UNIVERSITY OF MANITOBA

A FLEXIBLE ROBOT CONTROL SYSTEM AND ITS APPLICATION TO LASER-BASED OBJECT RECOGNITION USING NEURAL NETWORKS

by

RICHARD TYC

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL AND INDUSTRIAL ENGINEERING

UNIVERSITY OF MANITOBA

WINNIPEG, CANADA

ISBN 0-315-98998-X

Name RICHARD TYC

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

MECHANICAL ENGINEERING — SUBJECT TERM

0 5 4 8 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Nom ____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

______________________________ SUJET

[ ][ ][ ][ ] CODE DE SUJET

U·M·I

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS
Architecture 0729
Beaux-arts 0357
Bibliothéconomie 0399
Cinéma 0900
Communication verbale 0459
Communications 0708
Danse 0378
Histoire de l'art 0377
Journalisme 0391
Musique 0413
Sciences de l'information 0723
Théâtre 0465

ÉDUCATION
Généralités 515
Administration 0514
Art 0273
Collèges communautaires 0275
Commerce 0688
Économie domestique 0278
Éducation permanente 0516
Éducation préscolaire 0518
Éducation sanitaire 0680
Enseignement agricole 0517
Enseignement bilingue et multiculturel 0282
Enseignement industriel 0521
Enseignement primaire. 0524
Enseignement professionnel 0747
Enseignement religieux 0527
Enseignement secondaire 0533
Enseignement spécial 0529
Enseignement supérieur 0745
Évaluation 0288
Finances 0277
Formation des enseignants 0530
Histoire de l'éducation 0520
Langues et littérature 0279
Lecture 0535
Mathématiques 0280
Musique 0522
Orientation et consultation 0519
Philosophie de l'éducation 0998
Physique 0523
Programmes d'études et enseignement 0727
Psychologie 0525
Sciences 0714
Sciences sociales 0534
Sociologie de l'éducation 0340
Technologie 0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE
Langues
- Généralités 0679
- Anciennes 0289
- Linguistique 0290
- Modernes 0291

Littérature
- Généralités 0401
- Anciennes 0294
- Comparée 0295
- Mediévale 0297
- Moderne 0298
- Africaine 0316
- Américaine 0591
- Anglaise 0593
- Asiatique 0305
- Canadienne (Anglaise) 0352
- Canadienne (Française) 0355
- Germanique 0311
- Latino-américaine 0312
- Moyen-orientale 0315
- Romane 0313
- Slave et est-européenne 0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE
Philosophie 0422
Religion
- Généralités 0318
- Clergé 0319
- Études bibliques 0321
- Histoire des religions 0320
- Philosophie de la religion 0322

Théologie 0469

SCIENCES SOCIALES
Anthropologie
- Archéologie 0324
- Culturelle 0326
- Physique 0327

Droit 0398
Économie
- Généralités 0501
- Commerce-Affaires 0505
- Économie agricole 0503
- Économie du travail 0510
- Finances 0508
- Histoire 0509
- Théorie 0511

Études américaines 0323
Études canadiennes 0385
Études féministes 0453
Folklore 0358
Géographie 0366
Gérontologie 0351
Gestion des affaires
- Généralités 0310
- Administration 0454
- Banques 0770
- Comptabilité 0272
- Marketing 0338

Histoire
- Histoire générale 0578
- Ancienne 0579
- Médiévale 0581
- Moderne 0582
- Histoire des noirs 0328
- Africaine 0331
- Canadienne 0334
- États-Unis 0337
- Européenne 0335
- Moyen-orientale 0333
- Latino-américaine 0336
- Asie, Australie et Océanie 0332

Histoire des sciences 0585
Loisirs 0814
Planification urbaine et régionale 0999
Science politique
- Généralités 0615
- Administration publique 0617
- Droit et relations internationales 0616

Sociologie
- Généralités 0626
- Aide et bien-àtre social 0630
- Criminologie et établissements pénitentiaires 0627
- Démographie 0938
- Études de l' individu et de la famille 0628
- Études des relations interethniques et des relations raciales 0631
- Structure et développement social 0700
- Théorie et méthodes 0344
- Travail et relations industrielles 0629

Transports 0709
Travail social 0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES
Agriculture
- Généralités 0473
- Agronomie. 0285
- Alimentation et technologie alimentaire 0359
- Culture 0479
- Élevage et alimentation 0475
- Exploitation des péturages 0777
- Pathologie animale 0476
- Pathologie végétale 0480
- Physiologie végétale 0817
- Sylviculture et faune 0478
- Technologie du bois 0746

Biologie
- Généralités 0306
- Anatomie 0287
- Biologie (Statistiques) 0308
- Biologie moléculaire 0307
- Botanique 0309
- Cellule 0379
- Écologie 0329
- Entomologie 0353
- Génétique 0369
- Limnologie 0793
- Microbiologie 0410
- Neurologie 0317
- Océanographie 0416
- Physiologie 0433
- Radiation 0821
- Science vétérinaire 0778
- Zoologie 0472

Biophysique
- Généralités 0786
- Medicale 0760

SCIENCES DE LA TERRE
Biogéochimie 0425
Géochimie 0996
Géodésie 0370
Géographie physique 0368
Géologie 0372
Géophysique 0373
Hydrologie 0388
Minéralogie 0411
Océanographie physique 0415
Paléobotanique 0345
Paléoécologie 0426
Paléontologie 0418
Paléozoologie 0985
Palynologie 0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT
Économie domestique 0386
Sciences de l'environnement 0768
Sciences de la santé
- Généralités 0566
- Administration des hipitaux 0769
- Alimentation et nutrition 0570
- Audiologie 0300
- Chimiothérapie 0992
- Dentisterie 0567
- Développement humain 0758
- Enseignement 0350
- Immunologie 0982
- Loisirs 0575
- Médecine du travail et thérapie 0354
- Médecine et chirurgie 0564
- Obstétrique et gynécologie 0380
- Ophtalmologie 0381
- Orthophonie 0460
- Pathologie 0571
- Pharmacie 0572
- Pharmacologie 0419
- Physiothérapie 0382
- Radiologie 0574
- Santé mentale 0347
- Santé publique 0573
- Soins infirmiers 0569
- Toxicologie 0383

SCIENCES PHYSIQUES
Sciences Pures
Chimie
- Genéralités 0485
- Biochimie 487
- Chimie agricole 0749
- Chimie analytique 0486
- Chimie minérale 0488
- Chimie nucléaire 0738
- Chimie organique 0490
- Chimie pharmaceutique 0491
- Physique 0494
- PolymÇres 0495
- Radiation 0754

Mathématiques 0405
Physique
- Généralités 0605
- Acoustique 0986
- Astronomie et astrophysique 0606
- Electronique et électricité 0607
- Fluides et plasma 0759
- Météorologie 0608
- Optique 0752
- Particules (Physique nucléaire) 0798
- Physique atomique 0748
- Physique de l'état solide 0611
- Physique moléculaire 0609
- Physique nucléoire 0610
- Radiation 0756

Statistiques 0463

Sciences Appliqués Et Technologie
Informatique 0984
Ingénierie
- Généralités 0537
- Agricole 0539
- Automobile 0540
- Biomédicale 0541
- Chaleur et ther modynamique 0348
- Conditionnement (Emballage) 0549
- Génie aérospatial 0538
- Génie chimique 0542
- Génie civil 0543
- Génie électronique et électrique 0544
- Génie industriel 0546
- Génie mécanique 0548
- Génie nucléaire 0552
- Ingénierie des systämes 0790
- Mécanique navale 0547
- Métallurgie 0743
- Science des matériaux 0794
- Technique du pétrole 0765
- Technique minière 0551
- Techniques sanitaires et municipales 0554
- Technologie hydraulique 0545

Mécanique appliquée 0346
Géotechnologie 0428
Matières plastiques (Technologie) 0795
Recherche opérationnelle 0796
Textiles et tissus (Technologie) 0794

PSYCHOLOGIE
Généralités 0621
Personnalité 0625
Psychobiologie 0349
Psychologie clinique 0622
Psychologie du comportement 0384
Psychologie du développement 0620
Psychologie expérimentale 0623
Psychologie industrielle 0624
Psychologie physiologique 0989
Psychologie sociale 0451
Psychométrie 0632

A FLEXIBLE ROBOT CONTROL SYSTEM AND ITS

APPLICATION TO LASER-BASED OBJECT RECOGNITION

USING NEURAL NETWORKS

BY

RICHARD TYC

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ACKNOWLEDGMENTS

I wish to thank my advisors, Drs. S. Balakrishnan and N. Popplewell, who, through their teaching, guidance, countless discussions and genuine dedication to their fields, were instrumental in the completion of this thesis. I especially thank Dr. Popplewell, whose encouragement and support through both my undergraduate and graduate programs has been a major influence in my professional career. I would also like to thank Mr. Ken Tarte for his technical assistance in the manufacturing and automation lab.

I must also recognize and thank Vansco Electronics Ltd. of Winnipeg, for the gracious donation of both hardware and software for this work. In particular, I must thank Mr. Ed Van Humbeck, whose commitment to research and education in addition to his visionary business insight made the practical implementation of the robot controller possible.

Thanks must also be extended to Mr. Craig Judt, for his invaluable assistance during the preparation and original conception of this project. Through his efforts, many of the ideas presented in this thesis may soon find practical implementations.

Funding, provided in the form of a postgraduate scholarship by the Natural Sciences and Engineering Research Council, is gratefully acknowledged.

Finally, I wish to thank my parents for their support and enthusiasm towards my scholarly endeavors. Many years ago, they made the biggest decision of their lives which, ultimately, provided me with this opportunity and for this, I am also grateful. And last, but certainly not least, I must thank the most important person in my life, my fiancée, Marci, for her love, patience and unending support throughout my graduate program.

ABSTRACT

An artificial neural network based object recognition system was developed for robotic applications and tested. This system was designed to recognize and then determine an object's position and orientation within a robot's working envelope by using a gripper mounted, laser range finder. A flexible, microprocessor based, robot control system, also developed as part of this thesis, made this implementation possible. A single IBM PC compatible microcomputer was used and it required no additional processing capabilities. The control of a 5 degree-of-freedom (DOF) articulated robot (CRS model SRS-M1A) was replaced by a real-time, control system by using a multi-tasking operating system, QNX 4.1. The new control procedure provided effective multi-axis control through a modular software design.

Two separate, neural networks were required to first identify the object and, subsequently, to measure the object's angular position or orientation. The recognition involved extracting edge detection information over specific regions of the object and calculating a unique position and rotation invariant feature pattern, which could be classified by the first network. This procedure required a high resolution, line scan image of the object. Object position was found by calculating the centroid of the image whereas the orientation was found by formulating a pattern of edge detection time histories. A second neural network provided a non-linear transformation of the resulting pattern to form an output which was related directly to the object's orientation.

By using a relatively low image resolution (98 laser scans), object recognition successes of at least 99.9% were achieved with relatively few training patterns and teaching cycles. The methodology, however, assumes that the object is known a priori to be one of several different shapes through training of the first 'recognition' network. The second network was capable of measuring an object's angular rotation to within 2-9°. Position accuracy was of the order of 1-5 mm in trial tests. This level of performance is well suited to many practical applications requiring intelligent robot motion control in unstructured environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	$1/\tau_s$	s^{-1}
$\left[A_i^{i-1}\right]$	Denavit -Hartenburg homogeneous transformation matrix	-
A	Object image area	mm^2
AF	Acceleration factor	-
a_i	Link length	mm
a_i	Activation of node i in previous layer	-
A_i	Object image partial area represented by $drd\theta$	mm^2
$bias_j$	Threshold or bias of unit j	-
c	Index for each input pattern	-
d_i	Distance between links	mm
$D(r)$	Radial integration function	mm
$D(s)$	Equivalent Digital Controller in s-domain	-
$D(z)$	Digital Controller Transfer Function	-
E	Error function	-
e_i	Cartesian error at end effector	mm
$e_{\max}$	Maximum end effector cartesian error	mm
$e(n)$	n^{th} position error	$counts$
e_{ss}	Following error	mm
$f(x,y,z)$	3-D image function	-
$G_1(z)$	ZOH-$G(s)$ Combination Transfer Function	-
$G_{lg}(s)$	Open Loop Forward Path Transfer Function	-
$G(s)$	Transfer Function of Servo Drive	-
h	Path transition factor	-
i	Joint Number	-
i	Coordinate Frame	-

K	Overall Gain	*counts/DAC cnts*
K_A	DC Gain of Amplifier	*V/V*
K_{DAC}	Digital to Analog Conversion Gain	*V/DAC cnts*
K_{em}	Motor Voltage Constant	$V \cdot s/rad$
K_e	Rotary Encoder Gain	*counts/rad*
K_{ff}	Feed Forward Gain	*DAC cnts s/rad*
K_g	Tachometer Voltage Constant	$V \cdot s/rad$
K_m	Total Gain of Motor and Amplifier	$rad/s \cdot V$
K_t	Total Gain of Plant	
n	sample number	-
N	Total number of units in previous MLP network layer	-
N_c	Number of input patterns	-
N_j	Number of output units	-
$\{\vec{n}\},\{\vec{s}\},\{\vec{a}\}$	End effector orientation vectors	-
$O(n)$	n^{th} output command	*DAC cnts*
$\{\vec{p}\}$	End effector position vector	-
$[\vec{P}]$	Cartesian point in robot workspace	-
$q(t)$	Generalized end effector position over time	*mm*
$[R_{(\phi,\theta,\psi)}]$	Euler angle transformation matrix	-
r	Radius of object image function	*mm*
R_i	Distance to end effector from current joint location	*mm*
s	Laplace Operator	s^{-1}
s	Sweep number	-
$s(r,z)$	Feature value function	-
T	Sample period	s
$[T]$	Overall forward transformation matrix	-
t_1 - t_5	Orientation scan edge detection times	s
t_{acc}	Path transition acceleration factor	s
$t_{j,c}$	Target value of output unit j for pattern c	-
x_i	Perceptron input vector	-

x_i, y_i	Location of the centroid of A_i	*mm*
$\{\bar{x}, \bar{y}\}$	Object centroid	*mm*
z	Z-Transform Operator	-
$z_{j,c}$	Activation of output unit *j* for pattern *c*	-
ZOH	Zero Order Hold Function	-
α	Momentum factor	-
α_i	Link twist angle	*rad*
ϕ	Euler angle about robot OZ axis	*rad*
η	Constant learning rate	-
u	Perceptron output value	-
θ	Perceptron bias value	-
θ	Euler angle about robot OV'	*rad*
$\theta_a(s)$	Actual Motor Position	*rad*
$\theta_a(\mathrm{nT})$	Sampled Actual Position	*counts*
$\theta_d(\mathrm{nT})$	Desired Motor position	*counts*
$\theta_{err,i}$	Individual joint error	*rad*
θ_i	Angle between links	*rad*
$\theta(s)$	Angular position in Laplace domain	*rad*
τ_a	Amplifier Time Constant	*s*
τ_e	Electrical Time Constant	*s*
τ_m	Mechanical Time Constant	*s*
ν, ω	Strength of coupling (weights) between layers	-
w_{ji}	Strength of coupling between current layer node (*j*) and previous layer node (*i*)	-
ψ	Euler angle about robot OW"	*rad*
ζ	Damping ratio	-

CHAPTER 1

INTRODUCTION

1.1 Overview

The next generation of industrial robots will very likely benefit from adaptive motion control, based on feedback from various external sensors. The implementation of any form of real-time adaptive control will require a flexible and open robot control system. Robots are generally used in very structured environments requiring accurate fixturing of objects being handled. However, many environments exist today which will benefit from intelligence given to a robot in order to allow it to properly choose a suitable path and complete its task. When uncertainties exist in the robotic workcell, for example disturbances to the part pick up and placement site, peripheral sensory information may be used correctly to measure the actual positions . For example, vision is used extensively today in robotic welding in order to provide the on-line path adjustment necessary for accurately placing weld beads. The type of sensor employed will depend upon the application and accuracy required. Sensing may involve vision systems, laser, ultrasonic and infrared range finders, as well as force and pressure transducers. Regardless of the type of sensor, information must be communicated effectively and interpreted by the control system before progress can be made. A flexible, open architecture within the robot controller must exist to implement this type of control.

Non-contact sensing systems provide one form of sensory feedback which allows a robot to operate in unstructured environments. Video based, vision systems have been used successfully in applications such as semiconductor assembly and welding, but they suffer from disadvantages such as cost, computational complexity, hardware requirements, and deficiencies introduced by the optics (e.g. an inability to accommodate luminescent or transparent objects and problems associated with inadequate background lighting).

The task of interpreting sensory data and providing the ability not only to identify but also to determine an object's location within the scan field has proved to be extremely complicated and the subject of much research [1,2,3]. Although classical methods of scene interpretation or *pattern recognition* provide satisfactory performance in certain applications, a truly practical industrial vision system has yet to be developed. A visual sensor, based on a one dimensional laser range finder, can be utilized as a means of investigating an alternative to vision based methods for identifying objects within a robot's work space and subsequently to measure each object's position and orientation. Unlike a video based system, the laser sensor essentially retrieves a 'line scan image' of an object with high range resolution along the laser's path. Therefore, the structure and acquisition of scene data does not lend itself well to classical methods of image processing because these methods are based normally on video-based, pixel images of the robot's work space. A new technique is needed to deal with the highly non-linear image data retrieved by the laser sensor which precludes the use of classical, pixel based methods. The method must be adaptive, in that a simple means of introducing new parts should be available, and it should be independent of the object's complexity in terms of the object's overall shape or internal features.

Artificial neural networks (ANN) are well suited to pattern recognition problems involving highly non-linear yet unique image data. This is mainly because of their potential for solving very complicated, non-linear transformations as a result of their massively parallel structure of interconnected, non-linear systems [4]. ANNs have generated much interest recently, especially in the field of robotics and control. Their high adaptivity and high fault and noise tolerances, combined with their ability to *learn* the characteristics of the robot-sensor environment, make them a powerful tool in realizing the concept of an intelligent work cell. Their learning capability, much like that of the human biological system, offers a powerful means of improving past performance or interpreting an entirely new set of inputs. Their recent success in many different applications has paved the way for new applications in robotics and automation tasks. Neural networks will be used here to process and interpret scene data acquired by the laser range finder as a first step to developing a simple object recognition system for an industrial robot.

1.2 Objectives

The first objective was to extend the capabilities of a three axis, servo-interface card developed as part of an earlier study [6,7] to control a 5 degree-of-freedom (DOF) robot manipulator. The control system was to be microcomputer based and have a flexible, open software design. The servo motors and original control modules of a 5 DOF CRS articulated manipulator, donated by a local industry (Vansco Electronics Ltd., Winnipeg) were replaced with new servo drive systems. This unique, flexible robot control system allows simple integration of peripheral control and data acquisition devices as well as on-line neural network access . Unfortunately, commercial robots do not provide access to their control software. Adaptively changing the robot's path, for example, by employing previously unknown parameters can be all but impossible with most commercial systems. Such systems are intended, for the most part, to be used for specific, well defined tasks commonly found in today's automated production lines.

The second objective was to provide object recognition and measurement capabilities by using a relatively simple and inexpensive laser range finder attached to the end effector of the robot. In this study, artificial neural networks (ANN) were used to interpret the signal (or pattern of signals) from the range finder and identify randomly located objects within the robot's work area. ANNs were also used to determine the orientation of the object in order to accurately grasp and manipulate it. This task required on-line, adaptive control of the robot because the robot must operate in an unknown environment and its trajectory should be modified according to the object's location.

Artificial neural networks are ideally suited to pattern recognition tasks and their effectiveness in analyzing complicated problems of this nature was the driving force behind this first application of the flexible robot control system. Their application to object recognition, by using a relatively simple sensor, will be emphasized. The system must be capable of learning in order to handle new objects introduced into the robotic cell. Presently, training is done off-line on more powerful workstations in order to save time although on-line training is possible. The robot's control system must be capable of accessing the neural network algorithms for on-line recognition and orientation measurement of the parts. The back propagation neural network [8] is employed by using three layer networks of varying sizes.

1.3 Organization of the Thesis

The design of the robot control system is presented in Chapter 2. The laser and robot integrated experimental set up is presented in Chapter 3. Chapter 4 introduces ANN models, including a brief discussion of the back propagation neural network used in this study. This is followed in Chapter 5 by a description of the object recognition system. Experimental results and general discussion are presented in Chapter 6.

CHAPTER 2

DESIGN OF A FLEXIBLE ROBOT CONTROL SYSTEM

2.1 Introduction

Commercial robot systems are used typically in industrial applications where only limited task space sensory information is needed [9,10]. Such control systems allow the user to define a sequence of motions and program the robot to follow a desired path. Most commercial systems combine an operating system and a programming language with a proprietary hardware configuration. This combination is sufficient for most manufacturing environments. However, more advanced control is required in a robotic research environment aimed at improving the flexibility of the controller. Most importantly, the ability to include sophisticated peripheral sensors is needed for real time modification of the control parameters of the robot. Other important features, such as an off-line programming capability, easily accessible and modifiable control software, and the ability to easily upgrade the hardware for future improvements, are all factors leading to the requirement for a flexible, robot control system.

Many researchers have pursued this approach for a variety of reasons. Goldenberg and Chan [9] have replaced the VAL II controller found on the PUMA 560 robot with a multi-processor based controller, operating under a UNIX environment. Such a system can accommodate new sensory systems, robot programming languages and dynamic models for research and evaluation of advanced control methods. Kabuka, Glaskowsky and Miranda [11] have developed a micro-controller based, robot arm using sophisticated, high performance, digital signal processors (DSP) with the aim of utilizing the complete dynamic model of the manipulator. Most commercial control systems assume that the joints of the robot are decoupled and neglect such terms as the centrifugal and Coriolis forces. High speed robot operation with varying payloads is a major limitation to such an assumption.

Whitcomb and Koditschek [12] in the robotics laboratory at Yale University have used a message passing environment to design a real-time, distributed robot controller intended to standardize all real-time computations. Such a system, which is similar in nature to the one presented in this thesis, uses the Communicating Sequential Process (CSP) model for interprocess communication. This control system approach is based on a powerful microprocessor (μP), the Inmos T800, whose capacity is expanded easily.

With recent advances in the performance of PC based μPs, such as the 32 bit Intel 386 and 486 μP, PCs now offer an economical and flexible alternative to commercial systems that employ proprietary, custom designed hardware. The abundance of software, peripheral devices and operating systems available for PC based platforms make it an ideal candidate for controlling many processes in a flexible manufacturing cell. The robot control system presented here combines the flexibility and expandability of the Intel based architecture with a real-time, multi-tasking operating system. This thesis extends previous work on μP based motion control in computer numerical control (CNC) applications [5,6,7] to the robotic environment. Concerns over floating point computational speed, real-time control of five axes and high sampling rates required a more efficient operating system than the previously used MS-DOS based platform. The entire system has been developed for portability and flexibility by using the 'C' programming language .

2.2 Structure of a Robot Controller

The structure of a modern robot control system contains low level and high level controls [9]. The low level control consists of servo drive motors (attached to robot joints), servo amplifiers, as well as position and velocity feedback sensors. The high level control includes the microcomputer (μC), terminal, shared memory, disk drives and peripheral device cards. A schematic of the overall control architecture is shown in Figure 2.1.

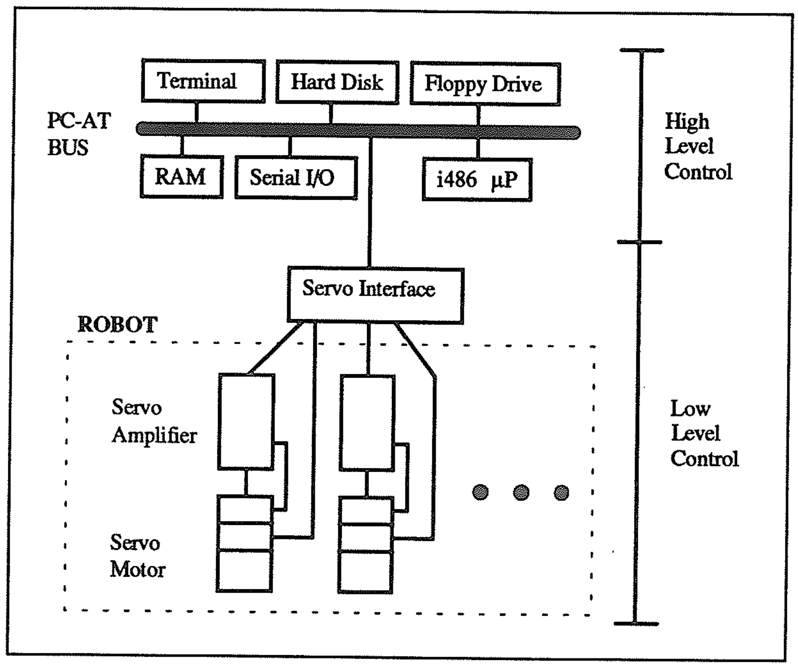


Figure 2.1. PC based control structure

The primary function of the control system is to accurately control the position of each joint of the robot in order to follow a desired path. Several feedback control loops are used to effectively accomplish this task. The inner-most loop of the servo-drive system is the velocity feedback loop between the amplifier and servo-motor. The velocity signal is generated by a tachometer mounted on each servo motor. This feedback acts to stabilize the servo system and provides a fast response to a reference speed command. The position feedback loop between the servo and the interface card is controlled by the main μP and usually consists of an incremental rotary encoder interfaced to an integrated chip (IC) which decodes the signal into a n-bit counter value. This process will be detailed in Chapter 3.

An example of a commercial robot system, the PUMA 560 (from Unimation Inc.) which is used commonly in industry and research environments, is shown in Figure 2.2. This system incorporates the DEC LSI-11/02 microcomputer. All peripheral devices are connected through the LSI bus and the robot's control software is loaded into EPROM memory. Each individual joint axis is controlled by a dedicated μP (Motorola 6503) which communicates directly through the arm interface board to the DRV-11-J parallel interface board on the LSI bus. The major limitations of this system, however, are [9] :

- lack of floating point hardware,
- limited capability for trajectory modification through external sensors,
- software loaded into EPROM memory cannot be examined or modified, and
- a non-standard, inflexible hardware configuration makes customizing or future upgrades expensive and complicated.

The proposed architecture, which is shown in Figure 2.1, offers a more standard and flexible hardware configuration that is used widely and is far more economical. The main differences lie in the use of the PC-AT bus (together with the Intel 486 μP) and the motion control interface card which currently does not have any processing capability. The single, high performance processing unit, the Intel 486 DX 33 μP, also houses a floating point co-processor and it is responsible for controlling the entire system including such tasks as:

- terminal I/O,
- floppy and hard disk access,
- memory management,
- discrete sampling of all 5 robot joint axes,
- motor command output through the interface card, and
- peripheral device communication such as digital I/O interface, data acquisition hardware, network access, etc.

