

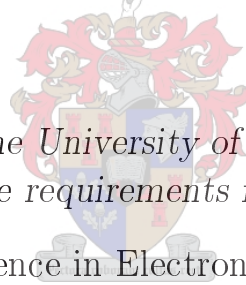


UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvenoot • your knowledge partner

Object Recognition and Automatic Selection in a Robotic Sorting Cell

by

Frederick Johannes Janse van Rensburg



*Thesis presented at the University of Stellenbosch in partial
fulfilment of the requirements for the degree of*

Master of Science in Electronic Engineering

Department of Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisors:
Mr J. Treurnicht & Prof. C.J. Fourie

October 2006

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

F.J. Janse van Rensburg

Date:



Abstract

Object Recognition and Automatic Selection in a Robotic Sorting Cell

F.J. Janse van Rensburg

Department of Electronic Engineering

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Electronic)

October 2006

This thesis relates to the development of an automated sorting cell as part of a flexible manufacturing line, with the use of object recognition. Algorithms for each of the individual subsections creating the cell, recognition, position calculation and robot integration were developed and tested.

The Fourier descriptors object recognition technique is investigated and used. Invariance to scale, rotation or translation of the boundary of an object makes this technique very favourable for this application of object recognition. Stereoscopy with basic trigonometry is used to calculate the position of recognised objects, after which they are handled by a robot. Integration of the robot into the project environment is done with trigonometry as well as Euler angles.

It is shown that a successful, automated sorting cell can be constructed with object recognition. The results show that reliable sorting can be done with available hardware and the algorithms developed.

Uittreksel

Voorwerpherkenning en outomatiese seleksie in 'n Robotiese Sorteringsel

(“Object Recognition and Automatic Selection in a Robotic Sorting Cell”)

F.J. Janse van Rensburg

Departement Elektroniese Ingenieurswese

Universiteit van Stellenbosch

Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (Elektronies)

Oktober 2006

Hierdie tesis handel oor die ontwikkeling van 'n geoutomatiseerde sorteringsel as deel van 'n aanpasbare produksielyn, met die hulp van objekherkenning. Algoritmes vir die verskillende probleme, herkenning, posisie bepaling en robot-integrasie word ontwikkel en getoets.

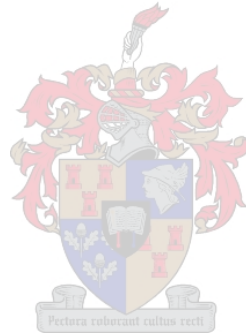
Fourier-beskrywers as herkenningsmeganisme word ondersoek en gebruik. Invariansie tot skaal, rotasie en translasie van die buitelyn van 'n objek maak hierdie meganisme gunstig vir gebruik in 'n sorteringsel. Stereoskopie en basiese trigonometrie word gebruik vir posisie bepaling van objekte in die sorteringsel. Integrasie van die robot geskied met die hulp van trigonometrie en Euler hoeke.

Daar word getoon dat 'n suksesvolle, outomatiese sorteringsel met die hulp van objekherkenning gebou kan word. Resultate toon die akkuraatheid van die stelsel is genoegsaam vir betroubare sortering.

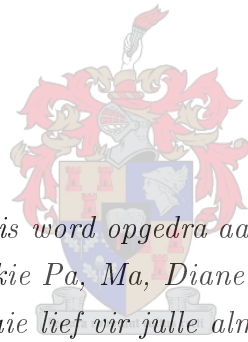
Acknowledgements

I would like to express my sincere gratitude to the following people who have contributed to making this work possible:

- Mr J. Treurnicht of the Electronic Engineering Department at the University of Stellenbosch as my supervisor.
- Prof C.J. Fourie of the Industrial Engineering Department at the University of Stellenbosch as my supervisor.



Dedications



Hierdie tesis word opgedra aan my gesin.

Baie dankie Pa, Ma, Diane en Nadia.

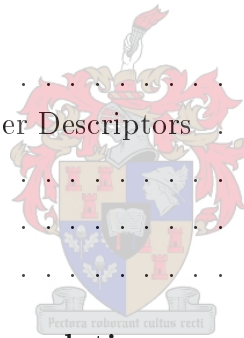
Baie lief vir julle almal.

Contents

Declaration	ii
Abstract	iii
Uittreksel	iv
Acknowledgements	v
Dedications	vi
Contents	vii
List of Figures	ix
List of Tables	xiii
Tradenames	xv
1 Introduction and Overview	1
1.1 Motivation	1
1.2 Background	2
1.3 Objectives of this Study	2
1.4 Dissertation Structure	3
2 Related Work	4
2.1 Image Recognition	4
2.2 Robotics	5



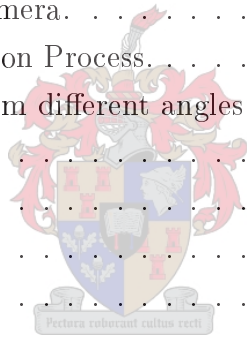
3	Object Recognition	15
3.1	Object Recognition in Flexible Manufacturing	15
3.2	Algorithm Requirements	17
3.3	Fourier Descriptors	18
4	Robotics	26
4.1	RTX Geometry	26
4.2	Robot Kinematics	27
5	System Integration	36
5.1	Hardware and Function Allocation	36
5.2	The Automated Sorting Cell	37
5.3	Hardware	40
5.4	Recognition	43
5.5	Handling	63
6	Results	67
6.1	Guard Algorithm	67
6.2	Recognition with Fourier Descriptors	69
6.3	Colour Identification	75
6.4	Position Calculation	75
6.5	Handling	77
7	Conclusions and Recommendations	80
7.1	Conclusions	80
7.2	Recommendations	82
	List of References	84
	Appendices	86



