0} \int_{t_0}^{t} e(t) \cdot e(t) dt} \]

Thus the performance index \( Q \) measures the root-mean-square ‘average’ of the tracking error, and a smaller index value represents better performance. Table 7.3.1 presents the corresponding performance index values of the experiments conducted and the ratios of them in percentage terms between the adaptive control result and the PID control result.

Table 7.3.1 Tracking control performance index value

<table>
<thead>
<tr>
<th>Joint 1</th>
<th>Joint 2</th>
<th>Joint 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(t) (PID Control)</td>
<td>0.007287 (deg)</td>
<td>0.005263 (deg)</td>
</tr>
<tr>
<td>Q(t) (Adaptive Control)</td>
<td>0.003834 (deg)</td>
<td>0.002399 (deg)</td>
</tr>
<tr>
<td>Ratio (Adaptive/PID)</td>
<td>52.61%</td>
<td>45.58%</td>
</tr>
</tbody>
</table>

The values in table 7.3.1 clearly show that the tracking performance of the adaptive control algorithm is superior to that of the conventional PID control algorithm for the experimental results. The root-mean-square average error is nearly 1:2 in amplitude.

7.3.4 Conclusions

The experiments validated the real-time applicability of the robust adaptive tracking control algorithm derived in Chapter 5 and provided a
performance comparison between the proposed adaptive control algorithm and a conventional PID control algorithm. As expected the adaptive tracking control algorithm demonstrated better performance than that of the PID control algorithm. This improvement was obtained at the cost of a much higher computational requirement. It has been observed from the experiments that, although the relative performance improvement of the adaptive control algorithm over its PID counterpart is fairly significant, the errors of both the PID control scheme and the adaptive control scheme are rather small in absolute terms, which in fact are near the boundary of the sensor resolution. Therefore it seems that any further effective performance improvement in practical cases requires an improvement to the sensor resolution as well as accuracy.

7.4 Static visual guidance tests

7.4.1 Objectives

The objectives of the static visual guidance experiments were to validate the external sensor interaction interface of the open architecture robot controller and to demonstrate the integration of the robot control platform, which consisted of the controller and a Puma manipulator, with the Surrey stereo vision system in achieving external sensory guidance.

7.4.2 Description of the experiments

The experimental set-up, as described in the introduction of this chapter, consisted of the open architecture robot controller, a Puma 560 industrial robot and the Surrey stereo robot vision system. The active stereo vision head was mounted on the end-effector of the Puma manipulator and the robot wrist was oriented in an appropriate direction so that the vision head was downward looking with the robot work table in the field of view.
Chapter 7: Experiments

The end-effector of the Puma was approximately 900mm away from the robot work table. The target for the experiments consisted of a white cardboard rectangle approximately 40x50mm which was placed on the robot work table. The robot work table was covered with black cardboard which extended to fill the entire field of view of the cameras when viewed from the nominal distance. Having such a high contrast object against a neutral background aided the vision system in segmenting and identifying the object from its environment. This aided the image processing system and also removed the need to tackle the problem of stereo correspondence. Attached to the extended robot toolpost was a simulated tool which consisted of a spring loaded pointer. This was used to visually check that the robot controller had moved the end-effector to the correct position.

Two experiments have been performed to demonstrate the capability of the integrated vision-robot system in achieving static visual guidance. In the first experiment the robot started from a random position, generated by adding a random distance value uniformly distributed over the range of -100mm to 100mm to the X, Y, Z values of a pre-determined home position. It attempted to touch the target’s centroid in a single cycle that consisted of determining the position of the target and then moving the robot. For the same static target this experiment was repeated 30 times. In the second experiment, designed to demonstrate the accuracy improvement issue by bringing the robot vision head to a closer distance to the target, a two stage approach was adopted. The robot was first moved from a random start position to a closer, and more advantageous, viewing position. This stand-off position was calculated by adding an offset to the target position determined by the vision system at the random start position of the first stage, and served as the start position for the second move. When the robot reached the stand-off position, a new relative target position was re-
calculated by the vision system and a second move of the robot was made to touch the target. Again the second experiment was repeated 30 times.