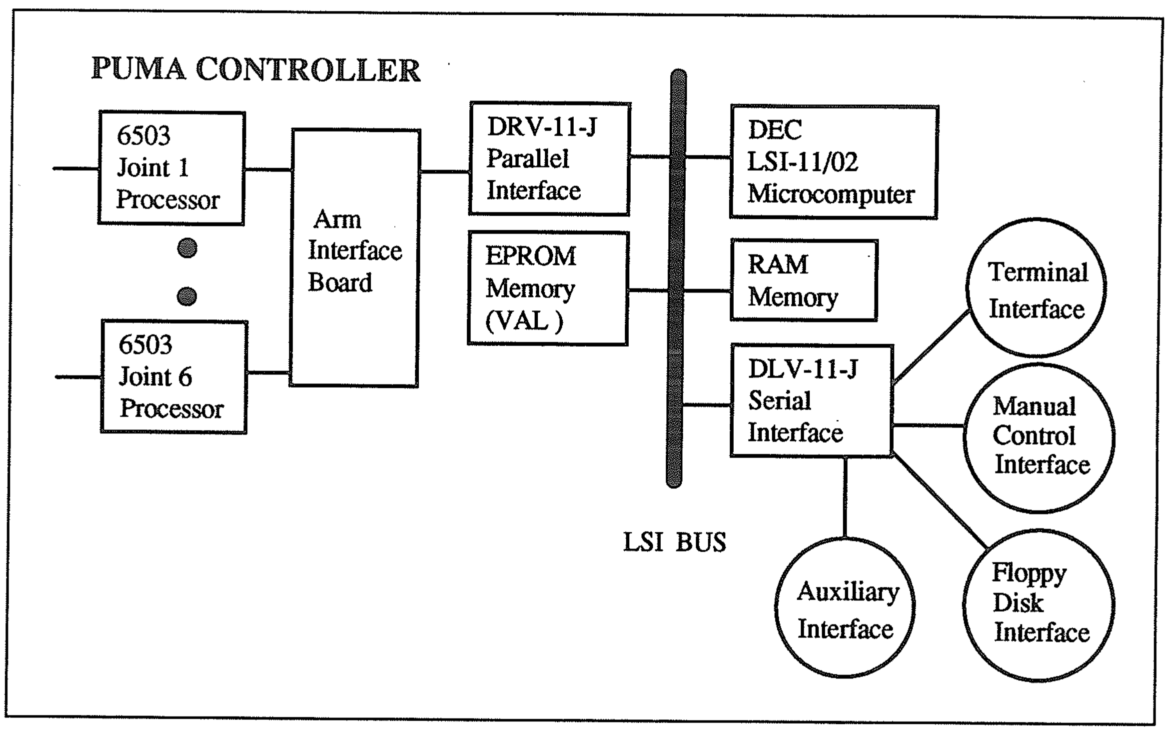


Figure 2.2. PUMA 560 control system

The following sections introduce the theoretical model for the sampled data control system. This theory is based on the assumption that the motor drive at each robot joint can be modeled as a linear system. If the motor and amplifier are sized correctly, this assumption is valid and suitable for a simplified control system analysis. A brief analysis of the servo drive system is presented to predict and study the performance of the overall position control system and to derive the control parameters. More importantly, a study of the digital controller and the effects of sampling on system dynamics require careful attention. Once the position control system has been presented, the kinematics of a 5 DOF manipulator are examined and homogenous transformations are developed.

2.3 Position Control

The position control system, a subset of the overall robot controller, is responsible for accurately positioning each servo motor to guide the end effector along a desired path. The position feedback loop illustrated in Figure 2.3 provides such a control but it is enhanced by other feedback elements (the velocity and current feedbacks) and feed forward compensation. In a sampled data system, the µP is responsible for sampling the position feedback information and then providing an analog control signal to the amplifier to correct errors. Incremental rotary encoders are used to provide a digital feedback pulse stream to the microcomputer. To improve the following error, without affecting the control system's parameters , feed forward compensation has also been added to the position control system.

The servo drive system consists of three control loops, position, velocity and motor current feedback - as shown in Figure 2.3. A standard commercial servo motor-amplifier system includes velocity and current feedback compensation within the analog circuitry on the power amplifier. Position compensation is controlled, however, by the microcomputer.

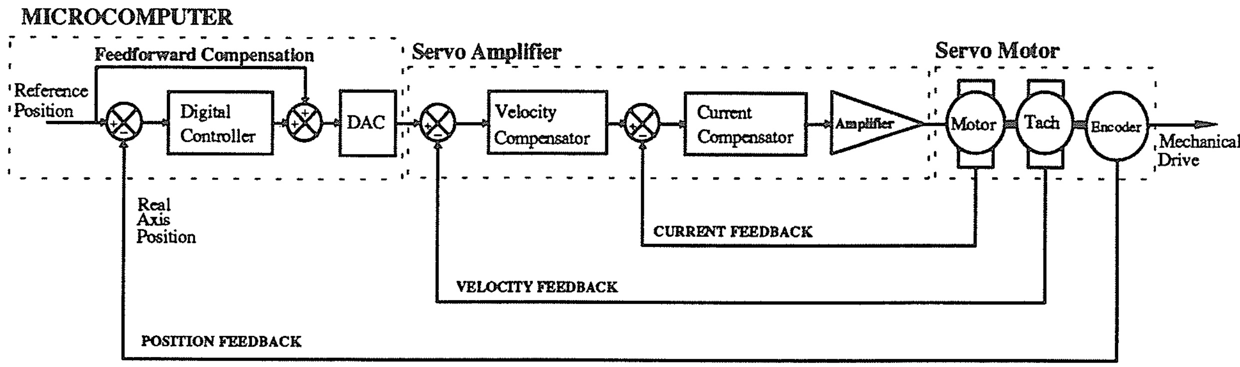
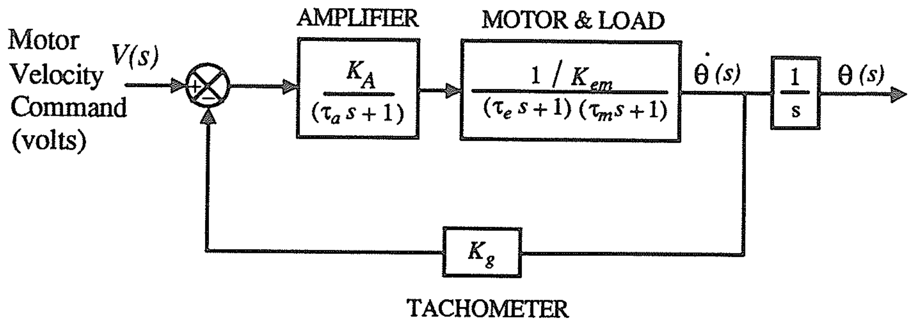


Figure 2.3. Position Control System

As mentioned earlier, the work reported in this thesis utilized an available CRS (model SRS-M1A) 5 DOF, articulated robot. The existing servo drive system (see Section 3.2.2) was replaced with new servo motors that included velocity feedback for improved performance. The velocity feedback improved the stability of the overall position control system. The decision was also influenced by the relatively slow sampling frequency of the motion control interface card [6,7] which is described later in Section 3.2.4 of Chapter 3. The effect of velocity compensation was found by studying the closed loop transfer function of the amplifier-motor combination with the velocity feedback shown in Figure 2.4.



where K_A = DC Gain of Amplifier $\quad K_g$ = Tachometer's Voltage Constant
τ_m = Mechanical Time Constant $\quad K_{em}$ = Motor's Voltage Constant
τ_e = Electrical Time Constant $\quad \tau_a$ = Amplifier's Time Constant

Figure 2.4. Block Diagram of Amplifier - Motor System

Figure 2.4 presents a simplified model of the amplifier-motor combination. Further simplification is possible because of the mismatch between the mechanical and electrical characteristics. The electrical (or armature) time constant of the motor $(\tau_e = L/R)$ is much smaller than the mechanical time constant, τ_m, and can normally be neglected if $\tau_m > 10\tau_e$ [13]. The amplifier's bandwidth is much greater than that of the closed loop system. As a result, the term $(\tau_a s+1)$ can also be neglected. This greatly simplifies the motor transfer function for the selected drive system. From the block diagram of Figure 2.4, the transfer function, *G(s)*, can be found, by using standard control theory [14], to be

$$G(s) = \frac{\theta(s)}{V(s)} = \frac{K_A/K_{em}}{s\left[(\tau_m s+1)+\frac{K_A K_g}{K_{em}}\right]} = \frac{K_A/(K_{em}\tau_m)}{s^2+\frac{(1+K_A K_g/K_{em})}{\tau_m}s} \cdot \tag{2.1}$$

This simplified but reasonable model of the transfer function produces the characteristic equation [14]:

$$s^2 + \frac{\left(1 + K_A K_g / K_{em}\right)}{\tau_m} s = 0 \quad . \tag{2.2}$$

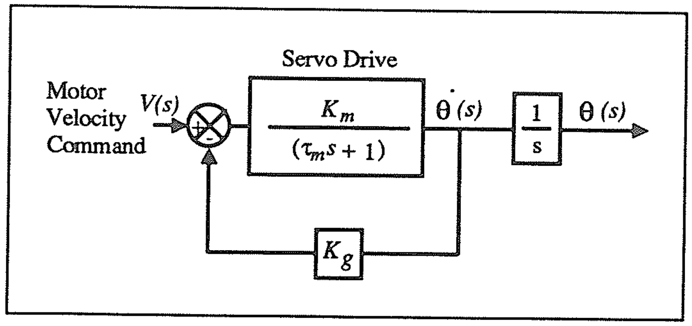
If no velocity feedback is used, the characteristic equation becomes:

$$s^2 + \frac{1}{\tau_m} s = 0 \quad . \tag{2.3}$$

The tachometer constant, K_g, increases the system's damping by a factor $\frac{K_A K_g}{K_{em} \tau_m}$. The addition of velocity feedback provides a *minor loop feedback* to the overall position control system and effectively increases the damping ratio. The importance of velocity feedback has been recognized by a number of motor manufacturers who have made available servo motors with integrally mounted tachometers or resolvers [13,14].

The inner-most, current feedback loop shown in Figure 2.3 limits the motor's armature current and reduces the effect of torque disturbances on the servo system. Both velocity and current feedback are implemented in analog circuitry housed within the servo amplifier. Potentiometers are used to obtain a stable velocity response to a step input and to limit the motor's maximum armature current.

The simplified, first order model of the servo drive, shown in Figure 2.5, is used for further analysis. It contains a single pole and describes the velocity output of the motor as a linear function of the input command voltage. The model is sufficiently accurate, providing linearity conditions are preserved. Excessive load torques or high accelerations outside the servo's capability would render the linear model assumption invalid and would require a more complicated, non-linear model. This is beyond the scope of this thesis.



K_m = Total Gain of Motor and Amplifier $(rad/s \cdot V)$

Figure 2.5. First order model of servo drive used for position control analysis

From Figure 2.5, the overall transfer function can be found to be:

$$G(s) = \frac{K_m/\tau_m}{s\left(s + \frac{1+K_mK_g}{\tau_m}\right)} = \frac{K_m/(1+K_mK_g)}{s\left(\frac{\tau_m}{1+K_mK_g}s+1\right)} = \frac{K_{Tot}}{s(\tau_s s+1)}, \tag{2.4}$$

where $\tau_s = \frac{\tau_m}{(1+K_mK_g)}$ = Loaded Servo Drive Mechanical Time Constant (s^{-1})

and $K_{Tot} = \frac{K_m}{(1+K_mK_g)}$ = Overall Servo Drive Gain $(rad/s \cdot V)$.

The objective of the position control system is to control the angular position of the servo motor's shaft, $\theta(s)$. The position controller is implemented in digital form so that an appropriate transformation of all system components is required to the z-plane. The block diagram of the resulting digital position control system is shown in Figure 2.6, where,

K_{DAC} = Digital to Analog Conversion Gain $(V/DAC\ counts)$,
K_e = Rotary Encoder Gain $(counts/rad)$,
K_{ff} = Feed Forward Gain,
(s) = Laplace Operator (s^{-1}),
$z = Z$ - transform operator,
T = Sample period (s),
$\theta_a(s)$ = Motor's Actual Position (rad),
$\theta_a(\mathrm{nT})$ = Actual Position $(counts)$,
$\theta_d(\mathrm{nT})$ = Motor's Desired Position $(counts)$,
$G(s)$ = Transfer Function of Servo Drive $= K_{Tot}/s(\tau_s s+1)$, and
ZOH = Zero Order Hold Function.

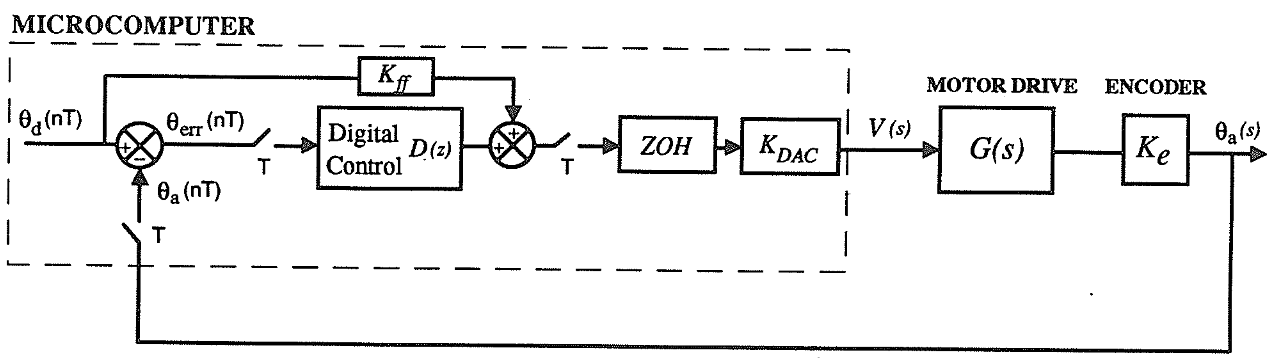


Figure 2.6. Model of position control system

2.3.1 Analysis of Position Control

For the simplified servo model shown in Figure 2.6, it is necessary to select an appropriate digital controller (or *compensator*) so that the system remains stable and provides good accuracy. The gains K_{DAC}, K_{Tot}, and K_e are selected through hardware while K_{ff} is adjusted through software. (This aspect is discussed further in Section 2.3.4.)

The feed forward term is not considered here because it lies outside the main block and does not affect the closed loop response.

In order to find the overall transfer function, $\dfrac{\theta(z)}{R(z)}$, the block diagram of Figure 2.6 must be transformed completely into the digital *z*-domain. This transformation can be performed in the two steps shown in Figures 2.7(a) and (b). The first reduction transforms the microcomputer portion of Figure 2.6 into a more simplified form, *D(z)* and *ZOH*, and the motor drive-encoder combination into *G(s)*, which represents the plant (shown in Figure 2.7(a)). The form shown in Figure 2.7(a) still lies in both the digital (controller and *ZOH* blocks) and continuous time domains (plant). A further reduction is made to the form shown in Figure 2.7(b) which includes only the digital controller and the plant, both represented in the *z*-domain. The transfer function $G_1(z)$ is found from standard tables [14] based on the form of the plant's transfer function, *G(s)*. In this case, *G(s)* has the form $K/s(s+a)$ so that

$$G_1(z) = \frac{K}{a}\frac{\left[\mathrm{T} + \frac{1}{a}e^{-a\mathrm{T}} - \frac{1}{a}\right]z + \left[\frac{1}{a} - \frac{1}{a}e^{-a\mathrm{T}} - \mathrm{T}e^{-a\mathrm{T}}\right]}{(z-1)\left(z-e^{-a\mathrm{T}}\right)} \tag{2.5}$$

where: $K = K_{DAC} \cdot K_{Tot} \cdot K_e/\tau_s$ and $a = 1/\tau_s$.

The digital controller, *D(z)*, can be found by selecting a compensator, *D(s)*, in the continuous system and approximating it with *D(z)*. The selection of *D(z)* can be verified by studying the system's stability and response for the desired sampling frequency. Proportional -Derivative (PD) compensation, discussed further in Section 2.3.2, was used to provide additional stability through damping corresponding to the derivative gain, K_d. An integral gain was not needed because the amplifier contains integral compensation so that problems such as integral "wind-up" are avoided. The following transfer functions represent PD compensation [8]:

$$D(s) = K_p + K_d s \tag{2.6}$$

and

$$D(z) = K_p + K_d \frac{z-1}{Tz} . \tag{2.7}$$

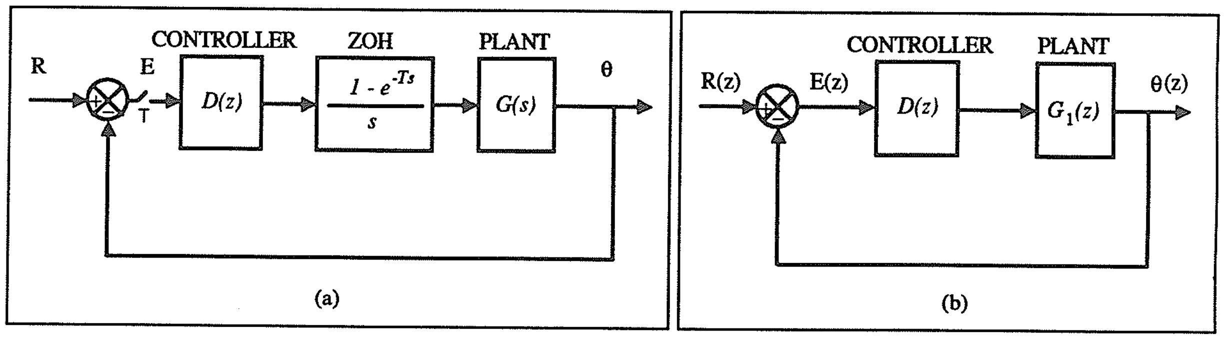


Figure 2.7. Block diagram reduction for the digital control system

The closed loop transfer function of the system shown in Figure 2.7(b) is,

$$\frac{\theta(z)}{R(z)} = \frac{G_1(z)D(z)}{1+G_1(z)D(z)} . \tag{2.8}$$

From the digital form of the closed loop transfer function (2.8) which contains $G_1(z)$ and *D(z)* (also see equations (2.5) and (2.7)) , it is evident that the closed loop response and stability depends not only on the compensator gains, K_p and K_d, but, more importantly, on the sampling time, T. In fact, sampling can render an otherwise stable, continuously controlled system unstable in the *z*-domain [14]. Before proceeding further, the PD compensator gains must be found.

2.3.2 PID Compensation

PID (Proportional plus Integral plus Derivative) control is the most commonly used form of dynamic compensation [13,14]. The proportional term increases the total gain of the servo system, thereby improving the response time and reducing the following error. However, increasing the proportional term alone also causes instability and possibly introduces oscillations in the servo drive. The derivative term is included to improve stability, to increase the system's damping, and to allow the proportional term to be increased. This approach can improve system accuracy because a higher proportional gain is used without loss of system damping and stability. The integral term reduces the steady state errors by providing a signal which is proportional to the time integral of the error. However, integral gain can lead to a final position overshoot due to "integral windup" caused by a large accumulation of errors during long movements. This accumulation tends to dominate the performance of the controller, effectively diminishing the properties of the proportional and derivative gains. Integral gain is implemented directly through analog circuitry on the servo amplifier and it is adjusted manually. Such a gain provides "stiffness" to load torque disturbances and reduces the steady state, velocity error. To avoid the effects of "integral wind-up", the integral term has been omitted in the digital implementation of the PID controller.

A Proportional-Derivative (PD) controller is implemented in software to provide a command signal to the amplifier (through the DAC) as a function of the error signal. The PD compensator possesses lead and lag filters which allow a high DC gain and extend the bandwidth of the system to make it more responsive.

The digital control system's compensator, $D(z)$, is found by selecting an equivalent, continuous system design, in this case a PD type, and approximating it in the digital domain. Equations (2.5) and (2.6) represent these two compensators. The designer must choose appropriate values of the PD gains to satisfy the performance requirements. The gains are found by studying the system in the s-domain because the servo drive is an analog system. (However, the method used applies directly to the z-plane as well.) Using equation (2.6) for $D(s)$, the open loop, forward path, transfer function or loop gain function of the continuous system is given as:

$$G_{lg}(s) = D(s)\frac{K_{DAC} \cdot K_{Tot} \cdot K_e}{s(1+\tau_s s)} = \frac{K_p K_t \left(1 + {}^{K_d}\!/\!_{K_p}\, s\right)}{s(1+\tau_s s)} \tag{2.9}$$

where: K_t = Total Gain of Plant $= K_{DAC} \cdot K_{Tot} \cdot K_e$.

The s^2 term in the denominator of equation (2.9) implies a second order, type 1 system which results in a zero steady state error for a step input $\left({}^{1}\!/\!_{s}\right)$ and a fixed steady state error for a ramp input $\left({}^{1}\!/\!_{s^2}\right)$, which is also known as a *following error* [8]. This error, unlike those in machine tool applications, can be tolerated in robot control but should be kept to a minimum in order to provide an accurately controlled path.

The following error is defined as,

$$e_{ss} = \frac{1}{K_p K_t} \quad . \tag{2.10}$$

Furthermore, if the ratio K_d/K_p in equation (2.9) is chosen equal to the time constant, τ_s, a pole-zero cancellation occurs, thereby reducing the order of the system and improving the steady state performance. The system then becomes first order with an exponential response and critical damping. The pole-zero cancellation, chosen in the s-plane, will result in a similar cancellation in the z-plane. This method to limit e_{ss} and provide pole-zero cancellation is used to find K_p and K_d.

In order to specify a reasonable following error limit, e_{ss}, the geometrical configuration of the robot must be examined. Normally, accuracy and repeatability are defined in a Cartesian space for robotic applications. The CRS robot used in this work provided a repeatability of ± 0.13 mm with its original control system [15]. The current analysis is confined to individual joints, so that this specification must be transformed to the joint space.

The following error can be interpreted as a path deviation in the Cartesian space. In comparison to the requirements of CNC controllers in metal removal applications [6,7], high tolerances throughout a path are not needed in robotic applications. Rather, a high accuracy is required at the end point in order to precisely position the robot end effector. For a type 1 system, a zero steady state error to a step input satisfies this condition. Therefore, the following error tolerance can be relaxed for smoother, continuous motion between path points. The original specification of ±0.13 mm is used as a guideline, for further analysis.

The worst scenario to consider in specifying the joint space error, relative to Cartesian error, is when the robot links are extended fully, as shown in Figure 2.8. Individual joint errors result in a maximum end point deviation with the largest errors caused by joints 1 and 2.

The Cartesian error at the end point can be expressed as :

$$e_i = 2R_i \sin\left(\frac{\theta_{err,i}}{2}\right) \tag{2.11}$$

where e_i = Cartesian error (mm),

$\theta_{err,i}$ = angular joint error (rad),

R_i = distance to end point from current joint location (mm), and

i = joint number.

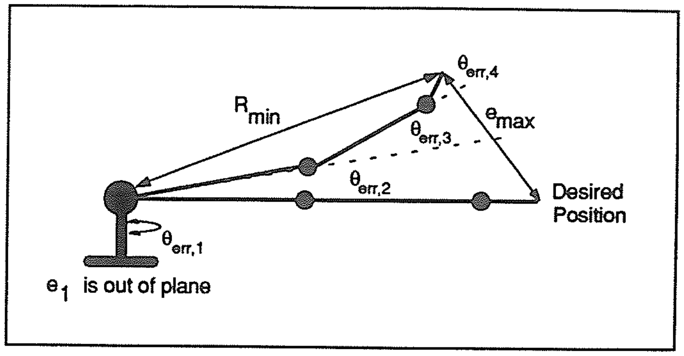


Figure 2.8. End point deviation due to individual joint error, $\theta_{err,i}$

For the CRS robot,

$R_1 = R_2 = 660.4\ mm,\quad R_3 = 406.4\ mm,$ and $\quad R_4 = 152.4\ mm.$

Joint 5, the end effector *roll* axis, contributes zero translational displacement errors and only minor orientation errors so that it is not considered further.

If each joint is in the configuration shown in Figure 2.8, the maximum Cartesian error can be found, by using the cosine law, to be:

$$e_{\max} = \sqrt{R_1^{\,2} + R_{\min}^2 - 2R_1R_{\min}\cos\left(\theta_{err,2} + \theta_{err,3} + \theta_{err,4}\right) + e_1^2}\ . \tag{2.12}$$

Setting $e_{\max}$ equal to 0.13 mm, the maximum allowed joint errors can be found to be

$$\theta_{err}(\max) = 9.4 \times 10^{-5}\ \text{rad}. \tag{2.13}$$

To find e_{ss}, the maximum joint error must be referenced back to the servo motor's shaft through the gear reducer. The CRS robot has a 72:1 harmonic reduction drive in joints 1, 2 and 3. Moreover, a 16:1 gear train is employed for joints 4 and 5. Therefore, the maximum allowable error at the servo motor's shaft is,

$$e_{ss} = \theta_{err}(\max) \times 72 = 0.006768 \text{ rad} \qquad \text{(for Joints 1,2 \& 3)} \tag{2.14}$$

$$e_{ss} = \theta_{err}(\max) \times 16 = 0.001504 \text{ rad} \qquad \text{(for Joints 4 \& 5)}\,. \tag{2.15}$$

Using these values for e_{ss}, equation (2.10) can be solved by substituting the following empirically found values :

$$K_{DAC} = \frac{10}{127} \frac{\text{V}}{\text{DAC counts}} \qquad = 0.07874 \text{ V/DAC counts},$$

$$K_{Tot} = \frac{2500}{10} \frac{\text{RPM}}{\text{V}} \cdot 2\pi \frac{\text{rad}}{\text{rev}} \cdot \frac{1}{60} \frac{\text{min}}{\text{s}} = 26.18 \frac{\text{rad}}{s \cdot V}, \qquad \text{(Joints 1,2 \& 3)}$$

$$K_{Tot} = \frac{2000}{10} \frac{\text{RPM}}{\text{V}} \cdot 2\pi \frac{\text{rad}}{\text{rev}} \cdot \frac{1}{60} \frac{\text{min}}{\text{s}} = 20.94 \frac{\text{rad}}{s \cdot V}, \qquad \text{(Joints 4 \& 5)}$$

$$K_e = 4000 \frac{\text{counts}}{\text{rev}} \cdot \frac{1}{2\pi} \frac{\text{rev}}{\text{rev}} \qquad = 636.62 \frac{\text{counts}}{\text{rad}},$$

and, $\tau_s = 0.014$ s.

These values of K_{DAC}, K_{Tot}, K_e, and τ_s, are found through measurement of actual hardware parameters. A more detailed derivation can be found in references [5,6] outlining previous work with similar servo components. The DAC gain, K_{DAC},, and encoder gain, K_e, are fixed constants which are determined from hardware specifications. An 8-bit DAC is used which provided a ± 10 V analog signal and 1000 line encoders (4000 quadrature counts) are used on all axes. The total motor-amplifier gain, K_{Tot}, is adjusted through analog potentiometers located within the amplifier. The drive system's mechanical time constant, τ_s, is defined as the time required for a loaded motor to respond to 63.2% of a step velocity command [14]. This was measured, on-line, for all axes and then averaged arithmetically. In fact, the values recorded were almost constant because each axis employed identical servo motors. The present version of the motion control interface (MCI) card is hardwired to sample at 250 Hz (i.e. T = 0.004 s).

Now, for joints 1,2,3:

$K_p K_t = 1/e_{ss} = 147.75, \quad K_p = 147.75/K_t$, and

$K_t = 0.07874 \times 26.18 \times 636.62 = 1312.34$ (counts/DAC counts$\cdot$s).

$$\therefore \quad K_p = 0.1126 \text{ DAC counts / rad.} \tag{2.16}$$

Solving for K_d,

$$K_d = K_p \tau_s = 0.1126 \times .014 = 0.00158 \text{ DAC counts} \cdot \text{s / rad.} \tag{2.17}$$

For joints 4 and 5:

$K_p K_t = 1/e_{ss} = 664.89, \quad K_p = 664.89/K_t$,

$K_t = 0.07874 \times 20.94 \times 636.62 = 1049.67$ (counts/DAC counts$\cdot$s),

$$K_p = 0.6334 \text{ DAC counts / rad, and} \tag{2.18}$$

$$K_d = K_p \tau_s = 0.6334 \times .014 = 0.00887 \text{ DAC counts} \cdot \text{s / rad.} \tag{2.19}$$

The PD compensator was implemented in software by using the simple difference equation:

$$O(n) = K_p \cdot e(n) + \frac{K_d \cdot (e(n) - e(n-1))}{\mathrm{T}} \tag{2.20}$$

where:

$O(n) = \mathrm{n^{th}}$ output (DAC counts),

$n =$ sample number,

$e(n) = \mathrm{n^{th}}$ position error (rad), and

$\mathrm{T} =$ sampling period (s).