List of Figures

2.1	The RTX robot assembly. (Adapted from [1].)	6
2.2	Moving the wrist outwards along a straight line between the shoulder and wrist spindles.	7
2.3	Range of movement and dimensions of the RTX robot. (Adapted from [1].)	8
2.4	Denavit-Hartenberg Parameters for a generic joint. (Adapted from [2].)	13
3.1	A digital boundary and its representation as a complex sequence. The points $(x_0, y_0), (x_1, y_1)$ shown are (arbitrarily) the first two points in the sequence. (Adapted from [3].)	19
3.2	Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction of the boundary. (Adapted for [3].)	20
3.3	Boundary, (a), and its Fourier descriptors, (b).	22
	(a)	22
	(b)	22
3.4	The approximation process featuring vectors related to the two lowest frequency components.	23
3.5	Approximating a square wave with different amount of sine waves.	24
	(a) Using only 1 term	24
	(b) Using 3 terms	24
	(c) Using 7 terms	24
4.1	Cartesian coordinates and rotations defined for the RTX robot.	27
4.2	Range of movement of the RTX robot. (Adapted from [1].)	31
4.3	Top view of RTX robot with shoulder and elbow.	32

4.4	Contributions of the wrist to the x, y and z position to the end position of the robot arm.	33
	(a) Side view of wrist showing the contribution of the pitch to x and z. (Adapted from [4].)	33
	(b) Top view of wrist showing the contribution of the yaw to x and y. (Adapted from [4].)	33
4.5	Using the law of cosines to find the shoulder angle, θ_s , and elbow angle, θ_e	34
5.1	Hardware and Function Allocation.	36
5.2	Setup of the Sorting Cell Environment.	37
5.3	Photo of the sorting cell environment.	38
5.4	Flow diagram of the Sorting Process.	39
5.5	Hardware connection for the automated sorting cell.	40
5.6	Velocity profile for each of the motors. Adapted from [1].	42
5.7	AVC526LN Mini CCD Camera.	42
5.8	Flow diagram of Recognition Process.	44
5.9	Pieces to be recognised from different angles.	45
	(a)	45
	(b)	45
	(c)	45
	(d)	45
	(e)	45
	(f)	45
	(g)	45
	(h)	45
	(i)	45
	(j)	45
	(k)	45
5.10	Object boundary with arrows showing the longest and shortest distances from the centre to the boundary.	46
5.11	Valid regions with sample points of width vs. shortest distance to centre.	47
5.12	Full Fourier descriptors sets of the king, (a), and subset containing 40 lowest frequency components, (b).	48



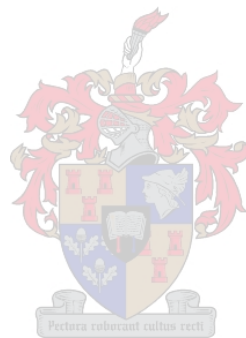
(a)	48
(b)	48
5.13 Fourier descriptors of a bishop's profile.	49
(a) Bishop (angle 1) and its 20 lowest frequency descriptors.	49
5.14 Fourier descriptors of a bishop's, castle's and king's profiles from different angles.	50
(a) Bishop (angle 2) and its 20 lowest frequency descriptors.	50
(b) Castle (angle 1) and its 20 lowest frequency descriptors.	50
(c) Castle (angle 2) and its 20 lowest frequency descriptors.	50
(d) King (angle 1) and its 20 lowest frequency descriptors.	50
5.15 Fourier descriptors of the king's, rook's and pawn's profiles from different angles.	51
(a) King (angle 2) and its 20 lowest frequency descriptors.	51
(b) Rook (angle 1) and its 20 lowest frequency descriptors.	51
(c) Rook (angle 2) and its 20 lowest frequency descriptors.	51
(d) Pawn and its 20 lowest frequency descriptors.	51
5.16 Fourier descriptors of the queen's profile from different angles.	52
(a) Queen (angle 1) and its 20 lowest frequency descriptors.	52
(b) Queen (angle 2) and its 20 lowest frequency descriptors.	52
5.17 Comparison between the boundaries of the first rook and the pawn.	54
5.18 Object for comparison, (a), and its resulting boundary (b).	55
(a)	55
(b)	55
5.19 The 20 lowest frequency components of the example queen's boundary.	56
5.20 Two chess pieces in the same line of sight, (a), and the resulting boundary (b).	57
(a)	57
(b)	57
5.21 Differences in Fourier descriptors.	57
(a) Difference between Fourier coefficients of a queen-queen comparison.	57
(b) Difference between Fourier coefficients of the queen-combined boundary comparison.	57
5.22 Chess pieces with different colours, a black king, (a), and white queen (b).	58

(a)	58
(b)	58
5.23	Environment setup for position calculations.	59
5.24	Position errors due to errors in the extracted angle.	61
(a)	Error in x-position for Δ_1 ranging from 0° to 50°	61
(b)	Error in y-position for Δ_1 ranging from 0° to 50°	61
5.25	Position errors due to errors in the extracted angle.	61
(a)	Error in x-position for Δ_1 ranging from 0° to 5°	61
(b)	Error in y-position for Δ_1 ranging from 0° to 5°	61
5.26	Illustrating limits of stereovision algorithm.	62
5.27	Different solutions to the inverse kinematic equation.	63
(a)	63
(b)	63
5.28	Manipulator pose in sorted area.	64
5.29	Elbow and wrist for handling strategy 1 with pitch at 0°	65
5.30	Elbow and wrist for handling strategies with pitch at -60°	66
5.31	Movement of the robot arm from the sorting to sorted area.	66
6.1	Image of unrecognised king angle.	71
6.2	Image and resulting boundary with overhead lighting, (a) and (b) and with added directional lighting, (c) and (d).	73
(a)	73
(b)	73
(c)	73
(d)	73
6.3	Handling an object behind another.	79
(a)	Handling viewed from the front.	79
(b)	Handling viewed from the side.	79