The experiments reflected the basic idea of employing an external sensor system to overcome workcell uncertainties as no a priori position information about the target on the worktable was assumed. The vision system was fully responsible for determining the target position and bringing the robot to the right place for the intended task. As the target was stationed statically within the workcell, the experimental results indicated the accuracy issue of the integrated vision-robot system and the possible improvement in accuracy by adopting a multi-stage approach.

7.4.3 Results

The results of the two experiments are shown in figure 7.4.1. For the case of the single shot experiments, under the conditions described, the accuracy was of the order of ±5mm in X, Y and Z. For the case of the two-shot experiment the accuracy was of the order of ±1mm. Although the accuracy's are poor, this is a clear indication of the improvement that is obtained by a multi-stage method. The main reason for the relatively poor accuracy is considered to be caused by the fact that the vision head has not been fully calibrated. All the calculations are currently based on the nominal design parameters of the vision head structure.

The position of the end-effector was measured by the robot itself in the experiments. The actual position of the target object was obtained by moving the simulated tool to the target’s centroid using the robot’s teach pendant. The position of the robot was then obtained from the robot controller to calculate the target position by plus the tool offset to it.
Figure 7.4.1 The accuracy results for single and two shot robot guidance.

7.4.4 Conclusions

As expected the accuracy of the two shot approach was considerably better than the single shot technique from the same starting conditions. In the experiments carried out the relative accuracy improvement was about five times. The disadvantage of the two shot method was the increased cycle time which was difficult to estimate due to the speed of the robot, the range, etc. The additional time consisted of stopping the robot at a second position, grabbing two images and processing them, and then moving the robot for the intercept. With improved calibration techniques an improvement of the overall accuracy should be obtainable. For tasks where accuracy is more important than throughput the two-shot approach offers significant improvements while still maintaining static sensor-robot interactions. It should be noted that a more desirable approach to achieving higher
accuracy without increasing the task cycle time is to execute the second viewing while the end-effector is still moving—the case of dynamic guidance. This requires the integrated system to be able to support dynamic interactions with very fast inter-system synchronisation.

### 7.5 Dynamic visual guidance test

#### 7.5.1 Objectives

The objectives of the experiment were to validate the integration of the open architecture robot controller and the Surrey stereo vision system as well as their dynamic interactions in achieving a dynamic tracking and intercepting task in a simulated manufacturing workcell. It was aimed at providing a realistic demonstration of how the vision guided robot system may be used in a manufacturing environment.

#### 7.5.2 Description of the experiment

The experimental set-up was the same as that of the static visual guidance test described in section 7.4 except that a motorised linear slideway was added to provide the dynamic movement of the target. The target was fixed on the slideway which could be programmed to run at various speeds.

The robot end-effector, with the active stereo vision head mounted on it, was initially stationed at a 'home' position where the stereo vision head had a wide view of the workcell and the whole system was working in a tracking mode. When the object entered the field of view of the vision cameras, the 3D position of the target relative to the end-effector of the robot was calculated and the resultant information fed to a least square estimator which predicted the future positions of the target through previous position history of the moving target. The predicted 3D position was then used to effect the robot motion to achieve tracking and approaching the target. This
process continued while both the target and the robot were in motion. Finally when the simulated robot tool had reached close proximity to the moving target, an interception action was activated. The interception was simulated by bringing the robot tool to touch the moving target and maintaining them in a relative motionless condition for about two seconds.

To achieve dynamic tracking and interception, the vision system needed to know where the stereo vision head was positioned when an image grabbing action was initiated. Additionally, the robot end-effector had to reach the right position at the right time to accomplish the task. Both the issues demanded precise synchronisation between the robot controller and the vision system, and this correct timing requirement significantly increased the complexity of the dynamic guidance case from its static counterpart. Besides the synchronisation issue, the use of some sort of position predictor or estimator was inevitable given that the vision processing took a significant time to complete. In the experiment, three four parameter autoregressive (AR) processes were employed to predict the target motion in X, Y and Z directions respectively. Their basic form is

$$\hat{x}(k+1) = a_3 \cdot x(k) + a_2 \cdot x(k-1) + a_1 \cdot x(k-2) + a_0,$$

where $\hat{x}(k+1)$ is an estimation of the next position while $a_0$, $a_1$, $a_2$ and $a_3$ are the four parameters that are updated on-line in accordance to recursive least square algorithm (Willsky, 1979; Sage & Melsa, 1982).