This concludes the derivation of the transfer function parameters for the digital, closed loop, feedback control system. A digital analysis can be performed now to asses the effect of sampling when using the current hardware parameters. This type of analysis, previously assumed unnecessary for the machine tool controller [6], is needed to determine if the current sampling rate is satisfactory for robotic control.

2.3.3 Digital Analysis

Now that the PD compensator gains have been determined, the system's transfer function, given by equation (2.8), can be found as:

$$G_1(z) = \frac{0.68331z + 0.6213}{(z-1)(z-0.7515)}, \tag{2.21}$$

and

$$D(z) = K_p + K_d \frac{z-1}{\mathrm{T}z} = \frac{(K_p\mathrm{T} + K_d)z - K_d}{\mathrm{T}z}. \tag{2.22}$$

From these last two equations, the characteristic equation of the closed loop transfer function (equation (2.8)) can be found to be:

$$1 + G_1(z)D(z) = 0 \tag{2.23}$$

or

$$\begin{gathered}\mathrm{T}z^3 + \left[0.68331\left(\mathrm{T}K_p + K_d\right) - 0.7515\mathrm{T} - \mathrm{T}\right]z^2 + \\ \left[0.7515\mathrm{T} - 0.68331K_d + 0.6213\left(\mathrm{T}K_p + K_d\right)\right]z - 0.6213K_d = 0\end{gathered} \tag{2.24}$$

so that

$$z^3 - 1.40463z^2 + 0.79693z - 0.24540 = 0 \qquad \text{for} \quad \text{(joints 1,2 and 3)}, \tag{2.25}$$

and

$$z^3 - 0.19333z^2 + 0.95619z - 1.10193 = 0 \quad \text{for} \quad \text{(joints 4 and 5).} \qquad (2.26)$$

Solving for the roots of equation (2.25) gives

$$z_1 = 0.78869, \quad z_2 = 0.30796 + 0.46508j, \text{ and } \quad z_3 = 0.30796 - 0.46508j$$

where $j = \sqrt{-1}$. The roots of equation (2.26), on the other hand, are

$$z_1 = 0.77967, \quad z_2 = -0.29317 + 1.15212j, \text{ and } \quad z_3 = -0.29317 - 1.15212j \;.$$

For stability in the z-plane, all the poles of the system's transfer function (or roots of the characteristic equation) must lie inside the unit circle. This condition has been satisfied only for joints 1,2 and 3 whereas joints 4 and 5 show borderline stability. This was seen in practice because joints 4 and 5 were difficult to tune in order to achieve the desired following error. Consequently, K_p and K_d were reduced by approximately 10 % (through visual and aural observation) for joints 4 and 5. One important point to note is that the damping ratio, ζ, in the z-plane has been reduced considerably due to sampling. The imaginary roots of equation (2.25) lie within a constant-ζ curve corresponding to ζ = 0.5 [14] - a significant reduction from the originally intended critically damped (ζ = 1.0) system. A low damping ratio ($\zeta < 1$) can cause excessive overshoot and undue oscillatory behavior [14].

The sampling rate, set at 250 Hz, can have a significant effect on the system's performance as shown in this analysis. If the sampling rate is doubled (T = 2 msec), equation (2.24) would have the following roots :

$z_1 = 0.28139, \quad z_2 = 0.54414, \quad z_3 = 0.87990$ giving $\zeta = 1.0$ (Joints 1,2 and 3), and
$z_1 = 0.87576, \quad z_{2,3} = 0.133 \pm 0.8204j$ giving $\zeta \approx 0.15$ (Joints 4 and 5).

If the sampling rate is increased to 1 KHz (T = 1 msec), conversely, then

$z_1 = 0.09130$, $z_2 = 0.93522$, $z_3 = 0.09130$ giving $\zeta = 1.0$ (Joints 1,2 and 3), and

$z_1 = 0.93357$, $z_{2,3} = 0.3248 \pm 0.4839j$ giving $\zeta \approx 0.45$ (Joints 4 and 5).

It can be seen that the sampling rate can dramatically affect the system's damping ratio, particularly for joints 4 and 5, and can even create instability in some cases. The current sampling rate of 250 Hz is adequate although significant improvement can be made if it is increased. During experimental studies, joints 4 and 5 did tend to oscillate in some cases, indicating an under damped response, as shown by the analytical results. They were also the most difficult axes to tune during installation. This difficulty was due mainly to their low gear reduction which required a significantly higher degree of accuracy at the motor's shaft.

It should be noted that the empirical values determined for the PD compensator gains were used as a benchmark during system tuning. The control software allowed on-line modification of gains on each axis so that they were tuned according to their actual response. This was done by trial and error, beginning with very low gains and slowly increasing K_p and K_d until good accuracy and a stable response was achieved. Once the axis was tuned properly, the gains were saved and used for all subsequent motion profiles. This is an important feature of a digitally implemented controller. On the other hand, the tedious process of tuning can be made significantly more efficient by using on-line, real time, adaptive tuning. However, this was not attempted.

2.3.4 Feed forward Compensation

The feed forward gain, K_{ff}, shown in Figures 2.3 and 2.6 generates a velocity command signal proportional to the derivative of the position command (or desired position $\theta_d(t)$). If no change in position command occurs, then the feed forward command is zero. Ideally, a 100% feed forward compensation would provide the exact velocity command to the servo-drive without any need for position error . A more conservative approach (50 - 70 % of the required velocity command) is used normally because the real system, including the mechanical load, is not ideal [16]. If too much feed forward gain is used, the actual position will overshoot the final position commanded.

The feed forward command is open-loop and will not affect the closed loop stability. The feed forward term, however, significantly reduces the following error without the need to increase the proportional gain, K_p [13,14,17]. For this reason, velocity feed forward control is used extensively in commercial position controllers. It has been implemented here, through software, in the micro-controller to improve accuracy and reduce the following error throughout the robot's path.

2.4 Kinematics of a 5 DOF Robot Manipulator

With the position control system modeled and optimized to achieve good accuracy, the kinematics of the manipulator must be developed in order to accurately compute the position reference commands for the servo drive system. A manipulator consists of a series of links connected by joints. A method of describing the position and orientation of the robot with respect to joint angles can be developed for the particular link configuration. Homogenous transformations and kinematic equations are used to represent the position and orientation in a world coordinate frame (Cartesian, polar or spherical) as a function of the joint coordinates.

2.4.1 Homogeneous Transformations

The method of Denavit and Hartenburg [18] is used to describe the transformation of one coordinate system (or *frame*) to another. Referring to Figure 2.9, the Denavit-Hartenburg notation represents each coordinate frame, *i*, at every joint relative to the previous frame, *(i-1)*. The position and orientation of coordinate frame *i*, relative to frame *(i-1)*, is represented by the homogeneous transformation, A_i^{i-1}, given by the following 4 × 4 matrix:

$$A_i^{i-1} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.27}$$

where:

- a_i is the link length,
- α_i is the twist angle,
- d_i is the distance between links, and
- θ_i is the angle between links.

The first three columns of A_i^{i-1} represent the coordinate transformation that specifies the orientation of frame *i* relative to frame *(i-1)*. The fourth column, on the other hand, represents the position of frame *i* relative to frame *(i-1)*.

Figure 2.9 illustrates the coordinate frames for the 5 DOF CRS robot considered. Several intermediate coordinate frames are used at link 5 (5a,5b,5c,5d) to account for the fact that the tool's roll axis (joint 5) is perpendicular to the wrist's pitch axis (joint 4). Intermediate transformations are needed to make z_5 collinear with the axis of revolution.

The Denavit-Hartenburg parameters for the coordinate frames shown in Figure 2.9 are listed in Table 2.1.

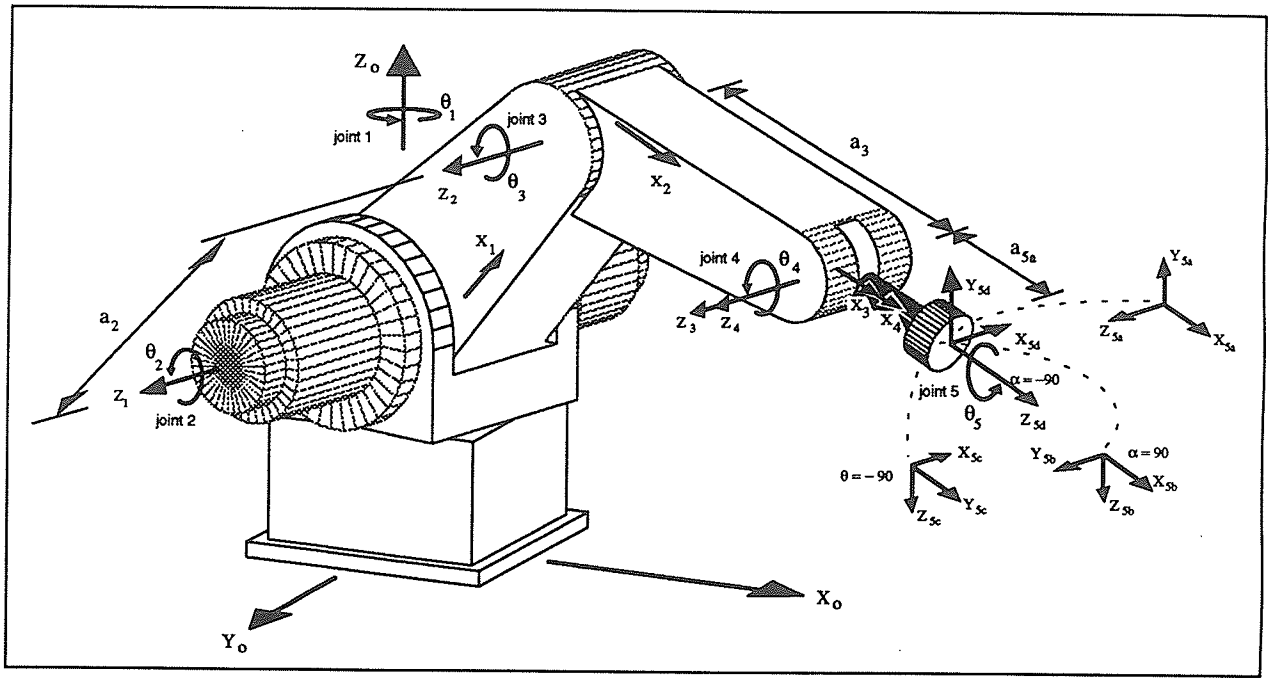


Figure 2.9. Manipulator's coordinate frames

Table 2.1. Denavit-Hartenburg table of link parameters

Frame, i	Variable	α_i (deg)	a_i (mm)	d_i (mm)	θ_i (deg)
1	θ_1	90	0	0	θ_1
2	θ_2	0	$a_2 = 254$	0	θ_2
3	θ_3	0	$a_3 = 254$	0	θ_3
4	θ_4	0	0	0	θ_4
5a	-	0	$a_{5a} = 152.4$	0	$\theta = 0$
5b	-	90	0	0	$\theta = 0$
5c	-	0	0	0	$\theta = -90$
5d	θ_5	-90	0	0	θ_5

The homogeneous transformations associated with all five joints of the CRS robot are determined by substituting the parameters listed in the table into equation (2.27). The homogeneous transformations are given then by:

$$A_1^0 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & c_2 a_2 \\ s_2 & c_2 & 0 & s_2 a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & c_3 a_3 \\ s_3 & c_3 & 0 & s_3 a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$A_5^4 = A_{5a}^4 \cdot A_{5b}^{5a} \cdot A_{5c}^{5b} \cdot A_{5d}^{5c} = \begin{bmatrix} 0 & 0 & 1 & a_5 \\ s_5 & c_5 & 0 & 0 \\ -c_5 & s_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad . \tag{2.28}$$

2.4.2 Kinematic Equations

Having derived the transformation matrices, the overall forward transformation equation or [T] matrix can be computed as:

$$T_5 = A_1^0 \cdot A_2^1 \cdot A_3^2 \cdot A_4^3 \cdot A_5^4 \tag{2.29}$$

or

$$T_5 = \begin{bmatrix} -C_1S_{234}S_5 - S_1C_5 & -C_1S_{234}C_5 + S_1S_5 & C_1C_{234} & a_5C_1C_{234} + a_3C_1C_{23} + a_2C_1C_2 \\ -S_1S_{234}S_5 - C_1C_5 & -S_1S_{234}C_5 - C_1S_5 & S_1C_{234} & a_5S_1C_{234} + a_3S_1C_{23} + a_2S_1C_2 \\ C_{234}S_5 & C_{234}C_5 & S_{234} & a_5S_{234} + a_3S_{23} + a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \quad (2.30)$$

This transformation relates the joint angles with the position and orientation of the robot's end-effector (or simply the fifth coordinate frame $\{X_5, Y_5, Z_5\}$ attached to it). The preceding transformation (2.30), contains the following elements :

$$T_5 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \quad (2.31)$$

The position of the end effector is represented by the $\{p\}$ vector whereas the orientation is represented by the combination of vectors $\{n\}, \{s\}$ and $\{a\}$. The orientation can be expressed in the more commonly used Euler angle description [19]. This beneficially reduces the number of parameters required to describe the orientation from the 9 parameters $(\{n\}_{3\times1}, \{s\}_{3\times1}, \{a\}_{3\times1})$ to the 3 parameters (ϕ, θ, ψ).

Euler angles describe any possible orientation by using a sequence of rotations about the X, Y and Z axes of the moving reference frame. Several different representations exist such as Z-Y-X, Z-Y-Z or ROLL, PITCH and YAW (fixed angle set) - all of which differ only by their sequence of rotations. The Z-Y-Z Euler description [19] is used here. The sequence of rotations shown in Figure 2.10, in which the order <u>is</u> important, is as follows:

- ϕ about the OZ axis,
- θ about the rotated OV' axis, and

- ψ about the rotated OW" axis.

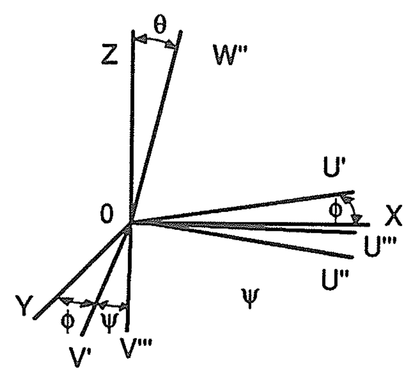


Figure 2.10. Z-Y-Z Euler angle description

If each rotation is represented in a transformation matrix and multiplied together, the following transform is found:

$$R_{(\phi,\theta,\psi)} = \begin{bmatrix} c\phi c\theta c\psi - s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi s\theta \\ -s\theta c\psi & s\theta s\psi & c\theta \end{bmatrix}. \tag{2.32}$$

This transform can be equated to the orientation portion of equation (2.31) and the three Euler angles can be found as a function of $\{\vec{n}\},\{\vec{s}\}$ and $\{\vec{a}\}$. In the case of the robot used in this study, the Euler angles are related directly to the position of the robot's joints 1 and 5 $(\theta_1\ and\ \theta_5)$ in the following manner:

- $\phi = \theta_1 = \mathrm{atan2}(p_y, p_x)$
- $\psi = \theta_5$.

Now, a Cartesian point in the robot's work space requires the following six parameters for a complete description of its location :

$$\bar{P} = \{p_x, p_y, p_z, \phi, \theta, \psi\}^T \quad . \tag{2.33}$$

This format is used to describe the path points of a given robot trajectory. (See section 2.5.)

2.4.3 Inverse Kinematics

The forward transformation matrix, equation (2.30), is useful only if the desired joint angles are known. Normally, the inverse transformation is required for coordinated motion. (See section 2.5.) The desired Cartesian position is usually known and the joint angles must be found from the inverse transform of equation (2.30). The solution is found by employing a combination of algebraic and geometric methods.

From equation (2.29), a rearrangement results in

$$\left[A_1^0\right]^{-1} T_5^0 = A_2^1 A_3^2 A_4^3 A_5^4 \ . \tag{2.34}$$

Multiplying out both sides gives

$$\begin{bmatrix} c_1 n_x + s_1 n_y & c_1 s_x + s_1 s_y & c_1 a_x + s_1 a_y & c_1 p_x + s_1 p_y \\ n_z & s_z & a_z & p_z \\ s_1 n_x - c_1 n_y & s_1 s_x - c_1 s_y & s_1 a_x - c_1 a_y & s_1 p_x - c_1 p_y \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{234} c_5 & -s_{234} s_5 & c_{234} & a_3 c_{23} + a_2 c_2 + a_{5a} c_{234} \\ -c_{234} c_5 & c_{234} s_5 & s_{234} & a_3 s_{23} + a_2 s_2 + a_{5a} s_{234} \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

By equating elements, the following parameters are found:

$$\theta_1 = \operatorname{atan2}(p_y, p_x) \tag{2.35}$$

$$\theta_5 = \operatorname{atan2}(c_1 n_y - s_1 n_x, c_1 s_y - s_1 s_x) \ \text{ or } \ \theta_5 = \psi \tag{2.36}$$

$$\theta_{234} = \operatorname{atan2}(a_z, c_1 a_x + s_1 a_y) \ . \tag{2.37}$$

The solution for θ_2, θ_3 and θ_4 is found geometrically from Figure 2.11.

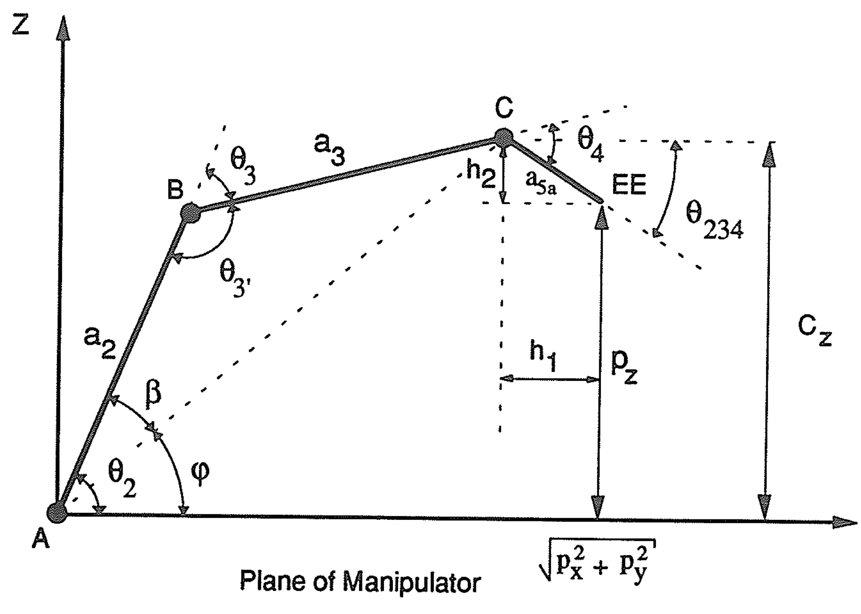


Figure 2.11. Geometrical representation of manipulator

The location of point C can be found by evaluating h_1 and h_2. They are given by

$h_1 = a_{5a} \cos(\theta_{234})$ where $\theta_{234} = \theta_2 + \theta_3 + \theta_4$

and

$h_2 = a_{5a} \sin(\theta_{234})$.

Thus, $C_z = p_z - h_2$ and $C_{Manipulator\ Plane} = \sqrt{p_x^{\ 2} + p_y^{\ 2}} - h_1$.

Triangle ABC of Figure 2.11 is now defined. By using the cosine law to solve for θ_3,

$$\cos\theta_3 = \frac{\left(\sqrt{p_x^{\ 2} + p_y^{\ 2}} - h_1\right)^2 + (p_z - h_2)^2 - \left(a_2^{\ 2} + a_3^{\ 2}\right)}{2a_2 a_3}, \tag{2.38}$$

$\sin\theta_3 = \pm\sqrt{1-\cos^2\theta_3}$, and

$$\theta_3 = \operatorname{atan2}\left(-\sqrt{1-\cos^2\theta_3}, \cos\theta_3\right). \tag{2.39}$$

Solving for θ_2 and θ_4, gives

$$\theta_2 = \varphi + \beta \text{, and} \tag{2.40}$$

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \,. \tag{2.41}$$

In equations (2.40) and (2.41),

$$\varphi = \operatorname{atan2}\left(p_z - h_2, \sqrt{p_x^{\,2} + p_y^{\,2}} - h_1\right),$$

$$\cos\beta = \frac{a_2^{\,2} + \left(\sqrt{p_x^{\,2} + p_y^{\,2}} - h_1\right)^2 + \left(p_z - h_2\right)^2 - a_3^{\,2}}{2a_2\sqrt{\left(\sqrt{p_x^{\,2} + p_y^{\,2}} - h_1\right)^2 + \left(p_z - h_2\right)^2}} \text{, and}$$

$$\beta = \operatorname{atan2}\left(\sqrt{1 - \cos^2\beta}, \cos\beta\right).$$

These derivations complete the inverse kinematic analysis. Equations (2.35) through (2.41) were implemented in software to compute the desired joint angles from a known Cartesian position. Further checks (software *if* statements) were needed for the parameters of equation (2.39) to account for the multiple solutions of $\sin(\theta_3)$. Also, the range of values for the *atan2* function required modification of θ_4 when it approached 180 °. For example, if the required joint position was 181°, the *atan2* function returned -179°. If the previous robot position was +179°, this would result in a 358° angular displacement in one command step.

2.5 Expressions for a Motion Trajectory

The most important function of the robot's control system is to control the motion of the end effector along a desired path. The path or trajectory is described by a start and an end position with intermediate points which define the path to be followed.

Trajectory equations can be expressed in joint coordinates or Cartesian coordinates. Joint coordinated motion is obtained by a linear interpolation of individual joint positions between path node points. The procedure is very efficient and it is limited only by the maximum joint accelerations and joint velocities [18] which the servo motors are capable of reaching. However, the end effector's motion does not follow lines or any type of well defined path between desired positions. Joint coordinated motion is normally used for large motion of parts when the end effector is clear of any obstacles. It cannot be used when the path must be well defined, for example when the robot is near fixtures or other equipment during part pick up and placement. In order to avoid multiple coordinated motion strategies requiring more complicated software, the current robot controller only implements Cartesian Coordinate Motion [7]. This form of motion is natural to Cartesian coordinates (along lines) and can be extended easily to other orthogonal coordinate systems such as cylindrical or spherical coordinates. This type of coordinated motion requires a continuous evaluation of the manipulator's Cartesian position with a subsequent transformation to joint coordinates made necessary to control an individual servo's position. Such a procedure is computationally expensive and can suffer from manipulator degeneracies [18] where joint rates can become infinite. It is also difficult to predict whether excessive joint velocities or accelerations will occur during a trajectory segment. However, the functionally defined motion is crucial for pick-and-place operations and obstacle avoidance.

Having derived the inverse kinematics, the Cartesian positions (including the Euler angles) can be transformed easily into corresponding joint coordinates. This transformation provides the joint command positions for the position control system. Thus, the motion between two points on a path can be described as a linear approximation between the initial and final Cartesian positions. However, a purely linear approximation would cause vibration or jerk at path transition points because of discontinuities in direction. Therefore, a fourth order polynomial is used to describe the transition from one trajectory segment to another [18]. It is given as :

$$q(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{2.42}$$

where $q(t)$ describes the generalized position over time, t (i.e. x, y, z, ϕ, θ or ψ) and $a_0 - a_4$ are constants found by applying the boundary conditions. This equation provides continuity of position, velocity and acceleration throughout the transition and results in a smooth, controlled robot trajectory.

Applying the end conditions, the position, velocity and acceleration during the transition are as follows:

$$q = \left[\left(\Delta C \frac{t_{acc}}{T_1} + \Delta B\right)(2-h)h^2 - 2\Delta B\right]h + B + \Delta B \tag{2.43}$$

$$\dot{q} = \left[\left(\Delta C \frac{t_{acc}}{T_1} + \Delta B\right)(1.5-h)2h^2 - \Delta B\right]\frac{1}{t_{acc}} \tag{2.44}$$

$$\ddot{q} = \left(\Delta C \frac{t_{acc}}{T_1} + \Delta B\right)(1-h)\frac{3h}{t_{acc}{}^2} \tag{2.45}$$

where $h = \dfrac{t + t_{acc}}{2t_{acc}}$.

After the transition at point B in Figure 2.12 ($t = t_{acc}$), the position, velocity and acceleration are given by:

$$q = \Delta C h + B, \quad \dot{q} = \frac{\Delta C}{T_1}, \quad \ddot{q} = 0 \text{ and } h = \frac{t}{T_1}. \tag{2.46}$$

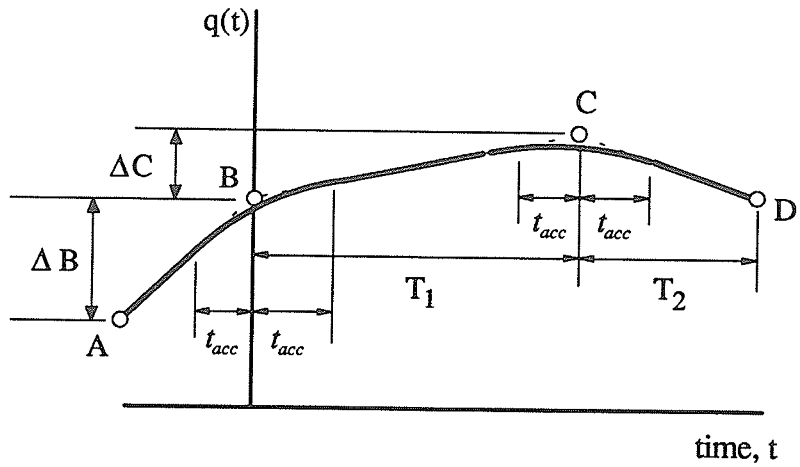


Figure 2.12. Path transitions (adapted from [18])

The transition acceleration instant, t_{acc}, is implemented in software as an acceleration factor, *AF*, where

$$t_{acc} = \frac{AF \cdot T_{segment}}{2} \quad . \tag{2.47}$$

In effect, the acceleration factor controls the shape of the velocity profile between the path points, as shown in Figure 2.13.

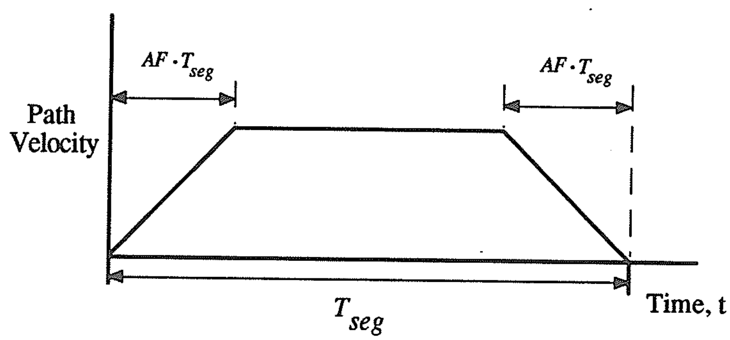


Figure 2.13. Velocity profile through path segments

The manipulator does not pass through intermediate points but, rather, passes nearby, as shown in Figure 2.13. If the acceleration factor is small (3-5 % of the total path time), the end-effector will come closer to the path point but will decelerate towards and accelerate away from the point much more rapidly. This can result in a "jerky" motion and should be avoided. Now the path acceleration is a complicated function of individual servo accelerations so that it is possible to exceed the capability of a servo-motor and cause instability. The only time the manipulator passes through a point is when it stops there. Therefore, if it is desired to pass through intermediate points, it can be done only at a zero velocity. This requirement is accomplished by repeating the trajectory point in the path specification [18].