List of Tables

2.1	Ranges of travel for the arm axes from the initialised position. (Adapted from [1].)	9
3.1	Basic Fourier descriptor properties. (Adapted from [3].)	21
5.1	Constants for converting distances and angles to motor encoder counts.	41
5.2	Valid regions for dimension properties to fall between to belong to the chess piece class.	46
5.3	Table of valid regions with centre point and radius of each region.	46
5.4	Calibration objects Fourier Descriptor comparison. Smallest difference to object of another type in bold.	53
5.5	Thresholds for each of the calibration boundaries.	55
5.6	Difference between descriptors of the test object and each of the calibration objects.	56
6.1	Results for the some of the chess piece profiles.	68
6.2	Attributes for object in non ideal conditions.	68
6.3	Attributes for invalid objects.	69
6.4	Fourier descriptor coefficient differences for test objects under normal conditions. Smallest difference in bold.	70
6.5	Thresholds for each of the calibration boundaries.	71
6.6	Fourier descriptor coefficient differences for sorting area under directed lighting. Smallest difference in bold.	74
6.7	Colour identification results.	75
6.8	Position algorithm results and accuracy.	76
6.9	Handling accuracy results.	77

6.10 Sorting time for each of the strategies. 78

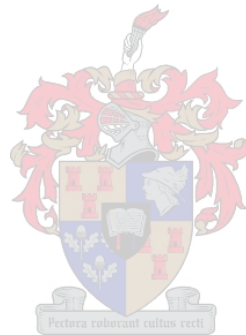


Tradenames

MATLAB is a registered trademark of The Mathworks, Inc.

JAVA is a registered trademark of Sun Microsystems, Inc.

MSDOS is a registered trademark of Microsoft, Inc.



Chapter 1

Introduction and Overview

1.1 Motivation

Over the years, rules governing competition in manufacturing have grown more stringent and are forcing manufacturers to enhance their efforts to improve competitiveness. Intelligent manufacturing through innovation and optimisation is the key to improved competitiveness. The sorting cell in the modern flexible manufacturing line can be optimised and improved with the addition of intelligence by means of intelligent object recognition.

Existing sorting methods used in manufacturing lines rely on object properties such as size or attached barcodes, but situations do arise where these methods are useless. Using object recognition as identification and sorting technique, with the help of a robotic arm, allows for more versatility in the sorting cell and overcomes the constraints of existing methods. The ability to detect some defects in products will be an added feature to the sorting cell as a side effect of the object recognition.

The addition of intelligence in the sorting cell is the next logical step in the evolution of intelligent and flexible manufacturing lines. The challenge is to perform this evolution at the lowest cost possible using of the shelf components and object recognition techniques which are robust, accurate and reliable.

1.2 Background

The Centre for Robotics (SENROB) at the University of Stellenbosch forms part of the Industrial Engineering Department. Research at SENROB is primarily based on the increase of productivity of manufacturing lines through automation and optimisation. Existing methods and hardware are evaluated and new methods are developed. The aim of the centre is to have a fully automated manufacturing line simulated within the lab.

The sorting cell forms an integral part of the fully automated manufacturing line. Having an intelligent sorting cell as part of the simulated manufacturing line will allow SENROB to investigate this area of manufacturing. Knowledge of the requirements placed on the rest of the manufacturing line, as well as the communication protocol between cells on the line, will be gained when the intelligent sorting cell is fully integrated into the simulated manufacturing line.

1.3 Objectives of this Study

The objective of this project is the investigation into and development of an intelligent, automated sorting cell in the manufacturing line using object recognition as sorting method. Firstly, the study involves the development and evaluation of an object recognition algorithm using the Fourier descriptors technique. Fourier descriptors use the frequency information stored in the boundary of an object to distinguish between different objects.

The second part of the study is concerned with the integration of the object recognition algorithm with a robotic arm which will handle identified objects. The performance of the object recognition algorithm, as well as a position calculation algorithm, must be evaluated when used with the supplied RTX robotic arm. Accuracy and reliability are considered more important than speed. Evaluation will be focused on these aspects.

1.4 Dissertation Structure

This thesis is structured as follows:

- Chapter 1 motivates the thesis, gives some background and an overview of the scope of the work.
- Chapter 2 discusses work related to this project.
- Chapter 3 introduces the object recognition theory.
- Chapter 4 contains the robotics theory needed for the project.
- Chapter 5 discusses the integration of the separate fields into a working automated sorting cell.
- Chapter 6 presents results from the evaluation of the developed automated sorting cell.
- Chapter 7 contains the conclusions and recommendations achieved from this project.



Chapter 2

Related Work

2.1 Image Recognition

2.1.1 Digital Image Recognition for Machine Intelligence

In 2000 a final-year project by J. Joubert [5] was the first to investigate the use of machine vision in SENROB. The goal of the project was to develop an image recognition program that can classify simple shapes in images stored in the JPEG format. Basic, free-standing shapes could be identified by comparing the pixel data in each of the images. Two different techniques of comparing the pixel data of an image were tested. Objects to be identified had to be precisely placed in front of the camera at a pre-defined spot. If an object was correctly identified a Chang-Shing Electric CS113 type robot was used to handle the object. The robot was controlled by its own computer and no interaction between the image recognition program and the robot took place. Instructions for the robot for each of the objects were programmed beforehand and when identification took place the user had to notify the robot which object it was going to handle. Although none of the image recognition techniques in this report were used, it gave an idea of what features and abilities a good object recognition algorithm should have.

2.1.2 Digital Image Recognition for Robot Vision

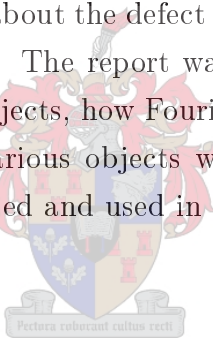
Following J. Joubert's investigation W.A. Joubert [6] looked into image recognition for robot vision the following year for his final year project. His goal was to develop

recognition software that can be used in conjunction with the RTX robot in SEN-ROB. The project was done in Delphi with the goal of merging it with a program developed by André Albers the previous year, controlling the RTX robot using a Delphi program. Once again comparison between images took place by comparing individual pixels to one another. This technique is very limiting but the project helped in the understanding of problems others have encountered and limitations their methods had.