The vision system was interfaced to the open architecture robot controller via the trajectory level interface of the latter. A pair of transputer differential links were employed to accomplish the communication, which provided a high speed interaction communication channel. By using this communication channel, the synchronisation between the vision system and the controller reached a resolution of a millisecond, which was
sufficiently small to meet the dynamic interactions' requirement of the experiment under discussion.

The on-line trajectory adaptation capability of the open architecture robot controller provided the necessary support for achieving smooth approaching to the target in the dynamic guidance experiment. At each target position updating cycle, the predicted target position together with its anticipated reaching time information, which was worked out according to a pre-determined approaching strategy, was forwarded to the robot controller to generate a timed smooth motion. By repeating this timed trajectory adaptation process, the robot eventually reached and kept still with the target, completing the tracking and intercepting procedure.

7.5.3 Results

![Figure 7.5.1 Dynamic guidance experimental result in X direction.](image)

The results of the dynamic guidance experiment are shown in figure 7.5.1 to figure 7.5.3. Each figure illustrates the recorded target and robot position with regarding to time in one dimensional form in the workcell.
world coordinate system. As can be seen from these figures, the robot approaches the moving target with a nice smooth profile and finally performs the interception.

Figure 7.5.2 Dynamic guidance experimental result in Y direction.

Figure 7.5.3 Dynamic guidance experimental result in Z direction.
7.5.4 Conclusions

This experiment demonstrated that the open architecture robot controller could interface to an external sensor system to achieve dynamic interactions. It also demonstrated the effectiveness of the on-line trajectory adaptation scheme adopted in the controller. The experiment validated a number of design considerations of the robot controller, such as timing issues and interface protocols, and showed that the integration of a vision system with a robot controller could overcome various uncertainties that may exist in a manufacturing environment.

7.6 Summary

A series of experiments have been conducted to demonstrate the capabilities of the open architecture robot controller and the integrated vision-robot system that consists of the robot controller and the Surrey stereo robot vision system. Each of the experiments demonstrates or validates at least one of the issues discussed in previous chapters of this research work.

The ISO pose accuracy experiments have demonstrated that the open architecture robot controller provides access for users to adopt calibrated kinematic models or to incorporate any other modifications to the kinematic control requirement. It also validates the inverse kinematics algorithm derived in Chapter 4 for solving the inverse kinematics problem when a calibrated robot kinematic model is employed. The experimental results prove that kinematic calibration techniques do have the potential to improve robot positioning accuracy, and indicate that certain pre-adjusting procedures, such as gear backlash eliminating, on a robot may be required before committing a calibration process on the robot.
Chapter 7: Experiments

The servo control experiments prove that the adaptive tracking control algorithm is workable in real-world cases and provides a significant performance improvement judged by the root-mean-square 'average' error in relative terms over the conventional PID control algorithm. The experimental results also seem to imply that further effective servo dynamic control performance improvement in absolute terms may require higher resolution joint sensors on the Puma manipulator.

The static visual guidance experiments are designed to illustrate the accuracy issue of the integrated system in reaching a static 3D object in the workcell from a variety of different starting positions. The experiment involves vision system calculating the position of the object and then instructing the robot to move a tool to the centroid of the target. In the first experiment the touch of the object is achieved in a single movement, while in the second experiment the estimated 3D position is used to move the vision system to a more advantageous viewing position and initiates another measurement and move cycle to fulfil the task. The accuracy is improved in these experiments by an order of five times but at the cost of increased cycle time.