This concludes the analysis of the position control system and the equations necessary for coordinated motion of the robot. These equations are implemented in the software based robot controller, whose details are presented in Chapter 3.

CHAPTER 3

CONFIGURATION AND PERFORMANCE OF THE ROBOT CONTROLLER

3.1 Introduction

A CRS robot was modified to evaluate the performance of the microcomputer based, robot controller. Only the frame and drive gear trains were retained. The five existing servo motors were replaced by Electro-Craft motors and matching amplifiers. The hardware and software configuration of the controller used for the object recognition experiments will be discussed in this chapter. A brief analysis of the system's performance will conclude this chapter.

3.2 Robot Hardware

3.2.1 The CRS Robot Arm

The CRS robot uses an articulated arm design and contains only revolute joints. The working envelope of the robot is shown in Figure 3.1. Harmonic drive gear reducers are used on joints 1, 2 and 3. Joints 1 and 2 are connected directly and joint 3 is driven by a preloaded drive chain. Joints 4 and 5 use a combination of spur and bevel gears driven by a second series of drive chains. A servo-driven gripper is used as the end effector.

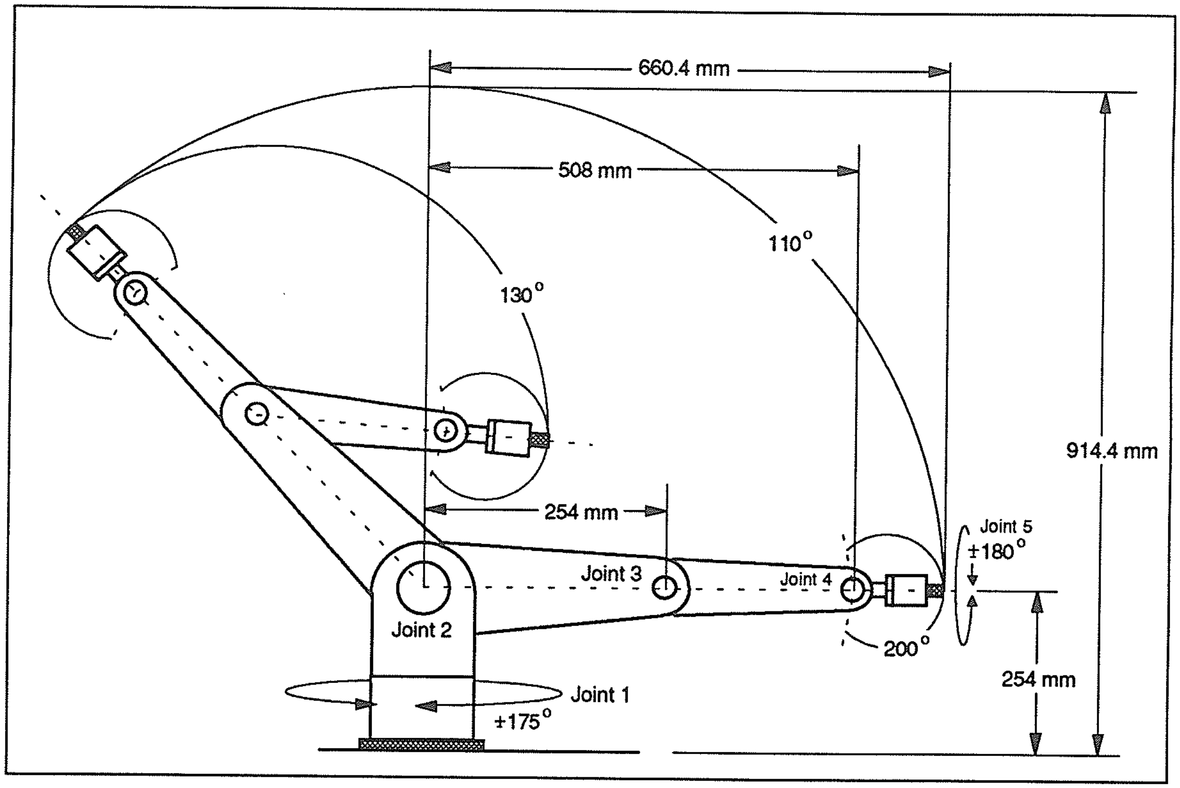


Figure 3.1. Arm configuration of CRS robot

3.2.2 Servo Drive System

A Max-100 PWM servo amplifier and the model E586 DC servo motor, both from Electro-Craft Corporation [13,20], were selected for all 5 axes. Each servo motor has an integral tachometer and optical rotary encoder for velocity and position feedback, respectively. The addition of velocity feedback, not present on the original robot control system, significantly improved the dynamic performance and stability, especially at the relatively low command frequency (250 Hz) used in this study. The flexible interface on the amplifier allows a simple connection to any controller which provides status outputs, directional limit inputs and an external inhibit capability for emergency situations.

3.2.3 Microcomputer

An IBM PC compatible, 486DX-33 microcomputer, also made available for this study by Vansco Electronics, was used for the robot controller. This is a full 32-bit processor with an integral floating point unit operating at 33 MHz. A full tower design was used giving ample space for the interface and I/O cards. The power and affordability of this system made it an ideal choice for a flexible control system. The software based, control system can be configured easily for future hardware upgrades, such as the new Pentium (P5) µC now on the market. This will be made possible by backward compatibility of all Intel products.

3.2.4 Motion Control Interface Card

Two custom developed, motion control interface (MCI) cards [6] are used to monitor and control each axis of the robot. Each card can control up to three axes. In the present configuration, five axes and an additional gripper can be controlled by the two cards. The card is designed to be compatible with the IBM PC/AT bus standard and includes:

- encoder interface,
- 8 bit Digital-to-Analog Converter (DAC-08), as well as
- address and Data bus interface (standard) .

The encoder interface uses the HCTL-2000 motion control IC from Hewlett Packard. It receives optical encoder pulses and performs quadrature decoding. The IC has a built in, 12 bit up/down counter and an output data latch used to transfer the counter value to the data bus for subsequent reading. The control system's software reads the counter value from the data bus and converts it to angular position units at the sampling frequency. The DAC-08 is needed to provide an analog voltage to the servo amplifier corresponding to the speed reference. It provides a linear, analog signal as a function of digital input, in this case an 8 bit quantity. The address and data bus interface provide a standard means of communication between the μC and the interface card. Each HCTL IC and DAC-08 exist as an I/O address on the AT bus allowing simple reads and writes to the appropriate address. The card's base address is preselected through switches and does not interfere with other peripheral devices such as the serial/parallel communication card, video card or disk controllers.

Only one 25 pin connector is available on the card for external connection and a secondary interface box is used to bring all encoder and amplifier connections into one main line. The MCI card design is simple and economical, utilizing less than $ 300.00 of hardware [6]. A new model, currently being developed, will offer improved performance and an even simpler hardware design. It will incorporate a new high performance motion control IC and a single PAL (Programmable Array Logic) IC to coordinate all communication and time sensitive I/O [21]. This technology reduces the number of components on the card and allows easy modification of logic functions.

3.2.5 I/O Expansion Card

A standard digital I/O card is used to interface with reference or home switches mounted on each axis of the robot. These switches are used to initialize the robot by always moving to a preset home position. The digital I/O card also allows the control system to interface with other external devices in the robot cell through standard TTL signals. Figure 3.2 illustrates the general structure of the controller used during the course of this work.

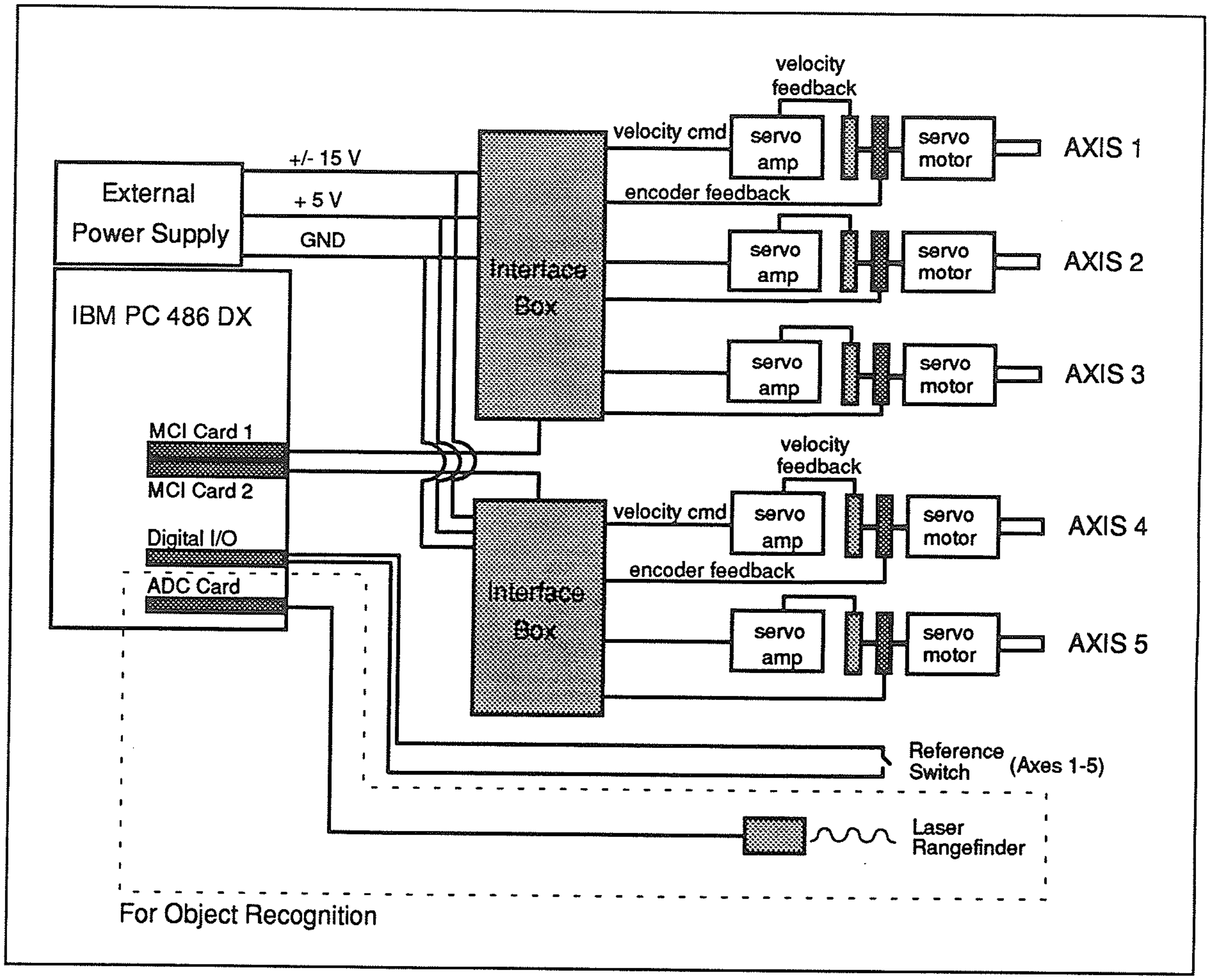


Figure 3.2. General structure of the controller

3.3 Software Design

A unique, software based, control system implements the equations derived in Chapter 2 for individual servo position control and manipulator kinematics in a modular design by using the 'C' programming language. The PC based control structure uses a single CPU to perform all functions of the controller, including servo control for each of the 5 axes. For this reason, a real-time, multi-tasking operating system (OS) was chosen. The multi-tasked OS is capable of providing pre-emptive scheduling (ie. the ability for low priority tasks to relinquish control of the CPU to higher priority tasks when needed) in addition to normal time-slicing operations. The software consists of the following eight separate processes or tasks.

- The front end server (*parent*) process is responsible for all terminal I/O and user interface. It also acts as a communications hub for all client tasks.
- Five individual servo control tasks (*child*) receive position information and command each servo-drive through the interface card.
- A scheduler process (*child*) synchronizes each servo task to follow a pre-defined sampling rate.
- The path control process (*child*) computes the kinematic equations and communicates with each servo task to provide command positions along a specified trajectory.

The most important function of the software based controller is to precisely close the position loop at the desired sampling frequency. The motion control interface card has been hardwired to sample each position encoder at 250 Hz. A joint position is read every 4 msec and a new command position is calculated and sent to the DAC. The servo command for each axis should be sent at the same time for proper control. For this reason, a priority driven OS is needed to ensure that the servo tasks close the position loop synchronously and at precise time intervals.

3.3.1 OS Description

The QNX™ operating system, developed by Quantum Software Systems Ltd. of Kanata, Ontario, was chosen as the underlying OS environment. The fundamental design of this system offers a multi-user, multi-tasking, real-time, distributed, message passing operating system [38]. The QNX OS, version 4.1, extends the capabilities of earlier versions. An important aspect of the design is the software's compatibility with current and proposed IEEE POSIX standards for operating systems. A version of OpenLook™ is also available to provide a graphical interface similar to that of the UNIX based system.

QNX consists of a small kernel, only 8 Kb in size, which is in charge of system related processes such as the network manager, process manager, file system manager and the device manager. This micro kernel supervises all interprocess communication, process scheduling, low level network communication and interrupt handling. The micro kernel's architecture is the key to its real time performance. The Watcom™ C compiler is supported by QNX and has been customized to provide several QNX related functions not found on other platforms.

A fundamental operating principle of QNX, known as message based interprocess communication, provides a means of synchronizing the execution of several processes. This feature allows the programmer to efficiently execute code and utilize CPU time between sampling instants and then synchronize the execution of important servo processes all simultaneously. For example, all five joint command signals must occur simultaneously in order to achieve the desired end effector path. However, the kinematic transformations and mathematical computations may demand different amounts of CPU resources for each axis depending on the type of path desired and the complexity of each joint transformation. In QNX, each servo control task can be considered as a separate process requiring a certain CPU time. When the transformations are complete and new joint positions are computed, the task becomes *blocked* and waits to receive a timed signal (in this case a *proxy*) from the scheduler task. The scheduler task is *receive blocked* during the 4 msec delay and then triggers a *proxy* owned by each servo task. As processes send, receive and reply to messages, each undergo various changes of state that affect when and how long they run. When a task is *blocked*, it relinquishes CPU control to all other system processes. By giving the scheduler and servo tasks highest priority, CPU time will always be available for their real time execution. This provides the fundamental software design concept on which the motion control program is built upon. The overall structure is shown in Figure 3.3.

QNX allows the creation of several real-time timers which can be used, in connection with the message passing system, to synchronize the servo output commands with the sampling rate of the motion control cards. The control algorithm must compute the desired position based on this sampling period and must command the motion control cards no faster than the sampling rate allows. Now several computations must occur between each sampling instant so that the scheduler process must be delayed the correct period before commanding the cards. It is critical, therefore, that the total execution time never exceed the 4 msec limit set by the sampling frequency.

Other advantages of the QNX based system include process monitoring in real time which allows external events to be monitored and interruptions to be created, if necessary. The PC based system can be used to control not only the robot but other automation related functions within the robot's environment. The multi-user capabilities can put the microcomputer on a system-wide network communicating with other tasks and processes. In fact, networking in QNX is completely transparent to the user and can provide additional CPU resources for demanding applications, effectively creating a "pseudo" multi-processor system.

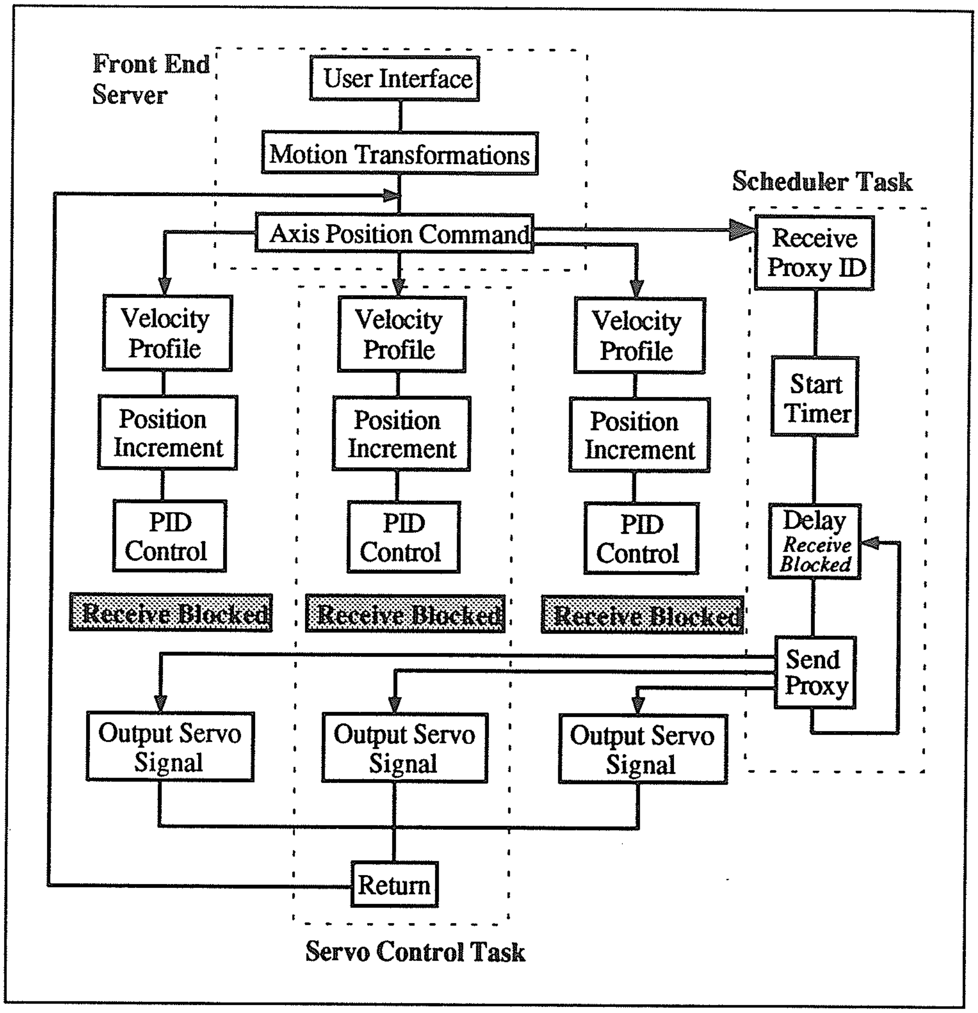


Figure 3.3. Multi-tasked software position control for three axes

3.3.2 Control System Tasks

The control system tasks, described in Section 3.3, form the basis for the robot's control system. Each task is a separate, executable module that is loaded by the main server. Additional tasks can be included easily to provide a variety of control or computational functions. For example, the object recognition system outlined in Chapter 5 runs concurrently with the robot controller. This type of modular design is extremely flexible and it is limited only by hardware deficiencies (i.e. CPU power, memory etc.).

A brief description of each task of the robot controller follows.

3.3.2.1 Front End Server

This is the parent process which begins or *spawns* all other system tasks that run concurrently as *child* processes. This task performs all screen I/O using QNX terminal functions and sets up the menu based, control panel. It performs all hardware and software initializations and communicates with all the *child* processes such as the servo tasks and path control task.

The main menu provides access to all the control system functions. A brief overview of the menu functions is shown in Figure 3.4.

3.3.2.2 Servo Task

Five separate servo control tasks receive position command information from the Server (Jog and Teach modes) or Path Control Task (Path mode) and communicate with the servo drive system to accurately position each joint. When the robot is idle, these tasks continue sampling the position to hold the current joint configuration by closing the position loop. At each sampling instant, a check is made for any pending messages from the Server, indicating that a new control mode should be initiated such as path control, jog mode or a hardware reset.

If a Cartesian path is to be followed, the servo task simply receives the joint position commands from the Path Control Task, reads the current position, performs PID compensation and sends a command signal to the servo drive, as described in Figure 2.1. For individual jog moves, the Server simply passes the desired final angular position and jog velocity. The servo task then commands the servo drive system to complete the jog move in a way that conforms to a trapezoidal velocity profile.

HOME	Sends all joints to a home or reference position.
RESET	Resets all position and counter variables to zero including the encoder pulse counter on the interface card.
DISPLAY	
SERVO PARAM	Dynamically displays current and total encoder counts .
GAINS	Displays PID control gains and feed forward gain, K_{ff}, for each axis. Allows on-line modification through simple keystrokes.
AXES JOG	Displays current axes jog distances used as defaults in *jog mode*.
GRIPPER	Controls gripper related functions (OPEN/CLOSE).
PATH	
FILENAME	Allows user to select desired path file.
GO	Executes path file.
DEMO	Performs pre-defined demonstration programs.
TEST NN	Executes desired path and tests Neural Network program to identify object during test path or measure part position or orientation.
TRAIN NN	Performs Neural Network training cycles using desired robot path.
JOG	
SELECT AXIS	Individual axis selection for jogging.
JOG DIST	Define desired jog distance.
VELOCITY	Define joint angular velocity during jog move.
ALL	Jog all axes simultaneously.
TEACH	Enter teach mode where robot can be commanded to move incrementally along Cartesian directions under keyboard control. Desired path points can be saved in a path file for subsequent execution or modification.

Figure 3.4. Description of main menu functions of control program

Other functions include performing a hardware and software reset, when signaled by the Front End Server, communicating with the I/O interface card during a home move, and storing joint positions in globally accessible RAM for subsequent analysis.

3.3.2.3 Path Control Task

This process is used to generate position command signals in order to follow a desired Cartesian trajectory. The kinematic equations (given in Section 2.4) and motion trajectory expressions (presented in Section 2.5) are used to perform all the required calculations. The Front End Server passes the desired path file and current joint positions to the Path Control Task to begin the process. The computed command positions are sent to each servo task at the sampling frequency until the end effector has reached the final path point.

3.3.2.4 Scheduler Task

The scheduler task is responsible for synchronizing all servo tasks so that command signals to each servo drive occur at the sampling frequency. As soon as the Front End Server is loaded, the scheduler and servo tasks are *spawned* (loaded as child processes) and begin execution at the highest priority. Real time synchronization is done by a software timer which notifies the scheduler with a *proxy event* at a predetermined interval. The scheduler task enters an endless loop where it receives the timer proxy and then triggers five *servo proxies* attached to each servo task.

The scheduler task is predominantly in a *blocked* state waiting for the timer proxy which occurs every 4 ms. The servo tasks are also in a blocked state because they wait for the proxy trigger from the scheduler task for most of their execution time . This allows other less important system processes (including the Front End Server) to continue with their execution. As proxy events can accumulate in the micro kernel, it is important that no task falls behind, or takes up too much CPU time entering the *receive blocked state*. This would be apparent immediately because it would result in all other lower priority tasks becoming "hung" or stop being executed. During the path following mode, the Path

Control Task, Servo Tasks and Scheduler all exist at the same priority and, as a result, normal time slicing occurs in a round robin fashion. Pre-emptive scheduling allows other processes to be executed at a lower priority with no effect on the position control system.

3.4 System Performance

To assess the performance of the robot's control system, the following error was measured during the execution of a Cartesian path. A test trajectory was designed that involved high speed line paths with abrupt changes in velocity and direction. The orientation of the gripper remained constant throughout the path. Figure 3.6 shows the result of the test. Figures 3.5 and 3.6 illustrate test results showing the following error for each axis during a standard jog move, which involves a trapezoidal velocity profile.

It is important to note that the actual position measured during the path was sampled directly from the rotary encoders. A transformation was performed on the joint data to calculate the Cartesian position. The desired position was equivalent to the command position sent by the path control task. The rotary encoder data does not reflect the true end-effector position because the encoder was attached to the servo-motor and not to the actual joint itself. In some joints, the mechanical transmission of the motor output suffered from large angular errors due to backlash in the gear reduction mechanism. Therefore, while the analysis may show good accuracy on the part of the control system, the end effector may have followed a much more inaccurate path during the test. This, in fact, was observed. Visible inaccuracies due to vibration were seen at the end effector. This problem is a result of the specific mechanical design employed, especially in joints 4 and 5.

The following error limit, established during the analysis of the position control system, was not realized in actual tests, as can be seen in Figure 3.7. The combination of each joint error during a Cartesian path contributed to the overall maximum measured error of about 0.83 mm in actual tests. This error was mainly due to a reduction in PD compensation gains from the originally specified values derived in section 2.3.2. Those values found for gains K_p and K_d in equations (2.16) through (2.19) were target values used during the tuning process, which essentially involved a trial and error process of slowly incrementing each parameter until the best possible accuracy was achieved without oscillation or "ringing". This would indicate a stable system. In addition, load inertias were not included in the analysis of section 2.3 because they were not available from the robot manufacturer. This would contribute to the apparent overestimation of the actual position control system's accuracy found by the digital analysis.

Finally, electrical interference or "noise" may be another cause of poor performance. The velocity command signal provided by the MCI card to the servo amplifiers contained considerable high frequency and high amplitude noise when measured by using an oscilloscope. The source of the noise was the PWM amplifiers themselves, a result of their high switching frequency. More sophisticated noise suppression techniques may be used to alleviate this problem.