2.1.3 Shape description with Fourier descriptors

A technical report by Petković and Krapac [7] found that defects in products can be found using object recognition based on shape. The Fourier descriptor technique was used to extract frequency information from the boundaries of objects, which was then used to compare correct and defective objects with each other. The possibility exists to enhance the algorithm to distinguish between various types of defects, thus obtaining information about the defect which can be used for automated control of the production process. The report was informative on the subjects of illumination, the descriptors of objects, how Fourier descriptors are calculated and how to compare descriptors of various objects with each other. Techniques discussed in the report are investigated and used in chapters 3 and 5.

2.2 Robotics



2.2.1 RTX Robot

The RTX robot manuals [1] contain information about the physical, electrical, electronic and software properties of the robot. Part 1 of the manual contains basic information such as how to set up the robot and run a demonstration program using the software originally supplied with the robot. Part 2 focuses on the physical attributes of the robot. Design information shows the dimensions of the RTX and the geometry of the robotic arm. The control and effect of each electric motor are described, as well as the electronics controlling them. Maintenance information on the robot is found in part 4 of the manual and the IPC protocol for communication

with the robot is explained in part 5. Some of the most important attributes of the RTX robot needs to be highlighted and are summarised from the manuals.

RTX is a robot arm designed by Universal Machine Intelligence (UMI) to be controlled from a personal computer. It is connected to the controlling computer via a serial interface. Communications between RTX is handled by a protocol called Intelligent Peripherals Communications (IPC), which has been developed at UMI specifically for this purpose. The assembly of the whole robot is shown in figure 2.1. Moving components includes the linear slideway, the upper and lower arm, the

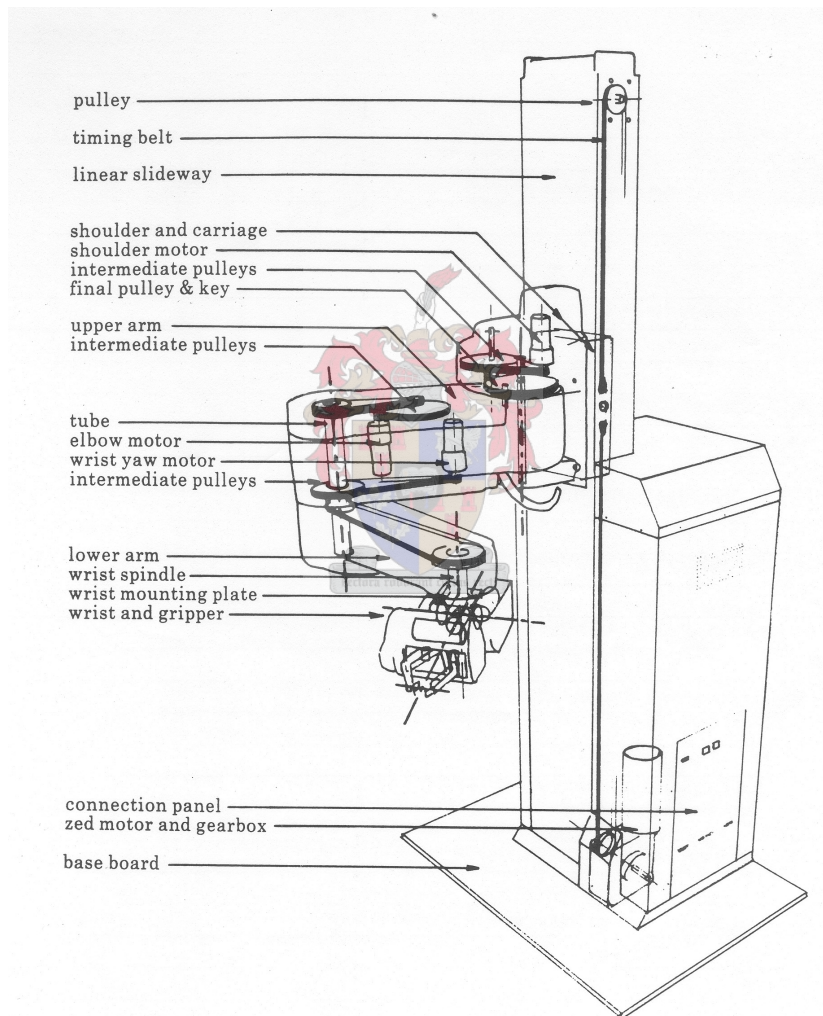


Figure 2.1: The RTX robot assembly. (Adapted from [1].)

wrist and the gripper. The linear slideway is driven by the z-axis motor and moves the whole robotic arm up and down along the tower of the robot. The upper arm is connected to the linear slideway at the shoulder. The shoulder holds a vertical spindle which lets the upper arm swing around the shoulder joint in a horizontal plane on its bearings when driven by the shoulder motor. The lower arm is fixed to a tube that rotates on bearings in the upper arm, allowing it to swing around the elbow joint in a horizontal plane when driven by the elbow motor. The wrist itself is connected to the rest of the arm at the end of the lower arm. It is capable of swinging in the horizontal plane around the wrist spindle when driven by the wrist yaw motor. The gripper of the wrist can perform a pitch rotation around the gripper joint and a roll rotation around an axis perpendicular to the gripper joint and parallel to the lower arm.