The dynamic visual guidance experiment demonstrates the ability of the integrated vision-robot system to perform the tracking and intercepting task on a moving object in the workcell. It validates the on-line trajectory adaptation scheme implemented in the open architecture robot controller. In the experiment, a least square estimator is used to predict the trajectory of the object which helps to overcome the inherent delays of the image processing system. It also reduces the effects of the vision sensor system noise that inevitably exists. The experiment simulates a manufacturing workcell where an object would pass through the workcell and requires an manufacturing operation to be performed during the pass.
CHAPTER 8
CONCLUSIONS

High performance robot systems are essential to reduce the difficulty in developing flexible manufacturing systems and to improve the cost effectiveness in medium and small batch sized manufacturing work. In general, industrial robots used today have only achieved limited success in living up to the expectation that originated from the flexible manufacturing requirement. Significant performance enhancement to current industrial robots is widely considered to be required to relax the constraint on their application environment.

The work described in previous chapters attempts to improve the performance of industrial robot systems by addressing a number of issues related to external sensory feedback and sensor integration, robot kinematic positioning accuracy, and robot dynamic control performance. The emphasis has been placed on establishing an advanced robot control platform and, by integrating it with a robot vision sensor, demonstrating the concept that an industrial robot system equipped with external sensors can 'intelligently' react to its environment. Such 'intelligence' is expected to significantly enhance the performance of robot systems in a flexible manufacturing environment as both uncertainties and unexpected events in the workcell can be dealt with.

This Chapter presents the conclusions of this research work and puts forward some suggestions for additional research that may bring about further improvements.
8.1 Conclusions

The work described demonstrates that the concept of integrating a stereo vision sensor with an industrial robot can lead to a general purpose flexible robotic system for use in a manufacturing workcell. The external vision sensor provides informative workcell feedback which greatly enhances the ability of the robot system to cope with variations or unforeseen circumstances. The fundamental difference between such an external sensor based robot system and a traditional one is that the former operates in a closed vision loop with the actual workpiece location (moving or static), thereby reducing the uncertainties related to the robot characteristics and the geometric and dynamic relationship between the robot and the workpiece location. Consequently the constraints on the robot operational environment can be significantly reduced, allowing, for example, randomly oriented workpieces arriving into the workcell on a conveyor system to be handled.

An open architecture robot controller has been designed and developed in this research. It employs a high performance transputer network (including an i860 vector processor based tram) and an MC68030 single board computer to provide the computational power required by modern robotics. The architecture of the controller described in Chapter 2 is a departure from previously developed robot controller systems. This has resulted in the controller featuring easy expansion in both computational power and peripheral hardware and being applicable to a range of robot manipulators. The controller system software has a default operational environment for general users of the open architecture robot controller. User applications can be programmed in a 'move to position' style and the compiled C program can be executed at a run-time user interface. The controller provides user accessibility to various robot control levels,
enabling users to make modifications to the default robot control modules for their particular application requirements.

To facilitate the incorporation of external sensors, the controller provides an interface at each of its three hierarchical levels, i.e. servo control, trajectory control and command programming levels. These interfaces act as servers which conform to the widely used client/server programming model. External sensor integration can be achieved either through these pre-defined interfaces or by exploiting the user accessibility to the controller modules at different robot control levels. By adopting such an approach, it is considered that the robot controller can accommodate a wide range of external sensors.

A heuristic on-line trajectory generation and alteration strategy has been devised and implemented in the open architecture robot controller to provide trajectory adaptation capability. The novelty of the scheme is reflected in its on-line ability to adapt robot trajectory with accurate timing synchronisation, enabling external sensor guided robot operations. It has a modest computational complexity and the implementation supports a number of different operational modes. Essentially, all these modes can be categorised into two types, one resulting in a time prioritised trajectory and the other a speed prioritised trajectory. They provide convenience for different scenarios. For example, in the experiment involving dynamic visual guidance, the time prioritised trajectory generating modes are used to precisely synchronise the robot motion with the target motion. It would be harder to achieve if the speed prioritised trajectory generating modes were used, as the acceleration and deceleration period may result in an averaging speed different to the speed intended.
Chapter 8: Conclusions

The integration of the open architecture robot controller with the Surrey stereo vision system to achieve visual guidance has been undertaken in this research. Both static guidance and dynamic guidance experiments have been conducted. In the static case, it has been demonstrated that the accuracy can be improved by adopting a multi-stage interception approach. As much as about five times improvement in accuracy was obtained in the two shot experiment described in Chapter 7. This improvement came at the cost of increased cycle time. Dynamic guidance is a more appropriate way to achieve higher accuracy without reducing the throughput rate. It has been identified that the synchronisation issue plays an important role in achieving dynamic guidance. In general, a high speed communication channel between the vision system and the controller is required. Additionally, a predictor has to be employed to counteract the effect caused by the inherent time delay in image processing if the workpiece is not stationary.