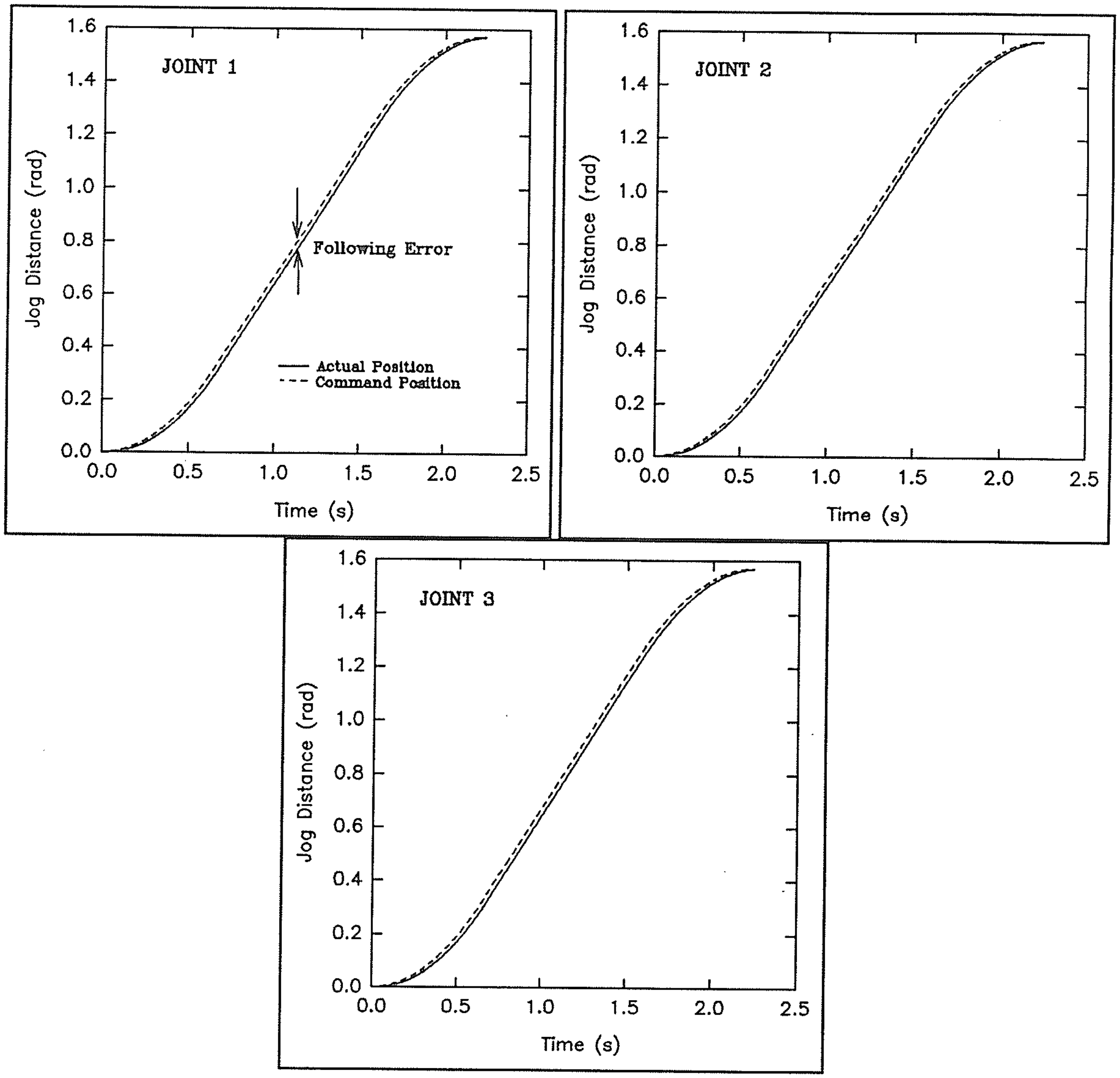


Figure 3.5. Following error in joints 1, 2 and 3 during a standard jog move using a trapezoidal velocity profile

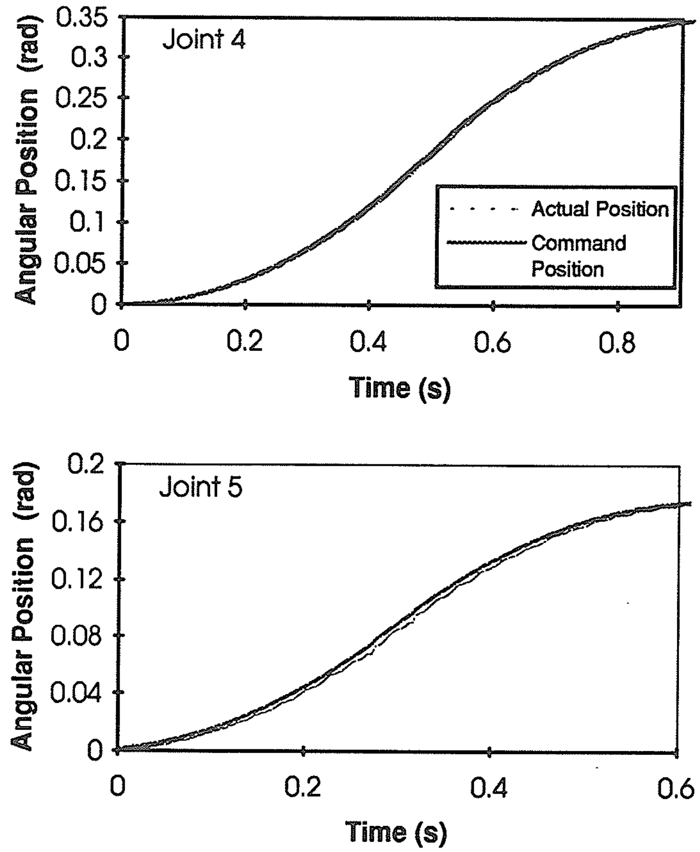


Figure 3.6. Following error in joints 4 and 5

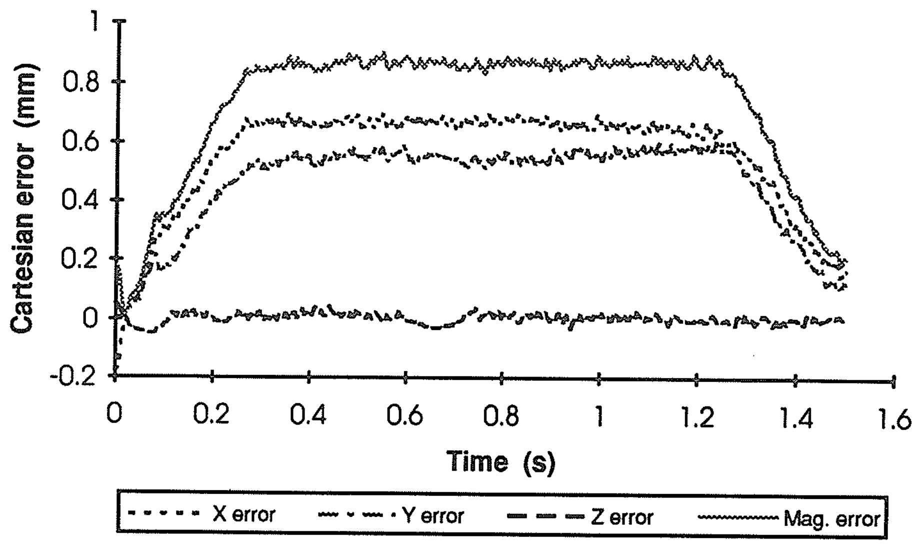


Figure 3.7. Following error throughout the test path

This concludes the description and analysis of the robot control system. The control system's architecture made the implementation of on-line, neural network based, object recognition (to be presented in the next chapter), possible. A brief overview of artificial neural networks and the laser-based object recognition system will be presented in Chapter 4.

CHAPTER 4

NEURAL NETWORKS AND LASER-BASED OBJECT RECOGNITION

4.1 Introduction

The application of robotic systems to tasks involving unstructured or unknown environments requires the use of sophisticated sensors for task related feedback. Object recognition or, more broadly, pattern recognition attempts to provide machines with a powerful, "human-like" sensory feedback allowing them to interact intelligently with the environment. Effective object recognition, accompanied by accurate determination of the object's orientation and position, can provide more efficient and flexible material handling abilities. An application would be in robotic assembly where intelligent work cells must recognize the arrival of workpieces in order to initiate scheduled operations. Perhaps the greatest use of machine vision for object recognition is in the area of printed circuit board inspection [22]. The recent trend towards batch type processing, in which engineered products are manufactured or assembled in batch sizes of fifty or less, results in the need for expensive jig and fixture development to present workpieces at precise locations for the robot to handle properly. Furthermore, the push towards automating assembly requires intelligent sensors such as vision, force or tactile sensors for feedback.

Many techniques have been developed to analyze sensor data and provide some type of recognition or identification capability. These techniques fall under two related sub-areas of *image processing* and *pattern recognition*. Image processing deals with operations on images to improve their quality or to extract features. Pattern recognition, on the other hand, is concerned with the identification or interpretation of the image. Depending upon the type of sensor used, image processing is normally needed before accurate identification can be done. Standard classification techniques, such as the nearest neighbor method, hyper plane separation, feature weighting and rotated coordinate method, all approach the problem in a rule based, analytical fashion [10]. These methods rely on initial image segmentation or edge detection algorithms to extract desired features for subsequent classification. In most cases, a video based sensor is used to provide image data in the form of a pixel array that can be in a binary, grey-scale or color format. Many commercial systems using video or CCD cameras are available for image processing and object recognition [10].

Many applications require more than just a 2-dimensional image provided by a video system. Methods for non-contact distance gauging, by using optic, acoustic or magnetic sensors, are being developed to provide complete 3-dimensional images for functions such as object grasping and object avoidance [10,22]. Stereo vision, which uses multiple cameras and triangulation to compute distances, is being studied quite extensively. No simple solution has been found for the complex task of dealing with multiple, stereo images because considerable data processing and ideal optical conditions are required. On the other hand, laser range finding cameras, providing full 3-D images, are also receiving attention despite their cost and complexity. However, they require delicate mechanisms to scan the light source across the field of view and are considerably more expensive than video based vision systems.

The work presented here focuses on the use of a relatively simple sensor, a one dimensional laser range finder, to provide a robot with the ability to recognize and measure an object's location. The sensor is capable of measuring the distance between the laser source and an object interrupting the emitted light beam. Hence, it can be classified as one dimensional. In addition, the sensor has a limited measurement range within which it is linear. Within this range, a simple analog signal is produced which is proportional to the distance to the target. Despite its limitations, this type of sensor has several advantages over other vision systems. These advantages will be discussed further in section 4.2 and Chapter 5. Considering its depth perception capabilities and compact size, the laser sensor provides a powerful tool in a robotic environment. However, the type of scene or image information retrieved is unconventional and unique compared with pixel based, vision systems. Pattern recognition and transformation capabilities of artificial neural networks will be employed to interpret the measured image data.

4.2 Description of the Laser Sensor

The concept of using a laser beam coupled with a mechanical scanning arrangement, provides one way of recovering 3-D information of the real world [10]. Depth information offers an alternative method of scene analysis which normally must deal with techniques based on intensity, color or texture of a 2-D video image. Most laser scanning techniques involve free standing scanning systems and they use rotating mirrors to sweep a beam across the object. A different approach has been followed here to examine the use of a much simpler and less expensive laser range finder.

The sensor utilized in this study is an Adsens Tech, model LAS-8010V, Laser Analog Displacement Sensor [24]. This device provides an analog signal which is proportional to the target distance within the measurement range of 100 mm $\pm$ 40 mm. The sensor is a lightweight 310 g and it is compact, measuring only $80 \times 60 \times 20$ mm. It uses a 1 mW, 680 nm, visible laser beam (class 3b) which is projected onto the target surface and requires a 12-24 VDC power supply for its operation. The reflected beam is collected by using a receiver within the device and an analog voltage is produced which is proportional to the distance between the laser and the target. The sensor is capable of measuring this distance to an accuracy of 50 μm, providing the target is within the measurement range. The visible laser spot of the sensor, which is oval in shape, is approximately 0.8×1.6 mm in size at a target distance of 100 mm. The size increases to 1.2×2.5 mm at 60 mm distance and decreases to 0.5×0.9 mm at 140 mm distance. A *control* output is available which is switched either on or off (so that it is called a *binary* output) depending on a threshold distance which is selected through an adjustable pot on the sensor. A *dark* output indicates that no object is present or that the reflected beam is too weak to register. The device, however, will have difficulty detecting mirrors, transparent or very dark objects. Figure 4.1 illustrates the present arrangement of the laser-robot configuration.

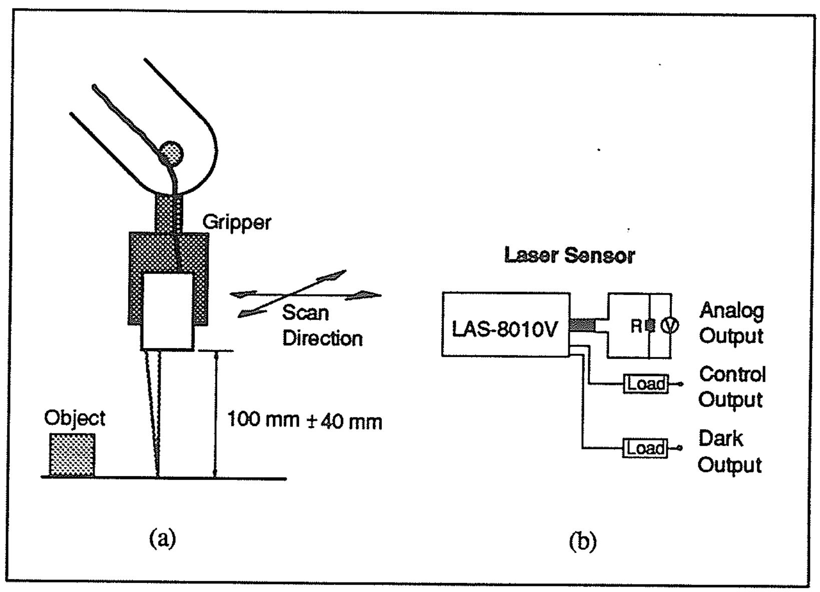


Figure 4.1. (a) The laser and robot configuration and (b) the sensor outputs

The sensor was attached to the robot's gripper and the analog as well as the control outputs are connected to a standard analog-to-digital converter (ADC card). The sensor is essentially 1-dimensional and, hence, it must be moved across the object while simultaneously sampling the outputs to retrieve a true 3-D image of the scene. Using a predefined, zig-zag robot path in a horizontal plane above the object, the sensor scans the work area and collects range data for any object within range. The quality of the image produced is a function of the robot's positional accuracy, the scanning speed and the time taken for conversion of the range to image data. For the present configuration, it is not possible to have high resolution images such as those obtained from CCD cameras. However, as will be demonstrated in this thesis, by using the pattern recognition capability of ANNs to retrieve sufficient data from relatively few scans, it is possible to identify as well as determine the object's location. The main advantage of the sensor is its size and simplicity and requires a standard A-to-D data acquisition interface for digital processing. Being a laser based vision system, it is not susceptible to the problems created by poor background lighting or to shadows created by objects that arise with video based vision systems. Edge detection and segmentation is simply a matter of locating range discontinuities in the image. The unique pattern of range discontinuities produced during a robot scan are interpreted by artificial neural networks, a powerful new tool being applied with great success in various applications [4,25].

4.3 Neural Networks in Robotics

Within the last 10 to 12 years, there has been an increased interest in systems that learn by using models of biological neurons [25,26]. Artificial neural networks (ANNs), as they are called, are being used in many new applications such as speech recognition, vision, motor and motor-sensor control and traditional areas such as pattern recognition and signal processing. ANNs have the potential to analyze very complicated behavior as well as the ability to solve complex non-linear problems because of their massively parallel structure of interconnected, nonlinear systems and, more importantly, because of their learning capability. A renewed interest in their ability for robot control and object recognition is due to both advances in hardware as well as recent developments of multi-layer networks which learn by employing methods such as back propagation.

A neural network learns the relationship between input and output variables by being exposed to representative examples of their relationship. This ability is used to find a general relationship between variables that are difficult or impossible to relate analytically. In terms of recognizing objects, the input may be the pattern of range data during a scan over an object and the output is simply a classification of the data into a category representing a previously trained object. A fully trained network becomes essentially a "black box" with the power to model complex, non-linear problems relatively easily. The procedure for using a network consists of two phases: (i) a training phase, and (ii) a testing phase. In *recognition* applications, training consists of exposing the network to input-output pairs allowing it to learn their relationship which, in many cases, may have no direct analytical solution. The testing phase involves presenting the network with an input which may have been corrupted by noise. The network is expected to provide an output corresponding to the input, despite the noise that is generally present. Another application, broadly termed *generalization*, differs only in the testing phase where a trained network is given an entirely new input and it is expected to predict an appropriate output based on the internal model developed through training.

Both types of applications are used in robotics. Several researchers have shown the applicability of ANNs to tasks such as the dynamic control of a manipulator [1], real-time control of robots with vision feedback [2], payload estimation for adaptive control [27] and path trajectory generation with obstacle avoidance [28]. Neural networks are particularly well suited to pattern recognition problems of varying complexity. Miller et al [1,2] use the CMAC (Cerebellar Model Arithmetic Computer) network for both pattern recognition and robot control in a real-time, robot tracking problem in which an industrial robot, fitted with a video camera attached to the gripper, tracks a moving object on a conveyer. This work focused more on robot control using ANNs and CCD cameras to track objects and did not provide any type of recognition or identification capabilities. Depth perception was not needed because it was assumed that all parts were at a fixed distance relative to the camera. Watanabe et al [3] successfully applied ANNs to 3-D object recognition by using an ultrasonic sensor and an array of receiving elements. Identification rates greater than 99% were obtained without any need for tight control on the object's position or orientation. However, this study was concerned only with object identification and did not attempt to solve the problem of determining the object's location. In addition, the ultrasonic receiver array was fixed above the work area and did not provide the flexibility and convenience attainable with a gripper mounted sensor. Ultrasonic sensing also suffers from inferior accuracy, lack of image detail and poor beam localization providing, in a sense, a badly focused image. The methodology presented in this thesis represents a departure from these studies in that it attempts to provide *both* recognition and measurement capabilities using ANNs and a simple, precise, compact, laser sensor.

Various types of neural network architectures exist for different applications. By far the most widely used is the multilayer perceptron (MLP) network which provides extremely powerful recognition and learning capabilities. It also accepts continuous input and output rather than simple binary values which gives greater flexibility in solving practical problems such as object recognition.

4.4 Multilayer Perceptrons and the Back Propagation Neural Network

The MLP is the most popular artificial neural network used today [29,30]. The back propagation learning algorithm is one of the most widely employed methods of training the MLP network and it has helped to revive the study of ANNs for many practical applications [29]. This system was developed first in 1974 [31] and it has evolved as a powerful tool capable of performing complex, non-linear mappings in pattern recognition problems. The MLP network is an extension of the simple *perceptron*, first conceived in 1958 by Rosenblatt [32]. A brief discussion follows.

4.4.1 The Perceptron

The perceptron shown in Figure 4.2 forms a weighted sum of the input vector and adds a bias value, θ, which acts as a threshold. The result of the summation provides an input to a non-linear function, such as the hard-limiting or the sigmoid function shown in Figure 4.2, which then produces a binary or continuous output, respectively.

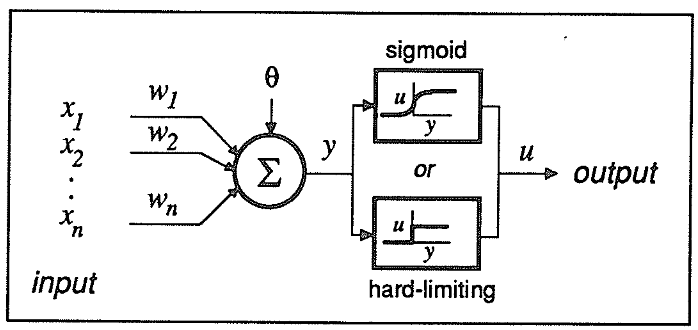


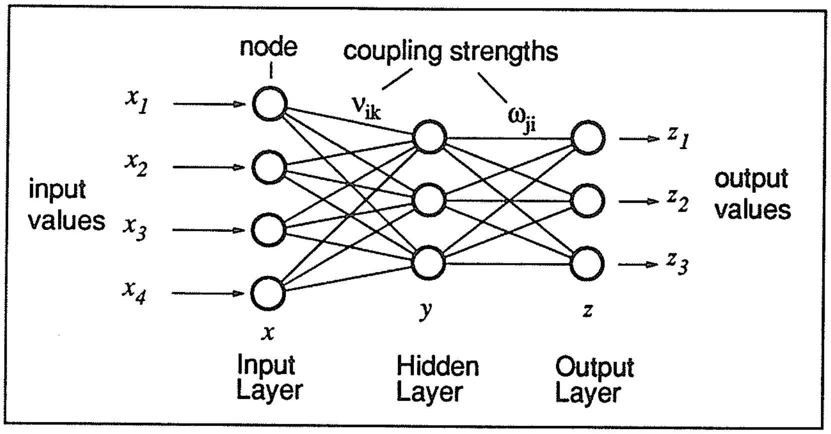
Figure 4.2. The perceptron

The perceptron can act as a discriminate function which performs a non-linear transformation from x_i, the input vector, to u, the output. Thus, for a two-class pattern recognition problem, the perceptron partitions the input into two regions by using a linear decision boundary [25]. It can also act as a binary logic function capable of performing Boolean algebraic expressions such as *AND*, *OR* or *COMPLEMENT*. However, its simple architecture is capable of many, but not all logic functions. For example, it is unable to solve the *XOR* problem. Moreover, practical problems requiring non-linear partitioning cannot be solved by using the simple perceptron architecture illustrated in Figure 4.2.

4.4.2 The Multilayer Perceptron

The multilayer perceptron is formed by cascading perceptrons into layers. Individual perceptrons are called neurons or *nodes* which are arranged as *layers*, as shown in Figure 4.3. The MLP can perform any logic function, can implement complex non-linear transformations and can solve complicated, *n*-dimensional recognition problems [25]. The applicability of this architecture for robotic object recognition will be pursued in this thesis. The details are presented in Chapter 5.

Three important issues must be addressed in using the MLP network with back-propagation. First, is the issue of network size. How many hidden layers are required ? How many nodes per hidden layer should be used ? Second, the ability of the network to generalize is important. The network may perform well when presented with training data, but how will it perform when new test patterns are used ? This is especially important here because the robot's positional inaccuracies will inevitably produce input patterns that are not highly repeatable. Therefore, generalization is important in retaining good accuracy for new patterns. Finally, the time required to learn the desired mappings must be considered. The back propagation algorithm used for training, discussed further in the following section, suffers from slow convergence requiring many thousands of iterations during training. This aspect is not so important in the present work because training could be done off-line on high performance computers. A SUN 4 and an HP-9000 workstation were utilized in this study. These machines are capable of performing thousands of iterations requiring millions of floating point computations in only a few minutes. Therefore, it could be determined quickly if training is going to produce accurate results with the current network architecture. This is done by observing the initial convergence rate of the network, which should rapidly approach zero error early in the training process (exponential decay). If it does not, the network size must be altered for a better solution.



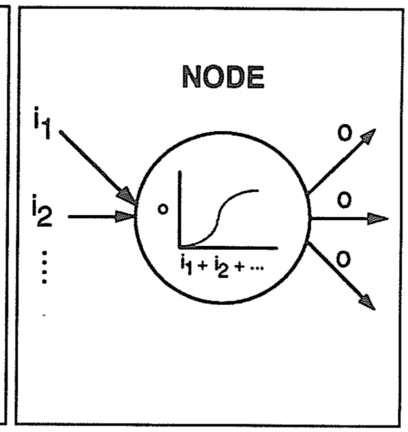


Figure 4.3. An MLP network having a single hidden layer

The network size refers to the number of layers (excluding the input layer) as well as to the number of nodes per layer. The size needed is difficult to predict because no standard rules or formulas exist for most problems. Guidelines have been formulated [30,33], although they are usually only an aid to improve a network's performance. If the size of a network is too small, it will not be able to create an accurate model. However, if the size is too big, the network may be *too capable* [25] in that it can implement numerous solutions, each with poor accuracy. Generalization can also suffer with an oversized network. Network performance is insensitive to over specification of size in "1-class" classifier problems [30] (i.e. problems in which decision boundaries completely surround the pattern classes). However, smaller networks (2 or 3 layers, excluding the input layer) are used more widely to reduce intensive computations. In fact, it has been shown that two layer networks perform better, in some cases, than three layer networks [30]. The lower bound on the number of nodes in the hidden layer is $(d + 1)$, where d is the dimension of the input vector. An optimal number of nodes has been shown to be $3d$ [30]. With little knowledge of the problem, trial and error is often used to determine a network's size. Beginning with a small network (single hidden layer and $d+1$ hidden nodes), training continues and more nodes are added until performance levels off. Using these guidelines, two layer networks (one hidden layer and one output layer) were used in this study. However, depending on the complexity of the recognition problem or the number of training samples available, the number of hidden nodes was varied.

The MLP network is ineffective if a method of changing the coupling strengths during the learning phase is not available. Gradient descent learning [29,30] is normally used to "train" the network to perform the non-linear transformations. This technique incrementally adjusts the network *weights* (coupling strengths) in order to minimize an error function, typically the total sum-of-squared-error (*tss*) of the output. The back propagation algorithm is used to train the multilayer perceptron.

4.4.3 The Back Propagation Algorithm

Back propagation has been developed as a method of training multi-layer, feed forward networks to map a set of inputs to a desired set of outputs. The structure of a two layer network is shown in Figure 4.3.

The mapping of input vectors to output (or target) vectors is specified by a desired activation state for each unit (or node) in the network. The input and output of the network corresponds to the activation state of each node in the input and output layer, respectively. Learning is done by iteratively adjusting the coupling strengths between each unit (ν and ω shown in Figure 4.3) of consecutive layers to minimize the difference between the actual output state vector and the desired state vector. During the learning process, an input state vector is presented and propagated forward to determine the output signal. The output and target vectors are compared which results in a difference or error signal that is *propagated back* through the network in order to adjust the coupling strengths or *weights*. This *back propagation* of error effectively *teaches* the network the relationship between the input and output vectors.

The general formula for the activation function of each unit in the network (except input units whose activation is equal to the input vector) is given by the sigmoid function :

$$a_j(w, bias, a_i) = \frac{1}{1+e^{-\left(\sum_{i=1}^{N}(w_{ji}a_i)+bias_j\right)}} \tag{4.1}$$

where:

w_{ji} = strength of coupling between unit j and unit i in the previous layer

a_i = activation of unit i in the previous layer

$bias_j$ = threshold or bias of unit j

N = total number of units in previous layer.

The bias is like a coupling to a unit having full activation and it is treated just like w. Consider the two layer network shown in Figure 4.3, which consists of 4 input units (x), 3 hidden units (y) and 3 output units (z). The activation for a hidden unit is,

$$y_j = a_j(\nu, bias_j, x)$$

and, for an output unit,

$$z_j = a_j(\omega, bias_j, y)$$

where

ν = strength of couplings between layer x and y, and
ω = strength of couplings between layer y and z.

The error function, E, measures the performance of the network. E is defined as

$$E = \frac{1}{2}\sum_{c=1}^{N_c}\sum_{j=1}^{N_z}\left(z_{j,c} - t_{j,c}\right)^2 \tag{4.2}$$

where

c = index for each input pattern,

N_c = number of input patterns,

j = index for output units,

N_j = number of output units,

$z_{j,c}$ = activation of output unit j for pattern c (actual value), and

$t_{j,c}$ = target value of output unit j for pattern c (desired value).

The goal of the back propagation algorithm is to minimize the error, E, through a least-mean-square (LMS) procedure. To minimize E, each coupling strength, ν or ω, is updated by an amount proportional to $\frac{\partial E}{\partial \nu}$ or $\frac{\partial E}{\partial \omega}$, respectively.

The partial derivatives of the error function are given by [8],

$$\left.\begin{aligned}\frac{\partial E}{\partial \omega_{ji}} &= \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial \omega_{ji}}\\ \frac{\partial E}{\partial z_j} &= z_j - t_j\\ \frac{\partial z_j}{\partial \omega_{ji}} &= z_j(1-z_j)y_i\end{aligned}\right\} \tag{4.3}$$

and

$$\left.\begin{aligned}\frac{\partial E}{\partial v_{ik}} &= \sum_{j=1}^{N_z}\frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i}\frac{\partial y_i}{\partial v_{ik}}\\ \frac{\partial z_j}{\partial y_i} &= z_j(1-z_j)\omega_{ji}\\ \frac{\partial y_i}{\partial v_{ik}} &= y_i(1-y_i)x_k.\end{aligned}\right\} \tag{4.4}$$

The factor $(z_j - t_j)z_j(1-z_j)$ occurs in both $\frac{\partial E}{\partial \omega_{ji}}$ and $\frac{\partial E}{\partial v_{ik}}$ which indicates that this factor is propagated backward from the output layer to the hidden layer.

Now that the derivative of E is defined, any weight in the network is updated according to the following rule [8]:

$$\Delta\omega_{ji}(s+1) = -\eta\sum_{c=1}^{N_c}\left(\frac{\partial E}{\partial \omega_{ji}}\right) + \alpha\Delta\omega_{ji}(s) \tag{4.5}$$

and

$$\Delta v_{ik}(s+1) = -\eta\sum_{c=1}^{N_c}\left(\frac{\partial E}{\partial v_{ik}}\right) + \alpha\Delta v_{ik}(s) \tag{4.6}$$

where

s = sweep number or number of times all cases have been presented (updating occurs after all cases have been presented),
η = constant learning rate, and
α = momentum factor or relative contribution of the previous change in the coupling strength.