As the upper and lower arm are the same length, it means that the wrist can be moved in a straight line outwards from the column, along a radial line between

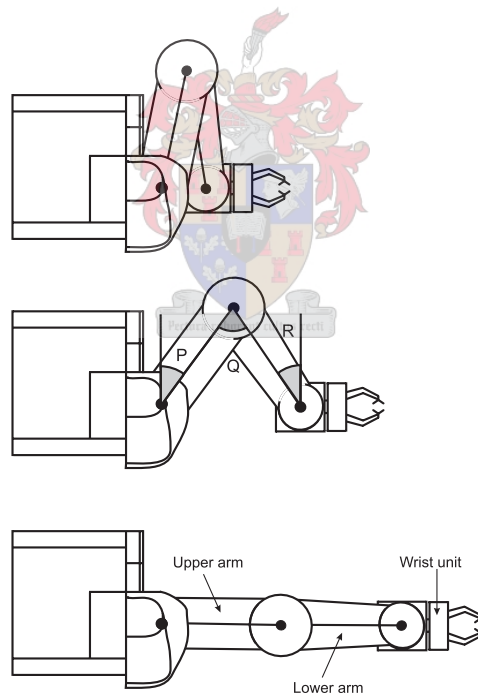


Figure 2.2: Moving the wrist outwards along a straight line between the shoulder and wrist spindles.

the shoulder and wrist spindles, by rotating the two parts of the arm, making sure that angle P is always half angle Q in figure 2.2. The gear ration from the shoulder to the upper arm is twice that of the elbow motor to the lower arm. To move the wrist in a radial line, you therefore drive the two motors at the same speed but in opposite directions. In addition, the gripper can be kept pointing along the radial line by keeping angle R the same as angle P . This is achieved automatically without needing to drive the yaw motor. When the lower arm moves through an angle Q , the wrist automatically moves through R , which is $Q/2$, because of the 2:1 gear ratio from the combined pulley which rotates on the elbow spindle and the wrist pulley.

The range of movement and the dimensions of the robot are shown in figure 2.3. When the arm is initialised, it should be calibrated with the arm positioned as in

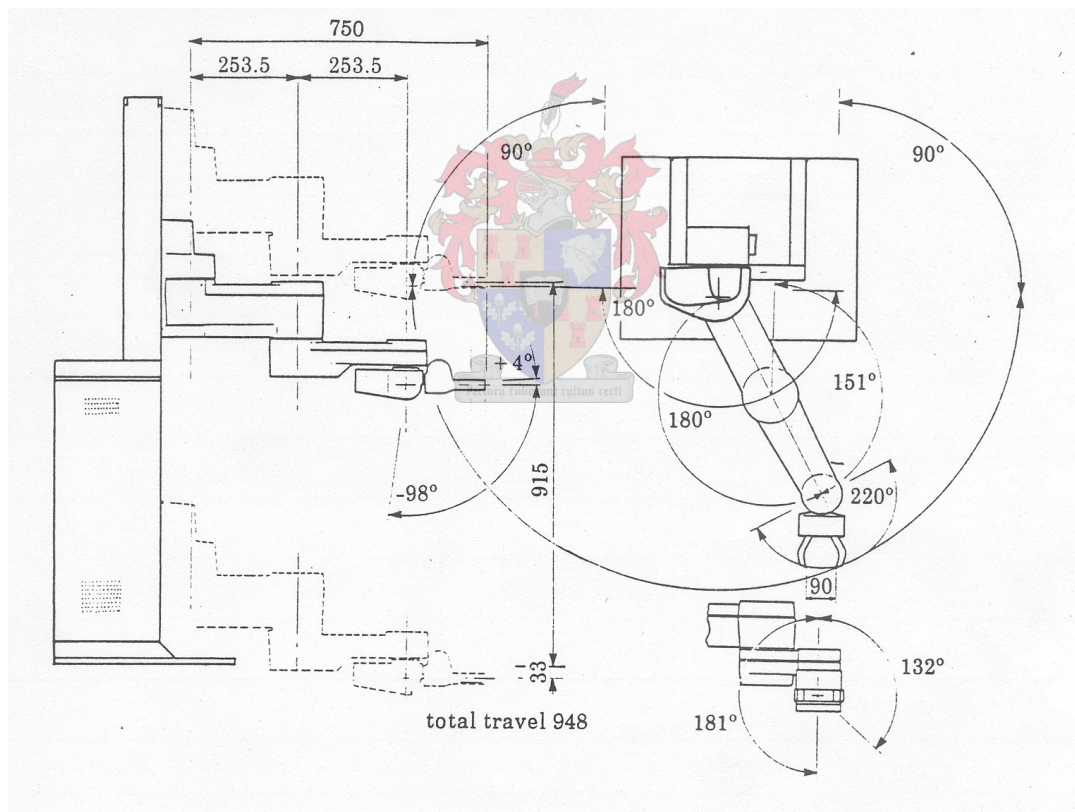


Figure 2.3: Range of movement and dimensions of the RTX robot. (Adapted from [1].)

<i>Axis</i>	<i>Endstop to endstop encoder counts</i>	<i>total range mm or degrees</i>	<i>encoder counts</i>
zed	0 to -3554	948 mm.	3554
zed	underside of wrist to baseboard	881 mm.	3303
zed	gripper pointing downwards to baseboard	738 mm.	2767
shoulder	+2630 to -2630	180 degrees	5260
elbow	+2206 to -2630	331 degrees	4836
yaw	+1071+e/3 to -1071+e/3	220 degrees	2142
pitch	w1 + w2 = 108 to -2642	102 degrees	2750
roll	w1 - w2 = 4882 to -3560	313 degrees	8442
gripper	1200 to -30	90 mm.	1200

Table 2.1: Ranges of travel for the arm axes from the initialised position. (Adapted from [1].)

the last image of figure 2.2 with the arm fully extended and at the top of the linear slideway. Encoders control the individual motors and moving a part of the robot involves changing the encoder count of the corresponding encoder through the IPC protocol. The ranges of travel for the arm axes from the initialised position are shown in table 2.1. In the table w1 is the encoder count for the first wrist motor and w2 for the second wrist motor.