A simple and effective method of calculating the inverse kinematics solution for calibrated industrial robots has been described in this research. The new method exploits the nominal simple structure of an industrial manipulator to iteratively approach a shifted pose which leads to an acceptable inverse kinematics solution to the calibrated robot model. It is particularly suitable for implementation in on-line operations. To make the proposed method applicable to an S-model based calibration case, model conversion techniques have been established to describe a pair of near parallel links, avoiding the problem of discontinuity in the model description experienced in DH conversion when one of the axes is perturbed away from the parallel case. Experiments that test the ISO pose accuracy on the calibrated Puma 560 robot have been undertaken. The results prove that kinematic calibration techniques do have the potential to improve robot positioning accuracy, and indicate that certain pre-adjusting procedures,
such as the elimination of gear backlash, may be required before undertaking a calibration.

Efforts have also been made in this research to develop a robust adaptive robot tracking control algorithm that provides higher dynamic performance without requiring the prohibitive computational power of some other adaptive algorithms. The proposed new adaptive control algorithm is stable, relatively simple and easy to use. It bears a very close link to the conventional PD feedback controllers which enables experiences obtained in PID implementations to be exploited in the realisation of the more advanced control algorithm. On-line dynamic control experiments have been conducted to validate the proposed algorithm in a real-world environment and the experimental results show a performance improvement over 40% with regard to the results of a PID control scheme, measured in terms of the root-mean-square ‘average’ error. The experimental results imply that further effective servo dynamic control performance improvement in absolute terms may require higher resolution joint sensors on the Puma manipulator.

8.2 Suggestions for future research

This research work has produced an open architecture robot controller, a robot control platform and an integrated vision-robot system. A number of issues related to sensor-robot interaction, improving robot positioning accuracy, and improving robot dynamic performance have also been discussed. Further possible research areas would include the following:

- Integration with a solid modelling system that can input CAD (Computer Aided Design) information and interact with the stereo vision system,
Chapter 8: Conclusions

- Develop software that can recognise a workpiece by using the 3D information from the vision system and matching it with the corresponding descriptions in a CAD environment,
- Develop task managing database that can automate the operation of the robot cell based on visual or other types of external sensory feedback and the CAD information,
- Integrating the vision-robot system with other types of external sensors to achieve active compliance control,
- Develop GUI (Graphics User Interface) environment for the robot controller,
- Investigate the on-line applicability of any other trajectory generation and alteration schemes that might result in smooth acceleration in joint space for some adaptive control algorithms.
BIBLIOGRAPHY


Khosla, K. K. and Ramos, S., A Comparative Analysis of the Hardware Requirements for the Lagrange-Euler and Newton-Euler Dynamics
Bibliography


PhD. 1994 Nongji Chen
Bibliography


Walters, W. J., Open architecture controller in flexible systems for welding. IEE Colloquium on Advances in Practical Robot Controllers, IEE Colloquium Digest No. 1993/225.

PhD. 1994 Nongji Chen
LIST OF CONTRIBUTIONS


Incremental encoder position measurement components can only record position displacement (more precisely, counting the encoder pulses) after they start to run. To convert this displacement into position information, a known initial position is a necessity. The procedure of establishing the initial position is called a start-up resetting and it only needs to be undertaken once for as long as the electronics in the measurement loop is not interrupted (e.g., power down, disconnection, etc.).

Enabling a start-up resetting to be performed at any arbitrary point over a joint's working range of a robot means that the joint position must be determined through an absolute position sensing sub-system which has the same or even higher accuracy and resolution. This is obviously pointless as there is no need for the incremental encoder sensor sub-system at all, given that the absolute position sensing sub-system can already do the job. Thus the start-up resettings for the majority of incremental encoder position measurement systems are performed at one or several fixed position points.