This method has been implemented in software to process the laser data retrieved during a scan across an object. It provides a means to teach the MLP network to identify a pattern of range values corresponding to a particular object. The software will be discussed further in the following section.

4.4.4 Back Propagation Software

The back propagation method described in section 4.4.3 is implemented through the **bp** software program included with [8]. This program can be used on an IBM PC compatible computer running MS-DOS or a SUN workstation running UNIX or SunOS. The program was modified in order to make it compatible with the QNX operating system and the Watcom C compiler used for the on-line robot controller.

The program requires three user-prepared files before it can be executed. These are the template file (*.tem), the network file (*.net) and the start up file (*.str). In addition, a pattern file is needed to specify the input-output vectors required during training. Upon completion of training, the network weights can be stored for later recall during testing. The reader is referred to reference [8] for a complete description of the format of these files as well as many sample problems.

The program is menu driven and it allows on-line modification of several network parameters during training. The display can be tailored to meet custom requirements through specification of the template file. The most important parameter during training is the total-summed-squared error, *tss*. This is a measure of how well the network is capable of mapping all the input vectors to their corresponding outputs for each training set. The *tss* parameter is displayed on screen and it is updated constantly, during training, to indicate the performance of the training. During a typical training run, the important parameters displayed include the pattern number, *tss*, *pss*, *nepoch,* the input vector, the target output and the network's actual output or activation at the output layer. The *pss* refers to each individual pattern's summed-squared-error, as opposed to *tss* which is a summation over all patterns in the training set. (The set of all patterns is known as an *epoch.*) The *nepoch* parameter indicates the number of times all patterns have been presented to the network during training. Each pass of all patterns (one *epoch*) results in an update of all network weights by using the method described in section 4.4.3.

The back propagation method suffers from long training times because of slow convergence inherent to gradient descent algorithms [34,35]. Training was done off-line on a SUN 4 and HP-9000 workstation. When the network converged to within a reasonable error limit, determined by the maximum *pss* for the current *epoch*, the network weights were saved to a file and then ported to the robot's control system for subsequent testing. Thus, the robot's controller is not responsible for training the network. With a multi-tasking operating system, on-line training is possible although very time consuming with the current hardware capabilities. Plug-in, DSP based, floating point processor boards accessible by the robot control software may provide one option to make on-line training possible.

This concludes a brief overview of the MLP network and the training method used in this work. The following chapter details its implementation in relation to a laser based object recognition application. Several methods have been developed recently [33,34] to improve the performance of the back propagation algorithm which generally suffers from slow training times which make real-time implementation difficult. These methods are beyond the scope of the thesis although they show considerable potential for improving the back propagation method's effectiveness as a MLP learning technique.

CHAPTER 5

DESCRIPTION OF THE RECOGNITION AND MEASUREMENT SYSTEM

5.1 Introduction

A new method of object recognition has been developed by using the sensor described in section 4.2 and the flexible robot controller outlined in Chapters 2 and 3. The transparent robot control software is integrated with an ANN routine to collect and interpret the pattern of range data from the laser sensor. In addition to recognition (i.e. *identification* of a randomly located, previously unknown object), a measurement is made of the object's location and orientation. This measurement is needed to achieve the ultimate goal of manipulating the object with the robot's end-effector. ANNs are used for the initial recognition and subsequent measurement of orientation. Details will be presented in this chapter followed, in Chapter 6, by a discussion of the experimental results.

The object recognition method developed here can be divided into the two main functions of object identification as well as measurement of the object's position and orientation. When an object has been identified, a pre-determined method of approach and grasp can be utilized (assuming that sophisticated methods of tactile sensing are unavailable) to make the task of manipulating a randomly located object possible. The measurement of an object's position, in addition to recognition, is absolutely necessary if objects are in an unstructured environment. i.e. placed randomly within the robot's workcell.

Recognition is the first step towards manipulating unknown objects within the robot's work space. Identification must be made before a suitable approach can be found to the measurement of position. Only planar objects are considered first. These objects are all assumed to have uniform thickness, although their shapes vary in complexity. The proposed method uses a gripper mounted, laser sensor and simple robot scan paths to accomplish the recognition and position measurement task. This approach allows the sensor to be mounted easily on any robot. The cost of the laser sensor is approximately \$ 2850 and a standard A-to-D interface card is about \$ 500. A typical vision system may cost between \$ 3000 and \$ 5000 which excludes the cost of the software. Although component costs are similar, the laser system provides further advantages such as ease of use, size, mounting arrangement and overall system complexity which requires far less attention compared with typical vision systems. The sensor does not require a structured lighting arrangement for its operation. In fact, it can even operate in complete darkness, if so required. Unlike vision systems, it can reach areas within machines and assembly cells wherein adequate lighting cannot be guaranteed. The compact size makes it possible to mount the sensor on a robotic gripper without reducing the payload significantly. Before discussing the methods used to recognize and measure an object's location, the data acquisition system will be described next.

5.2 Data Acquisition

The laser sensor, introduced in Chapter 4, provided range data in the form of an analog signal. The range data must be converted into a digital format for further analysis by the ANN routines. Standard PC based, A-to-D data acquisition hardware was used to interface with the laser outputs.

The precise end effector positions (or joint positions) during a scan path was necessary in order to correlate the sensor's output with its Cartesian position. This was required for object identification and the measurement of an object's centroidal position based upon a 2-D image (threshold range data versus $\{X,Y\}$ position of the object). This requirement alone precluded the use of most commercial robot controllers which are incapable of communicating high speed, real-time information about the joint's positions to peripheral equipment. To determine an object's angular position, range data was needed only as a function of time. Precise sensor locations were found on-line by storing all five joint positions and the laser's output into random access memory (RAM) during a scan and then post-processing the data to determine precise laser $\{X,Y\}$ coordinates and threshold range data. Moreover, many tasks were executed during a scan period and each task required significant CPU resources so that the most efficient method of position retrieval was to store the joint data into random access memory (RAM). Joint data was retrieved at the completion of the scan for further processing. The analog laser outputs were sampled at 250 Hz and they were synchronized with the position sampling frequency of the rotary encoders located on each robot joint.

At the beginning of a scan, five blocks of memory were allocated to store each joint's position and another block was used for the laser data. Depending on the size of available memory, only a restricted number of positions could be stored for any scan. Fortunately, unlike commercial robot controllers, a PC based controller can economically provide sufficient memory for retaining detailed images. At every sampling instant, each joint position, read from the motion control interface card, and the laser's range output, read from the ADC card, was stored in separately allocated memory blocks. After the scan was completed, a forward kinematic transformation was performed to convert the joint positions to Cartesian positions. The data was then in the form of the sensor's $\{X,Y\}$ position versus object range at every sampling instant.

To simplify the process of finding part edges during the scan, the binary output of the laser sensor was used instead of the analog output. The laser sensor will produce a binary output when the range value exceeds a certain threshold value. The threshold value is selectable by using settings contained in the laser sensor circuitry. The binary output acted as a switch which was turned *off* or *on* depending upon the threshold range setting of the sensor. The threshold range setting was typically set at a distance corresponding to a few millimeters above the working surface and the control line then switched a 5 volt output on the ADC input channel. The switch was *on* when the range values were measured beyond the threshold and it was *off* when the range values were within the threshold distance. This scheme resulted in much more accurate and easily extracted edge patterns compared with a true analog output from the sensor. However, it provided only a binary picture of the scene rather than a true 3-D image. The laser's linear range output was very sensitive to the angle at which an object was scanned, or, the orientation of the laser relative to the scanned object. The incident and reflected beams should be broken simultaneously for an accurate measurement of the range, otherwise, the analog output may experience high amplitude signal fluctuations. The binary output does not suffer from this problem and, for this reason, it was used in this study. More attention must be made to filtering the analog range output, the subject of future work, before it is suitable for accurate recognition and position measurement. To illustrate the difference between the two laser outputs, Figure 5.2 compares (a) the binary output, and (b) the analog range output for a single scan across a 38 mm $\times$ 38 mm square block located at the different angular positions shown in Figure 5.1. The square object protruded approximately 15 mm from the table surface and above the threshold distance set on the sensor. The laser's output was sampled at 250 Hz and the path velocity was approximately 25 mm/sec. Figure 5.2(a) shows a much sharper edge discontinuity during the laser scan, a direct result of using the binary output which provided only two states (0V and 5V). Using the analog output, the edge and, more importantly, the time at which the edge was reached, was less obvious and more difficult to determine accurately, as shown in Figure 5.2 (b).

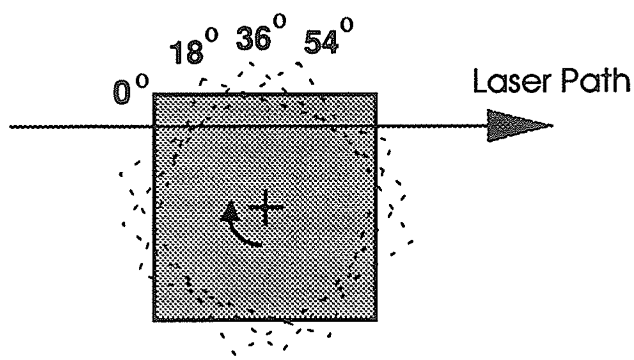
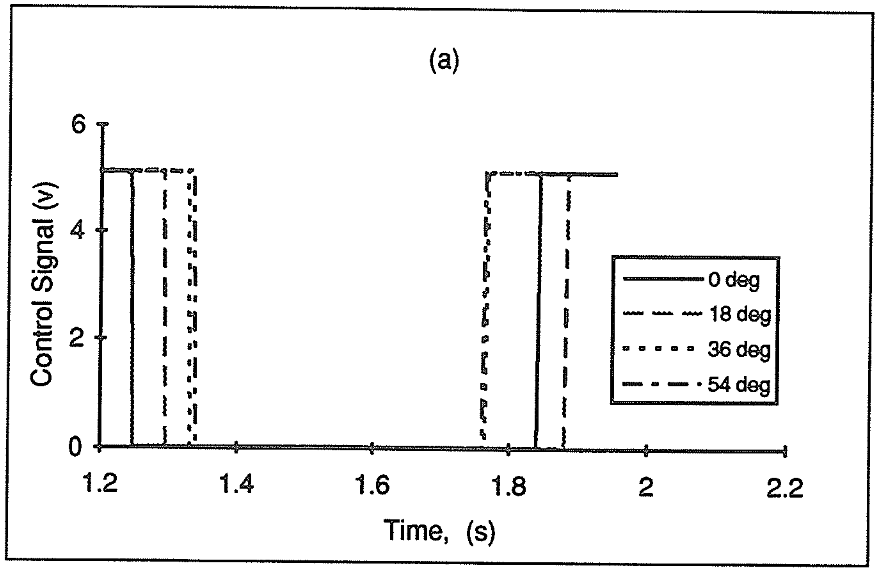


Figure 5.1. Single laser pass over a square block which was used while sampling the laser outputs (plotted in Figures 5.2 (a) and (b))

The task of recognizing and measuring an object's position was divided into two separate sub-tasks. Each sub-task required a different methodology, although both use neural networks to achieve their goal. The object recognition procedure required much more data, in the form of a 2-D binary image, than object position and orientation measurements. Each sub-task is described in the following section.



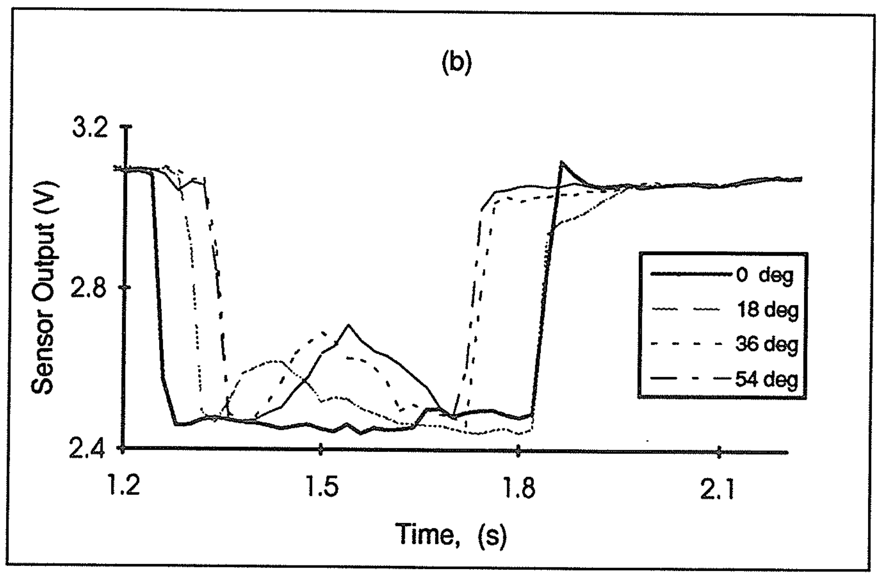


Figure 5.2. Edge detection of the square shape with a single pass using (a) the binary output and (b) the analog output for four angular positions

5.3 Recognition of Planar Objects

Recognition of any planar object must be independent of the object's orientation (in this case rotation about its vertical axis). A similar method to the one described in [3] was used to calculate rotation invariant, feature values for object recognition by employing neural networks. A feature value vector (FVV) was used as the input to the back propagation neural network. From a 2-D image, $f(x,y,z)$, a feature value $s(r,z)$ was found from

$$s(r,z) = \int_{D(r)} f(x,y,z)dxdy \tag{5.1}$$

where

$$D(r) = \{(x,y), r^2 \leq x^2 + y^2 < (r+a)^2\} \quad , \tag{5.2}$$

a = constant annular separation distance (in radial direction),

x, y, z = scan area coordinates corresponding to robot's world coordinate frame,

$f(x,y,z)$ = image value at location x, y, and z (z is fixed in this case), and

r = polar coordinate corresponding to position $\{x, y\}$.

This method integrated the scene data obtained with respect to a polar coordinate frame, $D(r)$, such that the center of the frame was located at the image's centroid. This, effectively, provided a feature value (i.e. an integration of binary image values) based upon annular sections located radially within the image. If the center of the polar coordinate system used for the integration was set at the object's centroid within the image, the feature value would always be independent of the object's rotation.

The method required the robot to perform a scan with the laser fixed in a horizontal plane above the object (65 mm above the working surface), as illustrated in Figure 5.4 (a). The height of 65 mm was chosen arbitrarily although it must be within the measurement range of the sensor (less than 140 mm and greater than 60 mm). It must also be greater than the thickness of any object being scanned to avoid collisions. An identical path was chosen for scanning all objects lying within the scan area. A zig-zag type of robot path, chosen for its effective scan area coverage and ease of programming, was used to scan all objects. Other types of paths were tested, including a spiral path and a purely horizontal and vertical path (ie. along lines of constant x and y) with short jogs at the end of each scan to position the sensor for completing the next traverse. These types of paths were more difficult to program, requiring many more positions to properly define their shape, and did not improve scan area coverage compared with the zig-zag path. Five different objects were chosen to evaluate the object recognition method. They included a

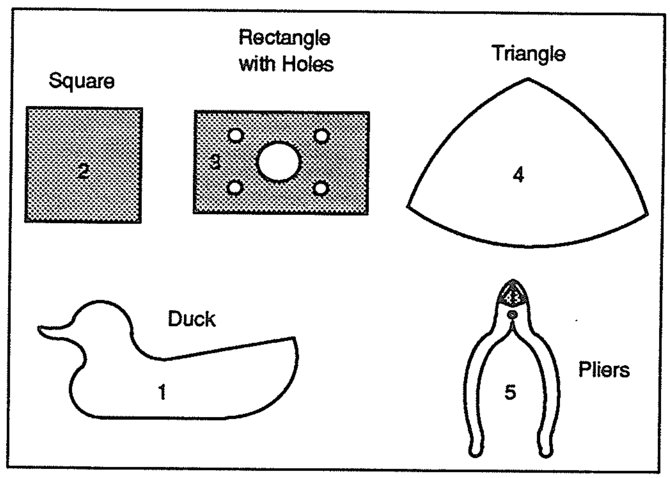
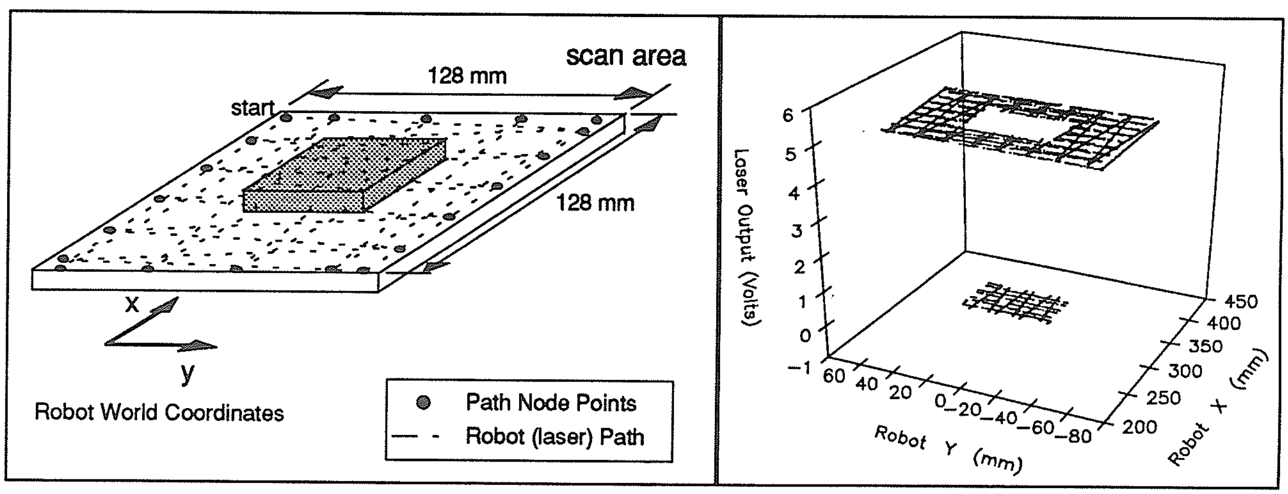


Figure 5.3. Planar shapes (and corresponding object number) used for object recognition experiments (not to scale)

square, a triangle with rounded edges, a rectangle with 5 holes drilled through its mid-section, a small pliers and an object with contours on the boundary in the shape of a duck. The objects ranged in size and complexity and they were all planar parts having a constant thickness. These object's are shown in Figure 5.3.

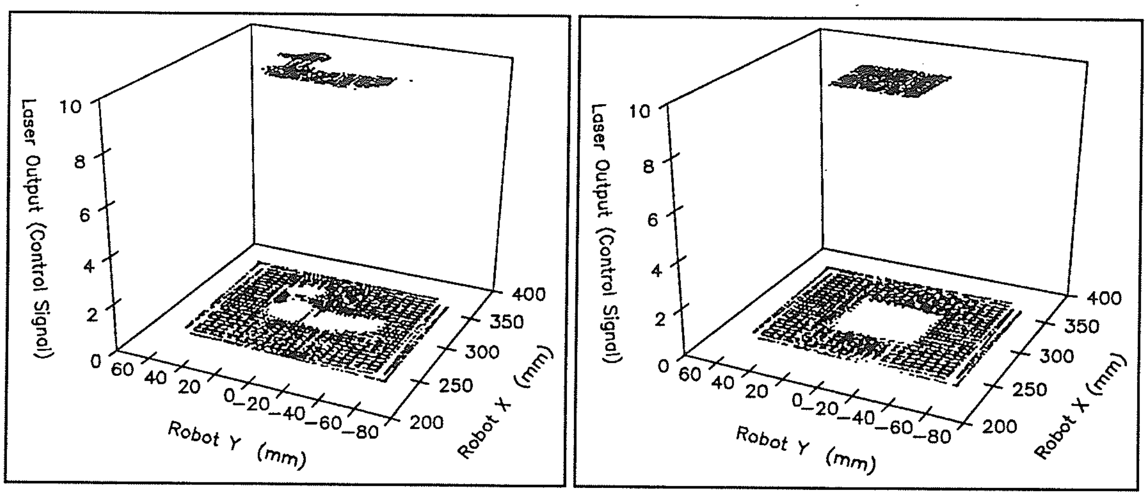
The robot manipulator positioned the laser sensor directly above the working surface at a fixed height and a predetermined starting position. A zig-zag path, illustrated in Figure 5.4(a), was used to generate the image. This path encompassed the entire field of view (or *scan area*), in this case a square having a plan view of 128 × 128 mm. During the execution of the path, the joint positions and laser output were read every 4 ms and stored in memory. The scan path illustrated in Figure 5.4(a) provided a very coarse, line scan image of the square object shown in Figure 5.4(b). The number of scans across the part was varied in both directions to determine an optimal number of *crossings* for a reasonably accurate recognition of a broad range of object shapes. Initially, 17 *crossings* were used in each direction, to give a total of 34 scans, for the part shown in Figure 5.4 (b). A trade-off occurred in defining the number of scans needed to recognize objects. Fewer scans can be executed more quickly but they provided a low resolution image which made it difficult to recognize similar objects with small, distinguishing features that



(a) (b)

Figure 5.4. (a) Low resolution object recognition scan path and (b) resulting 2-D image

would not appear in the image. However, increasing the number of scans to provide better resolution took more time for the robot to complete and would render the method impractical. Path speed of traversal by the robot and laser was also an important factor and it was related directly to the sampling rate of the joint position encoders, fixed at 250 Hz. As the robot's path speed was increased, the laser range data resolution along the path degraded because sampling the laser's output must coincide with the sampling rate of the encoders (a fixed value) in order to correlate actual joint positions with the laser's output. Poor laser data resolution along the path resulted in inconsistent edge detection patterns. Therefore, a balance was needed between image resolution, scan time and path speed in order to provide a practical recognition system with current hardware capabilities. Fortunately, the maximum path speed and acceleration the robot was capable of reaching before excessive vibrations occurred (approximately 50 mm/s at an acceleration of 500 mm/s/s) provided satisfactory edge detection patterns. After several



(a) (b)

Figure 5.5. High resolution line scan image of (a) a duck, and (b) a square

tests, the image resolution was increased to a total of 98 laser scans across the scan field, 49 in each of the robot's global *X* and *Y* directions, which required approximately 35 s to complete. This resolution provided feature value patterns for each of the objects shown in Figure 5.3 that adequately reflected their individual, distinguishing features. As a result, the high resolution scan path (98 scans) was chosen for all recognition tests used to asses the performance of the recognition method. Figures 5.5 (a) and (b) illustrate the resulting 2-D line scan image of two other shapes, shape 1 and shape 2 of Figure 5.3, using this high resolution path.

After a 2-D line-scan image was generated (as shown in Figures 5.5(a) and (b)), the FVV was calculated, using equation (5.1), which was independent of an object's rotation about the vertical. Equation (5.1) effectively integrates over the function (or object's image value) along a radius, r, in order to provide a rotation invariant value [3]. This method is analogous to summing individual image values within annular sub-sections $rdrd\theta$ of constant width ($a = 1$ mm) as shown in Figure 5.6. (Note: Figure 5.6 shows a very coarse representation of annular sections and sub-sections for illustrative purposes.) An image value, $f(x,y,z)$, is either 1 or 0, depending on the value of the laser's binary output sampled during the scan. (All subsections shown in Figure 5.6 that partially or completely contain a gray image feature would have a value of 1, all others are 0.)

Two integration calculations were performed. During the first integration, the object's centroid was calculated (see section 5.3.1). This centroid may, or may not coincide with the scan area's centroid. The center of $D(r)$ defined by relation (5.2) where $r = 0$, was set to the scan area's centroid during the first integration calculation. Therefore, the calculated FVV may not be rotationally invariant. A second integration was performed by setting the center of $D(r)$ to the image's centroid calculated during the first integration. (See section 5.3.1.) As an example, for a planar shape like a square, which produced essentially a 2-D or binary image regardless of the analog range or binary output used, the feature value would be constant within a circle inscribed within the square and then drop towards zero at the outer corners of the square, as shown in Figure 5.7.

The FVV was found by first generating a data file of laser position $\{x,y\}$ as well as the corresponding range values from the scan path. The scan path was in a horizontal plane at a fixed height above the working surface in order to keep the threshold distance setting fixed relative to the scan area's working surface. The data was converted to polar coordinates (r,θ) with the origin set to the scan area's centroid and sorted according to ascending radius values by using the standard *qsort()* function in the C library. This conversion increased the speed at which the FVV was calculated because the data file was in the order in which the calculation proceeded. The image value, $f(x,y,z)$, was not a continuous but rather a discrete function. Therefore, the integration in equation (5.1) must be discretized by separating the image into small areas. Each annular section had a 1 mm width (i.e. $dr = a = 1$ mm) which was divided further into angular subsections, creating small sub-areas (dA) of constant size (set at $\pi/2$ mm^2). The sub-area's size was chosen to provide a rapid yet accurate calculation of the FVV, which required approximately 5 s to compute. Smaller sub-areas resulted in little improvement, although a larger area significantly reduced the accuracy of the integration. In addition, a larger integration area, dA, would provide a better FVV pattern but would also degrade the resolution of the image during the integration procedure. The integration calculation required that, for each individual sub-area (dA), the data file be searched to verify if any binary output values equal to 1 were found in this region. If one was found, it was added to the total feature value for the annular section.

If no range value was found within a sub-area (ie. the scan path did not pass through the region), the sub-area's feature value was set to zero and contributed nothing to the overall feature value of the annular section. Figure 5.7 illustrates the FVV pattern for all five objects of Figure 5.3. Each object's curve in Figure 5.7 includes a FVV pattern for varying rotations about the object's centroid using the high resolution path of 98 laser

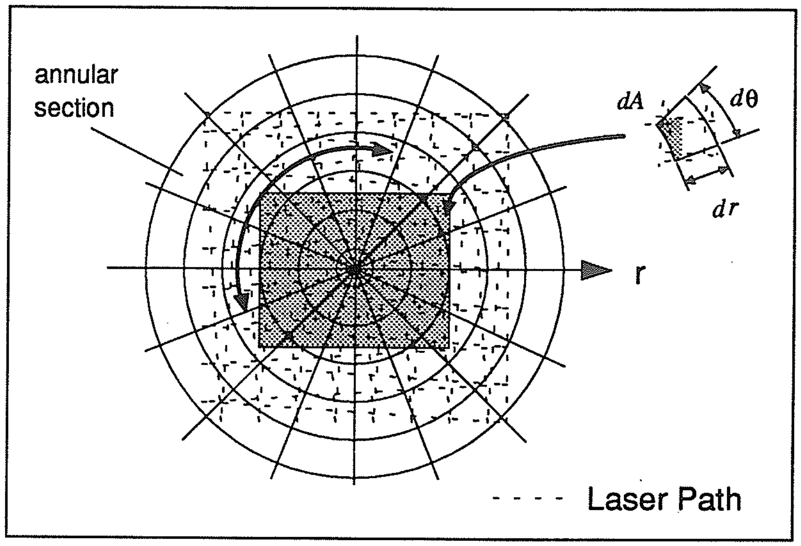


Figure 5.6. Feature value calculation for rotation invariance with square object located at scan field's centroid (x,y = 0,0)

scans. The fluctuations in the graph are caused by the coarse scan path, in that, binary values were available only along the scan path, which did not pass through every sub-area enclosing an image value ($f(x,y,z)$ = 1). A five point moving average was used to smooth the data before developing the FVV pattern for neural network training. This smoothing provided a much more repeatable FVV pattern for each object and decreased the amount of network training required. See Chapter 6 for further details.

The integration procedure was implemented in software as a post-processing routine after the scan path was completed. The software routine calculated a FVV for 91 annular sections, each 1 mm in width, which encompassed a region of 182 mm in diameter corresponding to the largest radial line of the scan area (i.e. the diagonal of the square scan area).