2.2.2 RTX MVAL Programming

The RTX robot was controlled by software called MVAL which is run from a computer connected to the robot by a serial cable. The software and driver controlling the robot was developed as part of a doctoral thesis by H.O.W. von Petersdorff in 1987. Developed in Pascal with MSDOS as operating system, MVAL was used as the controlling interface for the RTX robot until recently. A project in 2000 by André Alberts developed a Windows interface for controlling the robot. Using the interface it is possible to control the robot from a computer next to it, or through the internet. Although Alberts' project was successful MVAL was still being used when this project started. Early control and accuracy experiments were conducted using MVAL before a JAVA interface to the robot was developed by Thomas Wunderlich at the beginning of 2006. MVAL commands can be found in its manual

[8] written by A.R. Greeff. Through using MVAL to control the robot it was determined that the RTX robot is well suited to test the machine vision techniques developed in this thesis.

2.2.3 The Internet Control of a Robotic Work Station

A project by André Alberts [9] at the Industrial Engineering department of the University of Stellenbosch looked into the control of a robotic work station through the internet. The RTX robot was used in this experiment and was successfully controlled through the internet using an interface developed in Delphi. The IPC protocol developed by UMI for the RTX robot was implemented in a Delphi program which controlled the robot. Commands were sent over the internet from a Delphi interface to the controlling Delphi program. The commands were interpreted and sent to the RTX robot using the IPC protocol. The availability of the source code for the interface and control program helped in the understanding of the IPC protocol controlling the RTX robot. The Delphi interface to the RTX robot had limitations and was not used long before MVAL was used as primary control interface again.

2.2.4 RTX Robot Java Interface

To allow the RTX robot to be controlled with new hardware and software, Thomas Wunderlich set at out the end of 2005 to develop a new interface controlling the robot using the IPC protocol. To allow communications with the robot using various operating systems the new interface was written in JAVA. A user-friendly interface, with which the robot can be controlled directly, was also written. As the interface was written in JAVA it is possible to use it from within MATLAB and call specific functions. The JAVA interface was used with MATLAB in this project since it became available and allows the user to control the electric motors of the RTX robot individually. The JAVA interface needed specific motor counts for each of the electric motors to move the robotic arm to a specific position. The motor counts are calculated using the inverse kinematics solution for the RTX robot and converting the calculated joint angles to motor counts. The name and use of the

most important functions of the interface follows. The full function list can be found in the manual [10] written by Wunderlich.

`on()`: This function establishes the connection between the RTX robot and the control PC. It needs to be run after each programme restart.

`cal()`: When this function is called, the RTX robot is calibrated and moved to its calibration position. All positions are calculated from this position.

`stop()`: All robot activities are stopped when this function is called.

`setPos(c,s,e,wh,wv,wt,g)`: This function moves the robot's motors to the positions defined by the integers c , s , wh , wv , wt and g . The value of the column motor count is defined by c , the shoulder motor count by s , elbow by e , wrist yaw by wh , wrist pitch by wv , wrist roll by wt and the gripper motor count by g .

2.2.5 RTX Robot Control and Kinematics

Through research on the internet, two very informative documents were found concerning the control and kinematics of the RTX robot from a modern personal computer. Thomas Wunderlich's JAVA interface established communication with the RTX robot, but the JAVA interface requires motor counts as input, not specific Cartesian positions. To calculate the joint angles and subsequent motor inputs the forward and inverse kinematics solution of the RTX robot was needed. These documents helped in the calculation of these solutions.

The project of A. Fernandez [2] gives information about the history of robotics and basic terms associated with the field. Different kinds of robots are discussed, as well as their geometry and calculation of their kinematics. Detailed information about the control aspects of the RTX robot are discussed, some of which were used in this project. Sam Wane [4] of Staffordshire University's document gives a description of the kinematics associated with the RTX robot and shows how they are calculated. This document was used to calculate and confirm the forward and inverse kinematics solutions of the RTX robot to be used with the JAVA interface.

From [2] the Denavit-Hartenberg formulation was used to find the forward kinematics equation of the RTX robot. An extraction of [2] follows. Fernandez defines the Denavit-Hartenberg equation as a formulation which finds the explicit function linking the joints space with the Cartesian space of positions/orientations. This function takes a vector with as many components as degrees of freedom the robot arm has, and returns a 6-dimensional vector: the three first components are the special position of the terminal point and the last three give the orientation, in terms of an approximation vector or as orientation angles.

$$f : J \rightarrow \mathfrak{R}^6 \quad (2.2.1)$$

The first action in finding the Denavit-Hartenberg formulation is affixing a frame to each link. It can be done by the following steps:

Step 1: Identify the joint axes and draw lines along them.

For step 2 through 5 below consider two of these neighbouring lines:

Step 2: Identify the common perpendicular between them, or point of intersection.

At the point of intersection, or at the point where the common perpendicular meets the i^{th} axis, assign the link frame origin.

Step 3: Assign Z_i pointing along the direction of axis i .

Step 4: Assign X_i pointing along the common perpendicular, or if the axes intersect, assign X_i to be normal to the plane containing the two axes.

Step 5: Assign Y_i to complete a right-hand coordinate system.

Step 6: Assign 0 to match 1 when the joint variable is zero. For N choose an origin location and X_N direction freely, but generally so as to cause as many link parameters as possible to become zero.