Performing a start-up resetting at a fixed position point is simple in principle. It requires a point position sensor to be fixed at a known joint start-up resetting position. During a start-up resetting procedure, the robot joint is driven towards the fixed point. When the joint reaches it, the point position sensor sends out a signal to enable the incremental encoder position measurement sub-system to link the encoder counter status (e.g.,
Appendix A: Incremental Encoder Start-up Resetting

clear the counter value to zero or copy it to a storage register array) to the fixed joint position, and the task of start-up resetting is accomplished.

Problems arise at the issue of accuracy and resolution that can be provided by conventional point position sensors. This is particularly true in robot systems, where the joint position requires to be measured accurately, and therefore requires that the accuracy of the outcome of a start-up resetting procedure can match the measurement requirement. In general, many conventional sensors that can be employed to detect the point positions are unable to directly offer the accuracy and resolution required in robot applications. This leads to a situation where many robot systems have a special start-up resetting requirement.

![Diagram](image_url)

**Figure A.1 Start-up resetting signals for rotary incremental encoders**

Combining a conventional point position sensor, such as an optical-switch, with the rotary encoder index signal provides a simple and effective
way of achieving the start-up resetting with a reasonably high resolution for robot applications. This, however, comes at a price, requiring that the rotary incremental encoders must be physically fitted at a correct angle in relation to the actuator shaft so that the index pulse appears within the active signal window of the point position sensor (see figure A.1). The filtered index signal in figure 3.2.3 provides a unique pulse and the resolution is increased to the index width which is usually ranging between 180° to 720° of the electrical angle of encoder channel signals, with the precise value depending on a particular product. Further improvement in resolutions can be achieved through the quadrature technique, but the ambiguity of the filtered multiple quadrature pulses must be resolved.

It is possible to obtain one specific pulse of the filtered quadrature signal and mask off the rest through special electronic designs. This, however, will increase the interface complexity as the number of the filtered pulses can vary from one product to another depending on the index width of the product under question. A simple way to avoid this problem is to restrict the moving direction to only one in a start-up resetting process and use the first filtered quadrature signal pulse as the position signal (see the bottom line in figure 3.2.3). This in effect increases the start-up resetting resolution to the same level as the quadrature measurement. It should be noted that, the safest way that links the encoder counter status with the start-up resetting signal, under the one directional moving and the first pulse scheme, is to use the signal to reset the counter or to latch the counter contents to a storage register array. This is necessary since the timing information is the only information available to differentiate the first filtered pulse from the rest under the scheme.

Many robot systems have adopted dual measurement schemes for joint position sensing. The idea is to use a coarse absolute position sensor,
such as a potentiometer, to complement a fine incremental encoder sensor for joint position measurement. The purpose of coarse absolute position sensing is two fold. Firstly, it gives the absolute position information before a start-up resetting procedure has been undertaken for the encoder subsystem. This enables the robot controllers to perform some coarse operations, such as move the robot away from an obstacle so that a start-up resetting procedure of the whole robot can be undertaken. Secondly, the coarse absolute position sensor is employed to provide the multiple-points for encoder start-up resetting procedures.

![Diagram]

**Figure A.2 Start-up resetting for an encoder via a potentiometer**

Figure A.2 illustrates the relationship between the potentiometer output voltage signal and the start-up resetting signals of the joint configuration shown in figure 3.2.1, Chapter 3. By driving the joint in the direction of increasing joint angle, a start-up resetting point can be reached (assuming the signal used is the first filtered quadrature pulse) if the joint is
Appendix A: Incremental Encoder Start-up Resetting

not near the position of the upper limit before the procedure is started. At the detected start-up resetting position, the coarse position information obtained through the potentiometer indicates which is the detected start-up resetting signal therefore enabling the encoder start-up resetting procedure to be accomplished. It should be noted that, in order to specify one and only one start-up resetting signal, the accuracy of the potentiometer sensing subsystem should be higher than half of the equivalent joint angle between any two start-up resetting signals. For the case under discussion, the value is 3° of joint angle.

UNIVERSITY OF SURREY LIBRARY