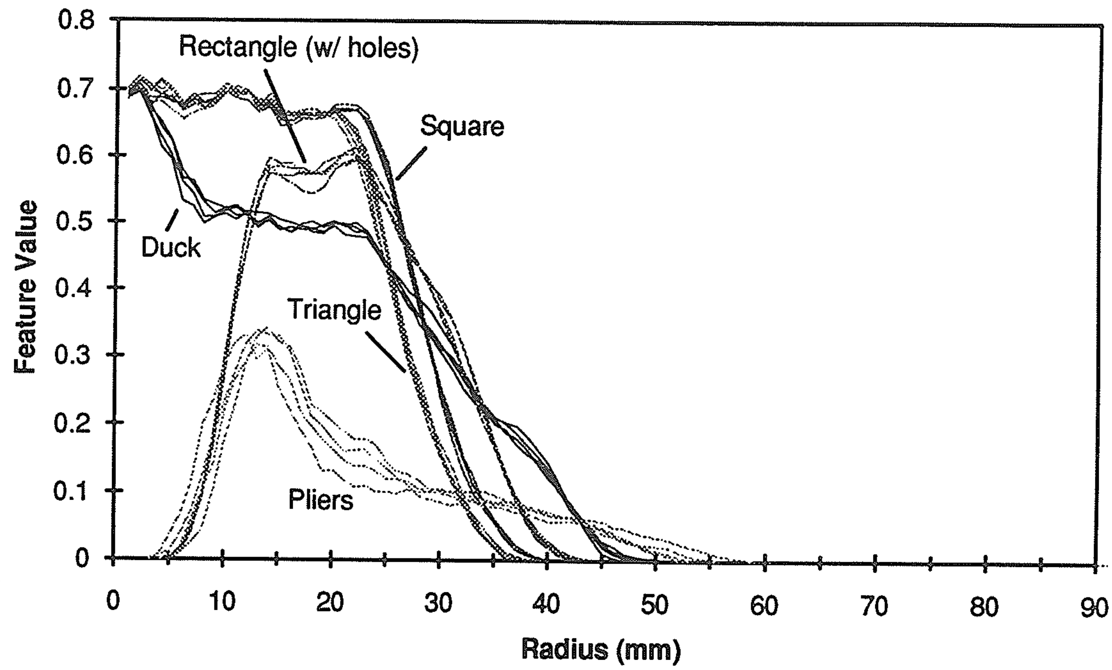


Figure 5.7. Feature value chart for the five shapes used during recognition tests

An important point to be made from Figure 5.7 is that repeated plots for each shape, corresponding to several angular positions, are almost identical. Hence the feature value was rotationally invariant. The pattern recognition capability of ANNs was used to identify the feature value pattern for each object, and, because each pattern was independent of rotation, recognition was possible regardless of orientation. The only requirement for accurate object recognition was that different objects must have unique pattern features in their FVV. Each shape in Figure 5.7 has a unique trend in its FVV plot which satisfied this requirement.

The MLP network was used to classify the laser data in order to recognize various object shapes. The laser image retrieved was, in fact, 2-D at this point because a threshold range was used to provide a binary picture (i.e. object features were either above or below the threshold range creating a *hi-lo* image). This method provided a new approach to the recognition problem by using a simpler sensor system with far less scene information compared to a typical vision system. This required less sophisticated support hardware and faster image processing. Experimental results are discussed further in Chapter 6.

5.3.1 Measurement of an Object's Position

The sampled data (x and y values versus range) found in the high resolution recognition scan used to develop the FVV pattern was also employed to calculate an object's translational displacement. The first integration performed in equation (5.1) required that $D(r)$ be located at the center of the scan area. This would result in a rotationally invariant FVV pattern only if the object's centroid coincided with the scan area's centroid. During the first integration, the object's centroid was calculated from the image value, $f(x,y,z)$, data by using the standard equations [36] :

$$\bar{x} = \frac{\sum_{i=1}^{N} A_i x_i}{\sum_{i=1}^{N} A_i} \quad \text{and} \quad \bar{y} = \frac{\sum_{i=1}^{N} A_i y_i}{\sum_{i=1}^{N} A_i} \tag{5.3}$$

where A_i corresponds to the subsection area represented by $rdrd\theta$ (Fig 5.6), and x_i, y_i is the location of the centroid of A_i.

The calculation of the centroid proved to be an accurate measurement, within ± 5 mm, of an object's translational displacement in actual tests despite the relatively low resolution line scan image used to represent the shape. If the object's centroid, $\{\bar{x}, \bar{y}\}$, did not coincide with the scan area's centroid used during the first integration calculation, another integration was performed by setting the location of $r = 0$ in $D(r)$ of equation (5.1) to correspond to the newly calculated centroid. In this way, the FVV pattern was identical for any position of the object within the scan field. This invariance was very important because an object can now be identified anywhere within the scan area, whereas, a FVV pattern was needed only for one location to train the neural network. This allowed a simple means of introducing new shapes into the network's pattern set, thereby increasing the model base of recognizable objects.

5.4 Determination of an Object's Orientation

The recognition method discussed in section 5.3 accomplished two main tasks: (i) identification of a randomly placed, previously trained object and (ii) subsequent measurement of its centroidal location. To properly grasp an object, the object's angular position must also be known in order to reorient the gripper. A second laser scan was used, in the same horizontal plane above the object as the recognition scan, to determine the object's orientation. The following method was developed for angular position measurement by using ANNs.

A simple, 3-pass scan was used to generate an input pattern for each angular position required for training. This necessitated an accurate means of locating objects at precise angular positions in order to develop a suitable training pattern set representative of all possible orientations. The experimental set-up used to collect training data is shown in Figure 5.8.

To develop an accurate set of training values for the measurement of orientation , a rotary platform was used to position each object at precisely defined angular locations. A stepper motor, with a resolution of 1.8°, was used for angular positioning. A circular platform was mounted on the motor shaft to hold each part during training.

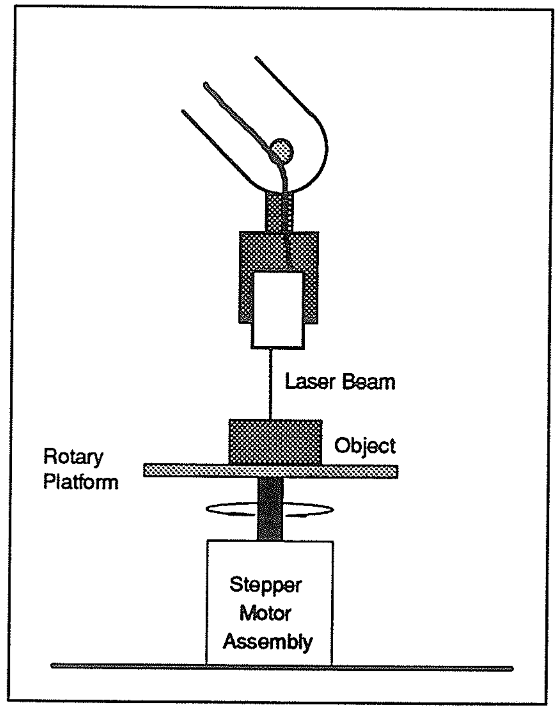


Figure 5.8. Experimental set up for angular position control used to collect training data

The whole process of collecting training data for an object positioned at various angular locations has been automated in software, an integral part of the robot controller. A sub-menu of the robot's main control software (Figure 3.4) initiates the training (and testing) process.

The operator enters the following data to begin training:

- scan path file to be used (recognition or measurement type),
- number of angular positions to be recorded, and
- angular position increment (a multiple of 1.8°).

When all necessary information has been entered, the robot begins executing the scan path and sampling the laser output. When the scan has been completed, the sensor data is post-processed to generate an input pattern vector for the current angular position (or target output). The stepper motor rotates the object and the process is repeated.

Figure 5.9 illustrates the scan path used on a square object. Unlike the recognition scan, described in section 5.3, joint position data was not needed to develop the input pattern vector for the ANN. This method simply identified the time at which the outer edge of the object was reached during each pass and generated a pattern file based on these edge detection time histories. Therefore, only binary output data was required as a function of time during the scan. The data was processed after the scan to identify the times t_1 - t_5 , as shown in Figure 5.9. The scan geometry was similar for all shapes,

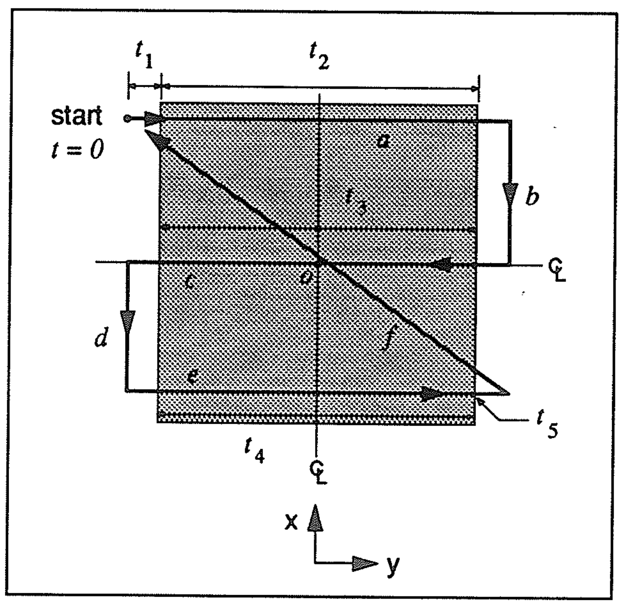


Figure 5.9. Scan path to determine an object's orientation

differing only in the x position of the first and last pass and total width of the scan, which was directly proportional to the object's size. The center of the scan path (point o in Figure 5.9) coincided with the scan area's centroid during training. A pass consisted of three sweeps of the laser across the object along a line of constant x (segments a, c, and e in Figure 5.9). The instants, t_1 - t_5, were the basis for the input vector used for training the neural network which ultimately provided a measurement of the angular position. These time instants were identified by a software routine which located the point of abrupt change in the sensor's output to indicate that the edge of the object was found. The time instants were then recorded.

The training procedure is described by the following sequence:

1. The object was placed on the rotary platform with its centroid coinciding with the platform's axis of rotation and at a known angular position (arbitrarily set at 0°). Then the robot positioned the sensor above the starting point at a fixed height above the object. This height, as discussed earlier, was set approximately 65 mm above the platform's surface.
2. The robot executed the specified path, according to the type of object identified, and began sampling the binary output of the laser sensor and storing all data in memory.
3. After the scan was completed, the robot then returned to the starting position and the stored binary output data was processed to identify the times t_1 through t_5. These times were stored in a data file, which eventually became the file for training the neural network.
4. The robot controller signaled the stepper to advance a user specified, angular step (a multiple of 1.8°) and the process was repeated.

5. Training was completed when a desired number of angular positions were scanned. This number depends on the symmetry of the part. For example, a square is unique over 90°, a rectangle over 180° and an odd shaped part over a full 360°. As a result, symmetrical objects required fewer training positions and a smaller neural network training data file.

After generating the pattern file for a part, the file was presented to the network as a set of input pattern vectors, each vector representing a different angular position. Training continued off-line. Each input pattern vector had a corresponding target output which was directly proportional to the angular position. The back propagation procedure required that all target output activation states lay between 0 and 1. Therefore, the target output depended upon the range of the angular positions used during the scan. The following equations were used to identify the desired output value for a given range of unique angular positions (assuming 1.8° angular increments):

- 360° range corresponded to 200 outputs of $o = \sum_{n=0}^{199} 0.0 + n(.005)$ (5.4)
- 180° range corresponded to 100 outputs of $o = \sum_{n=0}^{99} 0.0 + n(.01)$ (5.5)
- 90° range corresponded to 50 outputs of $o = \sum_{n=0}^{49} 0.0 + n(.02)$. (5.6)

Figure 5.10 illustrates the input pattern generated for the square by using the four inputs, t_1, t_2, t_4 and t_5 . The input value to the network was directly proportional to the actual scan times found during data acquisition. The pattern number corresponded to each angular position scanned during training, where pattern p000 = 0°, p001 = 1.8°, ... etc..

The curves in Figure 5.10 show the variation of the network's input vectors throughout the angular orientation range of a part (0-90°). It was important that the variation in the network's input pattern vector be unique for all angular positions so that training converged to an accurate solution.

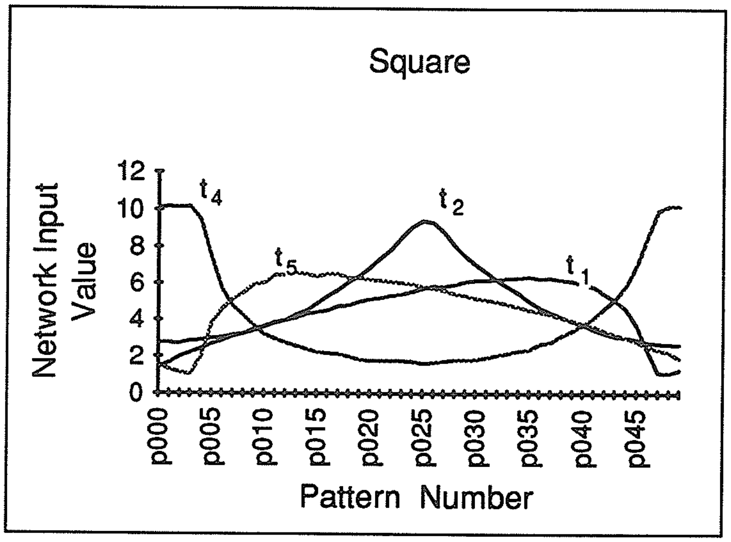


Figure 5.10. Input pattern variation for a square rotated through 90 degrees using the scan path shown in Figure 5.9

For good accuracy, the training procedure required many thousands of iterations depending on the number of input patterns and the variations between each input vector. Training was done off-line on a SUN 4 and HP-9000 workstation as a background process requiring, in some cases, 2 to 3 hours of training. The proposed method, in particular, required a great deal of training because the network must be capable of distinguishing between very similar input vectors representing each angular position. Once training was complete, the network's weight file was stored in the robot control system's memory. During actual robotic tests, the object was scanned, an input pattern was generated and a network output was found using the proper weight file. Network access at this stage, known as the *test* phase, occurred very rapidly (in real-time) providing an immediate measurement of the object's angular position. The experimental results are outlined in Chapter 6 for four of the five shapes used during recognition testing.

This method provided a unique set of patterns for any orientation of the object. One important advantage was that the method was independent of the object's shape or size. Sampling occurred at such a rate that the timing for the detection of edges was unique for each position. This, of course, was also a function of the path speed during the scan. The 3-pass scan was executed at a very slow speed (approximately 10 mm/sec) because of the limitations in sampling the ADC card through software. The ADC card used to sample the analog laser output did not have adequate DMA (Direct Memory Access) capabilities which necessitated software based sampling. This limited the sampling frequency to 250 Hz, in order to be synchronized with the sampling rate of the joint positions. Sampling at a higher speed tended to slow down all other control related tasks, running concurrently, and degraded the performance of the position control system. At this relatively low sampling frequency, the robot's path speed had to be decreased in order to provide enough resolution along the scan line so that important object features, such as edges (i.e. time instants t_1 through t_5), could be found. DMA would allow much higher sampling rates because sampled data transfers to memory do not depend on CPU resources, but rather, occur during CPU idle states thereby appearing transparent to currently executing processes. This type of data transfer can occur at much higher rates than 250 Hz (the sampling frequency used in this work), and would provide better resolution along the laser scan path allowing the robot to move faster.

Orientation training was far more demanding and time consuming than recognition training although the orientation results provided a significant improvement compared with previous ANN applications in robotics [3], which were concerned only with object recognition. In an unstructured robotic environment, recognition must be supplemented with object location measurement before a practical application can be considered.

CHAPTER 6.

EXPERIMENTAL RESULTS

6.1 Introduction

Studies were conducted to assess the performance of the object recognition system and the methodology proposed for the measurement of position and orientation. The first step involved collecting the training data (pattern files for recognition and orientation measurement) for a variety of objects. The objects shown in Figure 5.3 were chosen to represent a variety of sizes and shapes of varying complexity and not be constrained to simple objects. For the method developed, the most important parameter to consider is the object's uniqueness, compared with other objects to be recognized, in developing a network input pattern. Therefore, complicated shapes containing holes, slots, complex contours, etc. should be easier to recognize because they would produce unique patterns. On the other hand, recognizing similar shapes should be more demanding because they require better resolution in order to distinguish their differentiating features.

Before presenting the experimental results, several assumptions were made.

- Objects were situated on a flat surface and have a uniform thickness. They were located completely within the working envelope or, more specifically, within the scanning area of the robot.
- Position misalignments were allowed only in the *XY* plane of the manipulator.
- Rotational misalignments were only about the vertical (*Z*) axis.

- The inherent compliance in the robot's end effector will allow a degree of object misalignment. This reduces the accuracy needed for grasping operations. e.g. a two-fingered gripper will tend to 'align' the object during closure.

6.2 Experimental Set-Up for Network Training

Network training involved presentation of input-output pairs and generating a suitable relationship. Training data should provide an accurate representation of the measured inputs corresponding to the desired output in order to give correct results during the test phase. This was most important for the rotational measurement of an object. For example, if an input training pattern was collected for an object positioned at, say, 45° and the object was actually at 50°, then during the test phase a pattern generated for the object located at 45° would result in an inaccurate network output. Furthermore, more input patterns were needed, compared with recognition training, because the network output must change with small changes in the input pattern corresponding to a new angular position.

Object recognition, on the other hand, required only a small subset of all possible patterns (one $\{x,y\}$ location and several angular positions depending on the symmetry of the part) without regard to precise location. Training data was necessary only at one location, typically the scan field's centroid. Several angular positions were also included because the FVV patterns for each of the objects shown in Figure 5.7 were not identical at each angular position. Therefore, additional positions (as many as one or two) were used to provide a better representation of all possible input patterns that may be collected during subsequent testing.

The distinction between recognition and measurement was also made to reduce the total number of training samples required to properly identify and locate an object. It may be conceivable, from the analysis of the MLP network presented in Chapter 4, that training might involve presenting every possible configuration of each object and developing an accurate transformation in order to identify and calculate position. This would have been very inefficient, requiring extremely long training times, assuming that network training would, indeed, converge. Hence, two separate procedures were used to provide a fully functional object recognition system, each exploiting the power of ANNs in a unique way.

Training values were developed by using the rotary platform, described in section 5.4, to collect edge detection patterns at various positions. When all angular positions had been scanned, an output training file was generated which contained a set of all the input pattern vectors and target output values. This file was used for off-line training of the **bp** network software, outlined in section 4.3.4.

For object recognition, significantly fewer training samples were required for each object and precise angular positioning was not needed. In this case, the target output vector was the same for any location of the object because the goal was rotation and position invariant recognition. Generation of the input pattern vector for recognition training was slower than orientation training because post-processing required the calculation of the feature value vector (FVV) after each scan. The calculation used the method described in section 5.3.

6.3 Recognition Results

The five objects shown in Figure 5.3 were used to test the method of object recognition described in Chapter 5. Their shapes varied in complexity from very simple (square, triangle) to complex (duck, pliers). In order to train the network to recognize these different objects, each object was scanned at various angular positions to provide a representative training subset of all possible configurations. As outlined in section 5.3, a feature value vector (FVV) was calculated for each pattern within the training set. This vector was used as the input pattern to the MLP network.

A MLP network having 91 input units, 20 hidden units and 5 output units (one output unit per object) was used for classification of the objects. The number of input units corresponded to the number of annular sections used to calculate the FVV. The size of the network was determined experimentally and not through any standard procedure. The number of hidden units (HU) had a significant effect on network training because it controlled the rate of convergence. If the number of HU was too small, the network would not converge to an accurate solution. A target output value of 1.0 was used for the output node number corresponding to the object's classification number (1 through 5), shown in Figure 5.3. All remaining target values were set to 0.0. Figure 6.1 illustrates the training results for object identification. This figure indicates that the network converged to an accurate solution within 2500 training cycles. The *tss* parameter, a dimensionless quantity, was a good indicator of convergence although its actual value, which depended on the number of training patterns in the data file, was not important. Training continued until the network produced an output of at least 0.9 for the correct output node and 0.0 for all other nodes for each input pattern. This result indicated a strong classification and a sufficient mapping of the input to the desired output, which was tested in subsequent trials. Object identification rates and strength of recognition are shown in Figure 6.2. Table 6.1 lists several angular positions used with the rotary platform, whereas Table 6.2 gives those angles used for each object during recognition training.

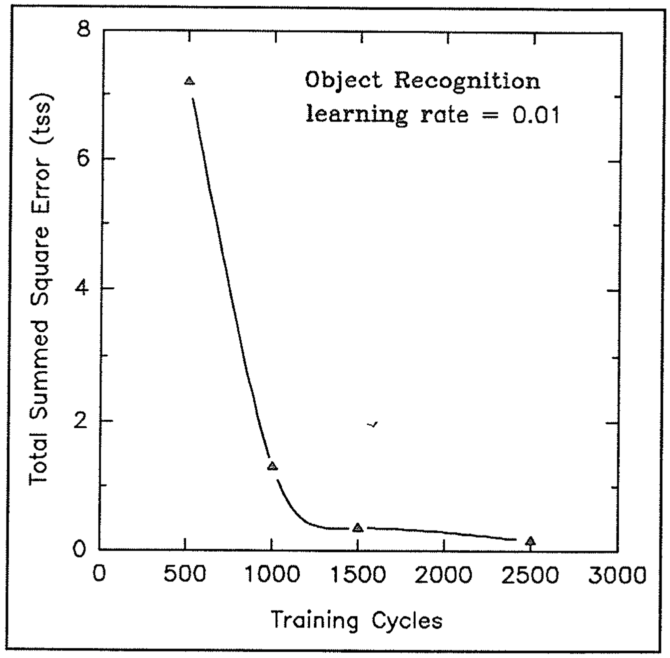


Figure 6.1. Training results for object recognition

Table 6.1. Angular positions used during testing

Angle (° deg)	0	27	45	54	72	81	90	108	135	162	180	225	270	315
Label	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

Table 6.2. Angular positions included in the feature value pattern training set

SHAPE	ANGULAR POSITION LABEL							
1. Duck	A1	A3	A7	A9	A11	A12	A13	A14
2. Square	A1	A2	A3	A5	-	-	-	-
3. Triangle	A1	A2	A4	A6	A8	-	-	-
4. Rectangle	A1	A3	A7	A9	A11			-
5. Pliers	A1	A3	A7	A9	A11	A12	A13	A14

The recognition results shown in Figure 6.2 indicate identification rates for objects placed at random translational and rotational displacements. Identification tests involved positioning each object at five different locations within the scan area and at two angular orientations per position for a total of ten tests per object. The results show that the ANN performed very well for each of the test objects because the identification rates are greater than 99.9 % in all cases. The strength of the identification was defined as the size of the network's output value which provided the correct identification result. Recall that during training, convergence was indicated by a strong value (> 0.90) for the output node corresponding to the object's identification number and 0.0 for all other nodes. Therefore, during testing, correct identification required that the largest value in the output vector $\{o_1, o_2, o_3, o_4, o_5\}$ corresponded to the object being scanned (i.e. if object number 2 is scanned, output o_2 should have the largest value). The average strength of identification was a measure of the size of this value during testing relative to the size of the other output values, averaged over all ten tests. In the case of the triangle, the output node corresponding to the square often showed a slight level of activation (< 0.1) even though the output node corresponding to the triangle shape retained the highest level of activation (and, therefore, a correct identification). This was due to their similar FVV patterns and low image resolution which resulted in poor repeatability during the test phase. If a FVV pattern lay between those shown in Figure 5.7 for the square and triangle, specifically between a radius of 21 and 37 mm, the result was usually misidentification (i.e. the square was mistaken for the triangle and vice-versa). This error in identification can be seen in Figure 6.3, which illustrates the true feature patterns for a square and triangle (as trained) along with feature patterns for a misidentified triangle shapes taken in subsequent tests.

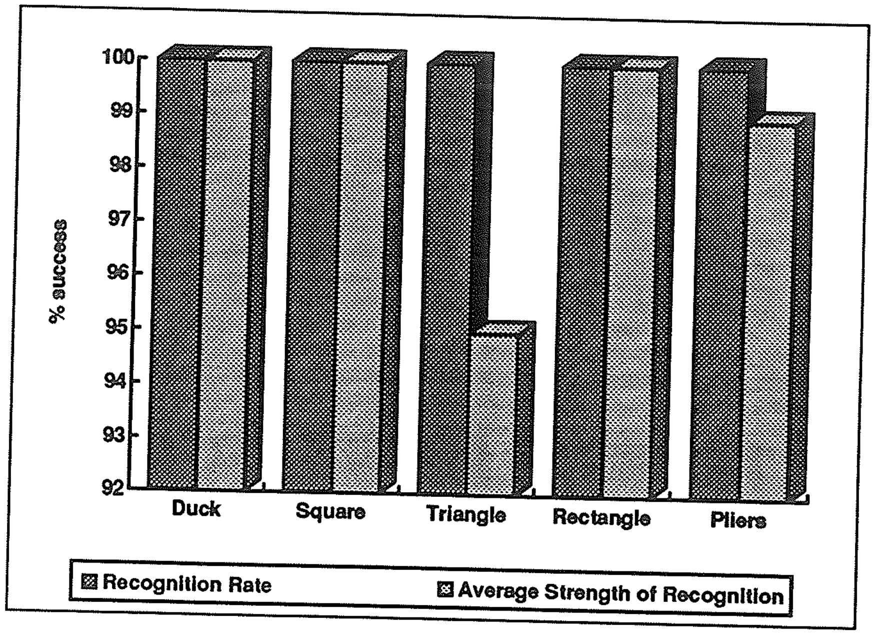


Figure 6.2. Identification rates

Clearly, misidentification was caused by a poor feature pattern, generated after scanning the triangle during a test, and which more closely resembled the square shapes pattern. Although each shape seemed unique to the human eye, their feature values were similar, as shown in Figures 5.7 and 6.3. Their only difference was the point at which the FVV curve value drops off for each shape at approximately $r = 20$ mm. One solution to this problem, other than increasing the line scan image resolution, is a hierarchical identification scheme involving a second modified FVV pattern calculated for similar shapes. For example, when a fourth moment is used in the integration of equation (5.1), the differences between the square and triangle are amplified at increasing radial distances from their centroid. Equation (5.1) would then be in the form:

$$s(r,z) = \int_{D(r)} f(x,y,z)\bar{r}^4 dxdy \quad . \tag{6.1}$$

The resulting FVV patterns for the square and triangle are shown in Figure 6.4.