From these premises, the Denavit-Hartenberg parameters choice is as follows:

- a_{i-1} = the distance from Z_{i-1} to Z_i measured along X_{i-1}
- α_{i-1} = the angle between Z_{i-1} to Z_i measured about X_{i-1}
- d_i = the distance from X_{i-1} to X_i measured along Z_i

- θ_i = the angle between X_{i-1} to X_i measured about Z_i

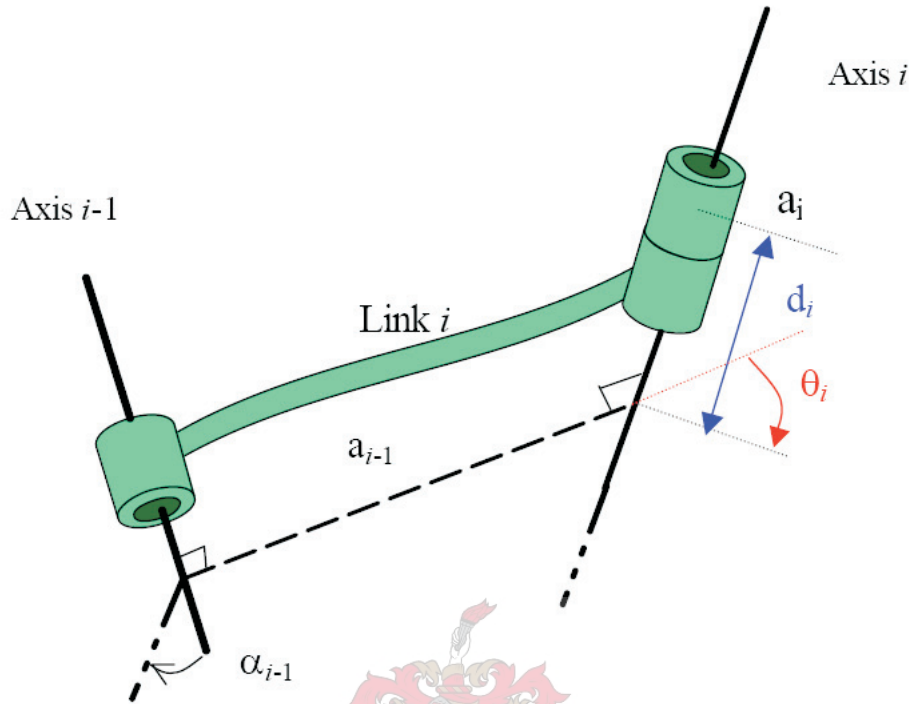


Figure 2.4: Denavit-Hartenberg Parameters for a generic joint. (Adapted from [2].)

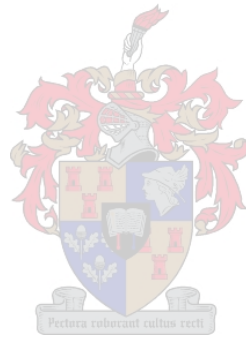
Now, for each link we can divide the transformation in four simple transformations which will drive up to the generic DH matrix for a joint. The resultant matrix, established between the axes $i - 1$ and i , is:

$${}^{i-1}\mathbf{A}_i = \begin{pmatrix} \cos \theta_i & -\sin \theta_i \cdot \cos \alpha_i & \sin \theta_i \cdot \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cdot \cos \alpha_i & -\cos \theta_i \cdot \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

And after that, the global transformation matrix is built by multiplication of successive link matrices from the base to the tool of the robot.

$${}^0A_n = {}^0A_1 \cdot {}^1A_2 \cdot \dots \cdot {}^{n-1}A_n \quad (2.2.2)$$

This homogeneous transformation matrix 0A_n relates a system solitary to the terminal point with the World reference, which has been already defined. Each one of the elements of this matrix T depends on some or every joint's variables and on the geometric constants of the robot arm.



Chapter 3

Object Recognition

3.1 Object Recognition in Flexible Manufacturing

Improving the competitiveness in the modern assembly environment is of the utmost importance for manufacturers. Optimisation of production lines holds the key to improved performance, as manufacturing lines are only cost effective when demand exceeds supply. With the rapidly changing customer environment and the growth of global competition, the demand for a certain product may vary drastically from time to time. To maximise profits production must be controlled according to demand. Pierpaoli and Urbani [11] state that 'Manufacturing companies in the 21st Century will face frequent, unpredictable market changes' and that 'to remain competitive, manufacturing companies must possess a new type of manufacturing system that is highly responsive. They must possess the exact production capacity and functionality when and where it is needed.' Thus a cost effective, responsive manufacturing system must be able to manufacture varying product batch sizes, with the smallest batch consisting of only one product, and be able to produce different products using, where possible, the same production line.

Due to the varying batch sizes and the different products produced, mixed batches will be formed somewhere down the production line. A sorting cell on the production line is needed. Placing the sorting cell near the end of the production line and integrating it with the handling process will improve flexibility and increase the speed of the line. Products will be identified, their position calculated

and handled by manipulators. Depending on the type of product, the manipulator will place the item in the predefined position for the specific product, thus sorting the mixed batches.

Sorting can be done by conventional methods like barcodes or sorting by size, but all of them have their own constraints - there may be not enough space to place a barcode or product dimensions are too similar to sort by size. When these methods cannot be used, a more versatile sorting method is needed - one that can be easily adjusted to sort other groups of products and which does not have the same constraints as current methods. It must be noted that any new method will have constraints of its own. To create a new, more intelligent sorting method, a new kind of recognition sensor is needed. The sensor must be able to differentiate between different objects of varying and similar shapes and sizes, colours and even position in the cell. A suitable sensor can be constructed using cameras equipped with object recognition capabilities. By recognising different objects within the field of view, it will be able to sort objects of the same type without the need for human interaction. Object recognition also makes it possible to identify defects in known products.

Object recognition with the use of cameras involves the translation of the scene into computer-understandable descriptions of objects. Different description techniques are available, of which only one is investigated, namely Fourier Descriptors. The versatility of the Fourier Descriptor technique is the reason behind this decision. In Farooq and Osadebey [12] it is stated that Fourier descriptors are a widely used, all-purpose shape description and recognition technique. It has been used in a variety of fields over the years, including the commerce, medical, biological and technical sectors. In the factory environment most of the work has been done on the identification of defects on manufactured components.