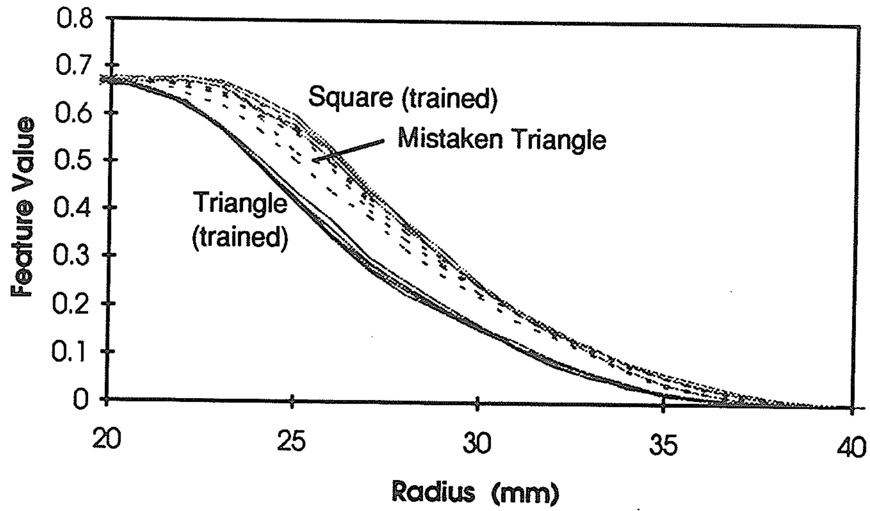


Figure 6.3. Feature value patterns of the square and triangle (extracted from Figure 5.7) including the pattern of a mis-identified triangle taken in later tests

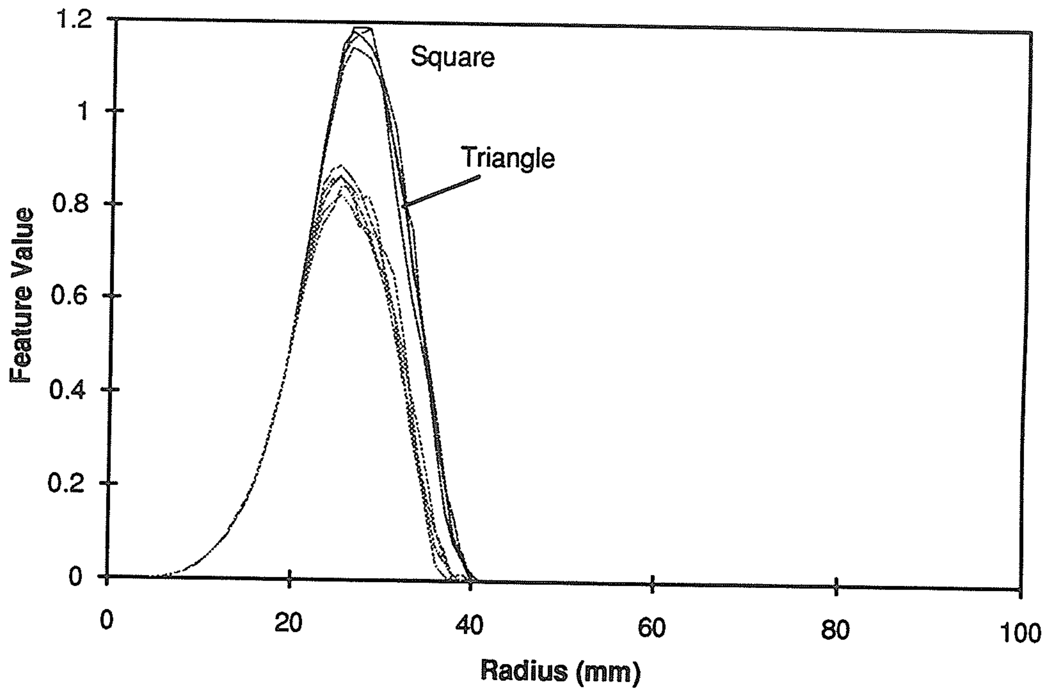


Figure 6.4. Feature value pattern corresponding to a fourth moment calculation for the feature value at several angular positions

It must be stressed that the experimental set up, particularly the mechanical robot arm, suffered from poor repeatability and positional accuracy due to mechanical backlash at the joints. The amount of backlash at the end effector far exceeded the level allowed, as reported in the mechanical robot's service manual. It was very difficult, if not impossible, to retrieve identical FVVs for repeated scans of a stationary object. If the FVV patterns contained a significant amount of 'noise' (pattern fluctuations), then a much more robust network was required which was capable of recognizing patterns having a broad range of input values. In fact, if the 'noise' produced by inaccuracy and low image resolution was enough to create similar input patterns for different objects, the network could very easily mis-identify the object, although it did not in this case. However, if good identification, as shown here, can be achieved with inaccurate or deficient equipment (requiring more training patterns), far better results can be achieved with superior equipment. Commercial, industrial grade, pick-and-place robots would be capable of much better accuracy and repeatability, improving the performance of the method developed in this thesis.

6.4 Measurement of Orientation

The next step involved a second neural network whose output was a measurement of the object's orientation or angular position. The method outlined in Section 5.4 required a second 3 pass scan over the object to develop an input pattern vector. A much smaller input vector (ranging from 4 to 8 input units depending on the object) was used which resulted in a smaller sized and faster network. However, many more training patterns were needed if the network was to produce an accurate output. Early tests proved that the network did not interpolate well between widely trained angular positions (5 and 10°) during the test phase and provided inaccurate measurement of angular position. Therefore, the maximum number of training patterns were collected (every 1.8° in angular position) using the experimental set up described in section 5.4. Furthermore, the target output was not a simple binary value but, rather, a continuously varying one. This posed a more complicated problem than recognition because estimation rather than generalization was required from the network.

Figure 6.5 shows the training results of measurement tests for each of the five objects. All the objects were located with their centroid coincident with the scan area's centroid. Training data included patterns generated through multiple passes of each object in order to teach in a variety of possible pattern sets at each angular position. It was immediately evident that the number of training cycles required for this problem far out numbered that needed for recognition. (See Figure 6.1.) There are two main reasons for this difference. First, the objective was to determine how accurately the network was able to measure orientation. To do this, an input training set was generated to represent the maximum angular range consisting of angular positions separated by the smallest possible increment (1.8 °) to be defined by the stepper motor's resolution. Therefore, asymmetrical objects like the duck and pliers consisted of 200 possible input patterns representing a full 360° range. The rectangle required 100 patterns (180° range), 66 patterns were needed for the triangle (120° range) and 50 patterns for the square (90° range). Each input had a unique target output. Therefore, the result was a large input pattern set with small differences between each pattern and a correspondingly small change between target outputs. This effectively reduced the error gradient computed at each pattern during training and caused slow convergence of the **bp** software. Secondly, poor repeatability of the acquired data (mainly due to mechanical robot inaccuracies) required multiple passes to be included in the training set for each angular position. This added even more input patterns to the training file but offered a more robust network capable of dealing with input pattern 'noise' associated with poor repeatability.

Training was complete when the desired orientation angle and the actual angular output fell within a reasonable tolerance, typically 2 to 5 °. Ideally, training should continue until zero error remains for the training pattern set, which would result in better accuracy during subsequent tests. This requirement was unrealistic in this case because training times would be extremely long. When the actual output of the network converged to within 2 to 5° of the target value during training, the measurement results fell within 2 to 9° of the actual position during subsequent tests. This result is shown in Figure 6.6. The level of accuracy achieved was sufficient for proper gripper manipulation, as outlined in the assumptions given in section 6.1. The total-summed-squared error

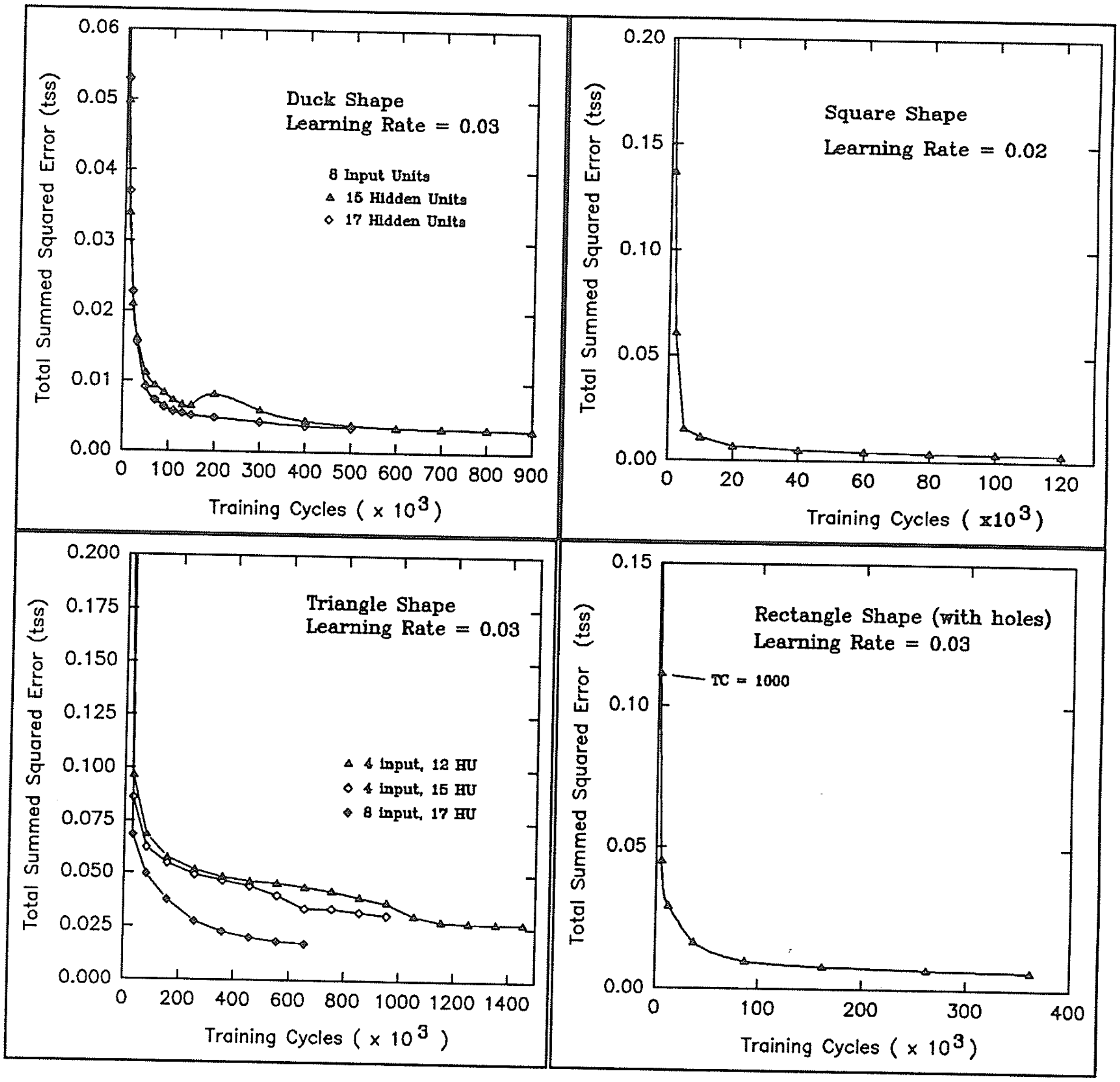


Figure 6.5. Training results for testing the rotational measurement of each object

parameter (*tss*) was a good general indicator of the network's convergence, although a further check was required for each pattern to determine how the *tss* error was distributed. At low *tss* values, only a few patterns were usually responsible for a majority of the error, indicated by the *pss* (pattern-summed-squared error) parameter. Training continued until all network outputs fell within 2 to 5° of the target output. In some cases, better error convergence was achieved (about 2 to 3°) with fewer training cycles, as was the case for the square and triangle. This was a direct function of the number of input patterns used for training.

The training results indicated that the MLP network required a large number of training cycles if it was to be used for accurate orientation measurement. The actual measurement results, given in Figure 6.6, indicated reasonable accuracies, within 2 to 9° in all cases.

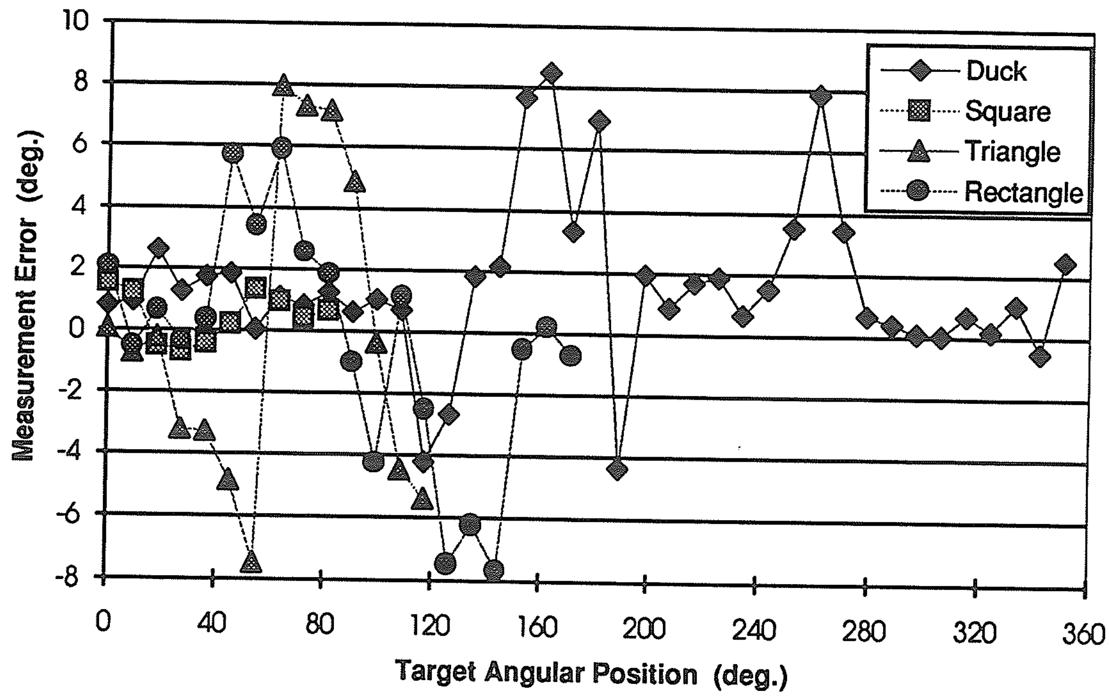


Figure 6.6. Angular position test results

6.5 Summary of Overall Performance

Artificial neural networks provided an effective form of object recognition and position measurement. The simple laser sensor, coupled with the mechanical robot arm, provided an alternative form of 2-D vision that was capable of acquiring image data with several distinct advantages. The method developed was independent of an object's complexity and can be adapted easily to 3-D objects. (The subject of on-going study within the automation laboratory.) In fact, complicated shapes can be learned and identified much more accurately because of their unique FVV patterns. The main disadvantage lies in the training required, which in this case was done off-line. However, the method adapts very easily to new objects, achieved by generating further training. The relatively low image resolution used currently (a function of the number of scans across the object) may lead to problems in identifying very similar parts or ones having only minor differences. More scan passes could be used or, a 2-D, line scan, range finder could be used to retrieve a better 3-D image and provide greater resolution.

The method of angular position measurement was capable of determining orientation by using a simple scan pass, although the robot's mechanical inaccuracies led to long training times and provided relatively poor accuracy. The mechanical positional error, measured at the robot gripper, was as high as ± 4 mm, a direct result of backlash in joints 4 and 5, which resulted in poor repeatability. Furthermore, backlash tended to induce vibrations at the end effector during a scan. This method relied on measuring and comparing edge detection time histories during the scan, and, as a result, poor repeatability tended to distort these patterns so that multiple passes were required for the training set. Using multiple passes in the training set provided a better representation of the patterns that would most likely be retrieved during testing. Although accuracy was improved, long training times were required. These mechanical problems are being addressed currently in the expectation of improving the reliability of the method.

The measurement of an object's orientation proved to be a more demanding problem than simple object recognition. The network was trained to classify similar patterns into distinct output regions representing each angular position. This problem is one of *estimation* rather than *classification* [37] because the output values are used as a measurement tool. In supervised learning methods, such as back-propagation, the power lies in the ability to recognize noisy or distorted patterns and classify the input according to previously trained data. For pure object recognition, only classification is important and the outputs take a value of zero or one. However, to provide accurate estimation capabilities for the measurement of orientation , the output still lies within the range of 0 to 1 but its specific value is related directly to a physical measurement of angular position. This type of output requires many training samples and longer training times in order to improve the generalization property of the network [37].

The tests performed here assumed random orientations and positions of objects, and, hence, required the largest training set possible. If additional constraints were imposed, for example, $\pm 45°$ angular misalignment range only, the method would be far more effective because the training set would be much smaller. Depending on the application, additional constraints may be imposed with simple fixturing or by the very nature of the process.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

Using a real-time, multi-tasking operating system, a flexible robot control system has been developed and tested. A standard PC microcomputer was shown to be capable of controlling a 5 DOF robot manipulator by using modular 'C' based software and two custom developed, motion control interface cards. The software performed Cartesian path control, individual PD control compensation, on-line gain adjustment, individual jog profiles, and included a keyboard-based, teach-mode used to develop robot paths. The modular architecture allowed straightforward implementation of additional tasks - including an artificial neural network interface, external stepper motor control, standard I/O interface and D-to-A hardware communication. The overall design is ideal for integration into an automated work cell where real-time communication with several processes is crucial. Furthermore, the whole system can be implemented, without modification, on a system wide, Ethernet or Arcnet based network to provide a means for centralized control and monitoring.

Using this control system, an object recognition and measurement system was developed by using a simple 1-D, laser range finder. A multi-layered neural network employing back propagation learning was trained off-line to provide the robot with the ability to recognize a set of objects and, subsequently, measure their position and orientation. High object identification rates of not less than 99 % were achieved in most cases, completely independent of position and orientation of the object. The recognition method was capable of identifying randomly located parts within the robot's scan area by using on-line neural network access, assuming these parts were included previously in the training set. Translational displacement measurement, by using recognition scan data, provided measurements within ± 5 mm of an object's true centroid location. An automated method of training the network was developed to simplify the process of introducing new objects into the model base.

Limited success was achieved with the orientation measurement method. Inaccuracies inherent to the mechanical system complicated the training of the neural network and significantly decreased the rate of convergence of the back propagation software. Rotational accuracies of 2 to 9° were achieved for orientation measurements alone by employing a simple 3 pass scan over the object. Better success can be achieved if the repeatability and accuracy of the measurement system (specifically the mechanical robot arm) is improved.

The method effectively demonstrates the use of ANNs in a practical application involving a simple laser sensor. The advantages of using this type of sensor include: (i) true depth information of the scene to provide a more useful image for robotic environments, (ii) lighting conditions have no effect, (iii) the sensor's lightweight and compact size allows direct mounting to a robot's end-effector, and (iv) simple data acquisition interface requiring only standard A-to-D hardware. ANNs provide an effective means of interpreting the sensor data without regard to an object's complexity. Processing time is independent of the object's shape and new parts can be included quickly.

7.2 Recommendations

7.2.1 Robot Controller

The control system is limited currently to a relatively low sampling frequency of 250 Hz both through hardware and indirectly, through software constraints. The control software is completely responsible for closing the position loop, utilizing the resources of a single CPU. An IBM PC 486DX, 33 MHz machine was capable of performing this task in real time with sufficient idle time to perform other tasks. However, this sampling rate resulted in borderline stability for two of the axes and should be increased. This would result in an increased damping ratio and better overall performance. Similar commercial robots in this class sample joint positions at rates up to1 KHz [15]. If the present MCI card was modified to increase its encoder sampling rate, the hardware capability of the current computer may still be inadequate to perform the control task in real time. Two solutions exist. First, improve the performance of the microcomputer by simply upgrading to a 50 MHz machine or adding a 66 MHz doubling chip optimizer. This would still require a modification of the current MCI card hardwired for a 250 Hz sampling speed. Secondly, enhance the function of the MCI card to provide processing capability directly on-board. This second solution is underway currently with the design and development of a new MCI card capable of sampling at speeds up to 7.812 KHz under its own control [21]. This will relieve the host computer of time-intensive control functions and enable such features as on-line neural network training to be feasible.

The robot's control software requires a Robot Programming Language (RPL) interface if it is to be used in a commercial environment where operators have limited programming capability. Presently, programming skill in 'C' is required to develop a typical assembly or pick and place routine. The RPL will simplify the way in which the robot is controlled through simple commands [18].

7.2.2 Object Recognition and Measurement

One disadvantage of the proposed method is the slow identification and measurement speed. It currently takes 45 seconds to complete the recognition scan (98 scans across the scan field) and another 19 seconds to develop the FVV input pattern and retrieve a network output. The path scan time can be reduced with the aid of the improved MCI card. If the robot moved faster across the part, fewer data points would be sampled resulting in a loss of edge detection accuracy. More importantly, the path speed is limited by the mechanical capability of the robot arm. Faster speeds resulted in vibration and jerk at transition points in the zig-zag path which will contribute to long term wear and poor accuracy of the robot links. Of course, a different type of sensor capable of scanning under its own control would alleviate this problem. The remaining time (19 s) is related to the processing power of the microcomputer. The FVV calculation and network interface is computationally expensive and must be executed at a low priority which causes long execution times. If faster hardware was used, this time could be minimized.

The measurement scan path, requiring 3 scans across the object, also must be executed slowly due to hardware limitations. This path required 14 seconds to complete followed by a much faster computation of the input pattern and network access requiring less than 1 second. The A-to-D interface card is not capable of performing DMA (Direct Memory Access) transfers without software intervention. Therefore, another software task was required to sample the laser output during the scan. This could not be done faster than 250 Hz. The goal was to accurately measure edge detection time histories. Therefore, the path was slowed to allow accurate laser output measurement. If true DMA was available, the control system would be relieved of the A-to-D sampling burden and the scan path could be performed much faster by using a correspondingly higher sampling rate.

REFERENCES

[1] Miller, W.T., III, "Sensor-Based Control of Robotic Manipulators using a General Learning Algorithm", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 2, pp. 157-165, April, 1987.

[2] Miller, W.T., III, "Real-Time Application of Neural Networks for Sensor-Based Control of Robots with Vision", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 19, No. 4, pp. 825-831, July/August, 1989.

[3] Watanabe, S. and Yoneyama, M., "An Ultrasonic Visual Sensor using a Neural Network and its Application for Automatic Object Recognition", Proc. of IEEE Ultrasonics Symposium, pp. 781-784, 1991.

[4] Lippmann, R.P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-22, April, 1987.

[5] Gadsby, C.G., "A Microcomputer Based Machine Controller for Contouring Applications ", M.Sc. Thesis, University of Manitoba, Winnipeg, Manitoba, December, 1986.

[6] Kostyniuk, T.M., " Development of a Flexible, Microcomputer Based, Three Axis Machine Tool Controller ", M.Sc. Thesis, University of Manitoba, Winnipeg, Manitoba, August, 1988.

[7] Toutant, R.P. ," Cutting Force Adaptive Control for Turning ", M.Sc. Thesis, University of Manitoba, Winnipeg, Manitoba, 1989.

[8] McClelland, J.L. and Rumelhart, D.E., Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises. The MIT Press, Cambridge, Massachusetts, 1988.

[9] Goldenberg, A.A. and Chan, L., "An Approach to Real Time Control of Robots in Task Space. Application to Control of PUMA 560 without VAL II", IEEE Transactions on Industrial Electronics, Vol. 35, No. 2, pp. 231-238, May, 1988.

[10] Robot Sensors. Vol. 1 - Vision. International Trends in Manufacturing Technology series, Pugh, A. ed., IFS (Publications) Ltd., Bedford, UK, 1986.

[11] Kabuka, M.R., Glaskowsky, P.N. and Miranda, J., " Micro-Controller Based Architecture for Control of a Six Joints Robot Arm", IEEE Transactions on Industrial Electronics, Vol. 35, No. 2, pp. 217-221, May, 1988.

[12] Whitcomb, L.L. and Koditschek, D.E., " Robot Control in a Message Passing Environment: Theoretical Questions and Preliminary Experiments", IEEE International Conference on Robotics and Automation, pp. 1198- 1203, 1990.

[13] DC Motors. Speed Controls. Servo Systems. 5th Edition, Electro-Craft Corporation, Hopkins, Minnesota, July, 1980.

[14] Van De Vegte, J., Feedback Control Systems. 2nd Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[15] CRS SRS-M1A Industrial Robot System Operation Manual. CRS Plus Inc., Burlington, Ontario, 1987.

[16] Electro-Craft Servo Systems Handbook. Robbins & Myers/Electro-Craft, Eden Prairie, Minnesota, 1992.

[17] Dorf, R.C., Modern Control Systems. 2nd Edition, Addison-Wesley Publishing Co., Reading, Massachusetts, 1974.

[18] Paul, R.P., Robot Manipulators: Mathematics. Programming. and Control. The MIT Press, Cambridge, Massachusetts, 1981.

[19] Craig, J.J., Introduction to Robotics: Mechanics and Control. 2nd Edition, Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.

[20] MAX 100 PWM Servo Drive Instruction Manual. Document 0013-1015-001 Rev A, Reliance Electric Co., 1992.

[21] Tyc, R., "The Development of a Servo-Motor Interface Card using a New High Performance CMOS Motion Control IC", Internal Report, Department of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Manitoba, October, 1993.

[22] Horn, B.K.P., Robot Vision. The MIT Press, Cambridge, Massachusetts, 1986.

[23] Fairhurst, M.C., Computer Vision for Robotic Systems: An Introduction. Prentice-Hall, Hertfordshire, UK, 1988.

[24] Laser Analog Displacement Sensor: Model LAS-8010 Technical Manual. Adsens Tech. Inc., La Puente, CA.

[25] Hush, D.R. and Horne, B.G., "Progress in Supervised Neural Networks", IEEE Signal Processing Magazine, pp. 8-39, January, 1993.

[26] Nagy, G., "Neural Networks - Then and Now", Letter to IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 316-318, March 1991.

[27] Leahy, M.B., Jr., Johnson, M.A. and Rogers, S.K., "Neural Network Payload Estimation for Adaptive Robot Control", IEEE Trans. on Neural Networks, Vol. 2, No. 1, pp. 93-100, January, 1991.

[28] Chen, N. and Chung, H., " Robot Path Planner: A Neural Networks Approach", Proc. of the 1992 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 548 - 553, July, 1992.

[29] Hush, D.R., Horne, B. and Salas, J.M., "Error Surfaces for Multilayer Perceptrons", IEEE Trans. on Sys., Man, and Cyber., Vol. 22, No. 5, September/October, 1992.

[30] Hush, D.R., "Classification with Neural Networks: A Performance Analysis", Proc. of the IEEE Intl. Conf. on Systems Eng., pp. 277-280, 1989.

[31] Werbos, P.J., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", Doctoral Dissertation, Applied Mathematics, Harvard University, Boston, MA, November 1974.

[32] Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", Psychological Review, 65:386-408, 1958.

[33] Huang, Shih-Chi and Huang, Yih-Fang, "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons", IEEE Trans. on Neural Networks, Vol. 2, No. 1, January, 1991.

[34] Van Ooyen, A. and Nienhuis, B., "Improving the Convergence of the Back-Propagation Algorithm", Neural Networks,Vol. 5, pp. 465-471, 1992.

[35] Hush, D.R. and Salas, J.M., "Improving the Learning Rate of Back-Propagation with the Gradient Reuse Algorithm", Proc. of the IEEE Intl. Conf. on Neural Networks, Vol. 1, pp. 441-448, 1988.

[36] Beyer, W.H., ed., CRC Standard Mathematical Tables, 28th edition, CRC Press Inc., Boca Raton, FL , 1987.

[37] Pao, Yoh-Han, Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, MA, 1989.

[38] QNX 4.0 Operating System User's Guide. Quantum Software Systems Ltd., Kanata, Ontario, April, 1991.

[39] Berns, K., Dillmann, R. and Hofstetter, R., "An Application of a Backpropagation Network for the Control of a Tracking Behaviour", Proc. of the IEEE Intl. Conf. on Rob. and Auto., pp. 2426-2431, April, 1991.

[40] Allen, P.K., Robotic Object Recognition Using Vision and Touch. Kluwer Academic Publishers, Boston, MA, 1987.

[41] Nevatia, R., Machine Perception. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[42] Carpenter, G.A. and Grossberg, S., ed., Pattern Recognition by Self-Organizing Neural Networks. The MIT Press, Cambridge, MA, 1991.