At the 5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering held from 25 to 28 July 2006 in Ischia, Italy, the paper, "The use of Fourier Descriptors for Object Recognition in Robotic Assembly" [13], was presented by the author. It covered most of the aspects of an automated sorting cell using object recognition with Fourier descriptors as presented in this chapter. The

implementation of the recognition process as a whole, as well as the implementation of each of the separate sections, are covered in detail in Chapter 5.

3.2 Algorithm Requirements

Before developing the object recognition algorithm, one needs to take certain factors into consideration. These factors include the environment in which the algorithm is going to be used, the kind of objects that are going to be identified, the placing and orientation of the objects and the available hardware with which images are obtained and calculations are done.

The environment in SENROB where the recognition takes place can be easily modified to suit a specific kind of recognition. Environment factors such as lighting or background can be changed easily; thus environment factors do not contribute to the requirements set for the object recognition algorithm. The same cannot be said about the factors implicated by the objects that need to be recognised. As objects of any shape or colour may be produced on the production line, some with similar features and other with distinct differences, the algorithm needs to be able to handle all of them. Furthermore, objects will not be symmetrical around one or more axis and the orientation in which they are presented to the camera will differ, as well as the distance from the objects to the camera. Limitations imposed by hardware include the resolution of images taken of objects and the computing power at the disposal of the algorithm.

Thus an object recognition algorithm is required that is able to do recognition invariant to the rotation, translation or scale of an object in an image of set resolution within an allowable period of time. Simple colour recognition may also be needed.

3.3 Fourier Descriptors

3.3.1 Mathematical Background

To enable a computer to know what a camera is seeing, the objects in the image need to be described with a systematic computer description. Fourier descriptors describe the shape of an object by considering its boundaries. A short synopsis of the Fourier descriptor mathematics as explained by Gonzalez and Woods [3] follows. It will give the necessary background to understand how the method operates.

Suppose the digital boundary of an object is plotted on the XY plane. Starting at an arbitrary point (x_0, y_0) , coordinate pairs $(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_{K-1}, y_{K-1})$ are encountered in transversing the boundary in a given direction, say clockwise. These coordinates can be expressed in the form $x(k) = x_k$ and $y(k) = y_k$. With this notation the boundary itself can be represented as the sequence of coordinates $s(k) = [x(k), y(k)]$, for $k = 0, 1, 2, \dots, K - 1$. Moreover, each coordinate pair can be treated as a complex number so that

$$s(k) = x(k) + jy(k) \quad (3.3.1)$$

for $k = 0, 1, 2, \dots, K - 1$. The X axis is treated as the real axis and the Y axis as the imaginary axis of a sequence of complex numbers. The advantage of this representation is that it reduces a 2-D problem to a 1-D problem. The discrete Fourier transform (DFT) of $s(k)$ is

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K} \quad (3.3.2)$$

for $u = 0, 1, 2, \dots, K - 1$. The complex coefficients $a(u)$ are called the Fourier Descriptors of the boundary. The Fourier Descriptors are a frequency based description of the boundary of the image. The inverse Fourier transform of the $a(u)$'s restores $s(k)$. That is,

$$s(k) = \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K} \quad (3.3.3)$$

for $k = 0, 1, 2, \dots, K - 1$. The Fourier descriptors uniquely describe the boundary of an object.

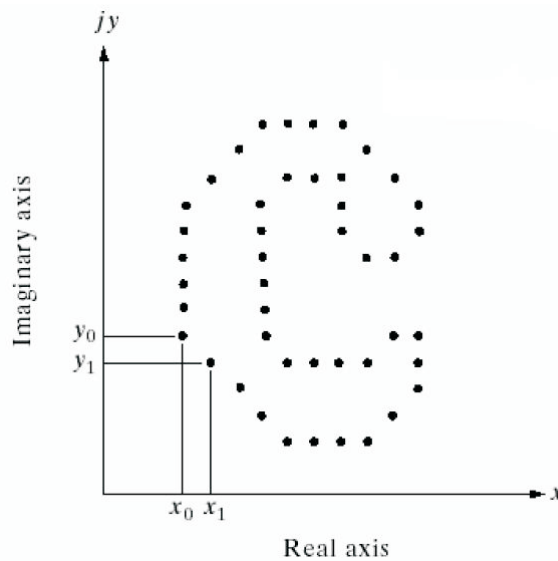


Figure 3.1: A digital boundary and its representation as a complex sequence. The points $(x_0, y_0), (x_1, y_1)$ shown are (arbitrarily) the first two points in the sequence. (Adapted from [3].)

Comparing the descriptors of different objects gives a measurement of their similarity, but do we need all of the descriptors for such a comparison? Using only the most significant descriptors will improve speed and allow for a more intuitive understanding of the information, as negligible data is removed.

Meaningful results can still be obtained by using a subset of the descriptors. For practical images most of the boundary information is located in the low frequency part of the descriptors. In choosing a low frequency subset, only the first P coefficients are used. This is equivalent to setting $a(u) = 0$ for $u > P - 1$ in equation 3.3.3. The result is the following approximation of $s(k)$:

$$\hat{s} = \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K} \quad (3.3.4)$$

for $k = 0, 1, 2, \dots, K - 1$. Although only P terms are used to obtain components of \hat{s} , k still ranges from 0 to $K - 1$. The same number of points still exists in the approximate boundary, but fewer terms are used in the reconstruction of each point. By using fewer terms to approximate the boundary some of the higher frequency components of the boundary are lost, leading to a loss in finer detail in the boundary. The remaining coefficients account for the lower frequency components, which determine the global shape of the boundary. Thus the smaller P becomes, the more detail is lost on the boundary.

It is shown in figure 3.2 that only a few Fourier descriptors can be used effectively to capture the primary shape of a boundary. The square boundary in figure 3.2

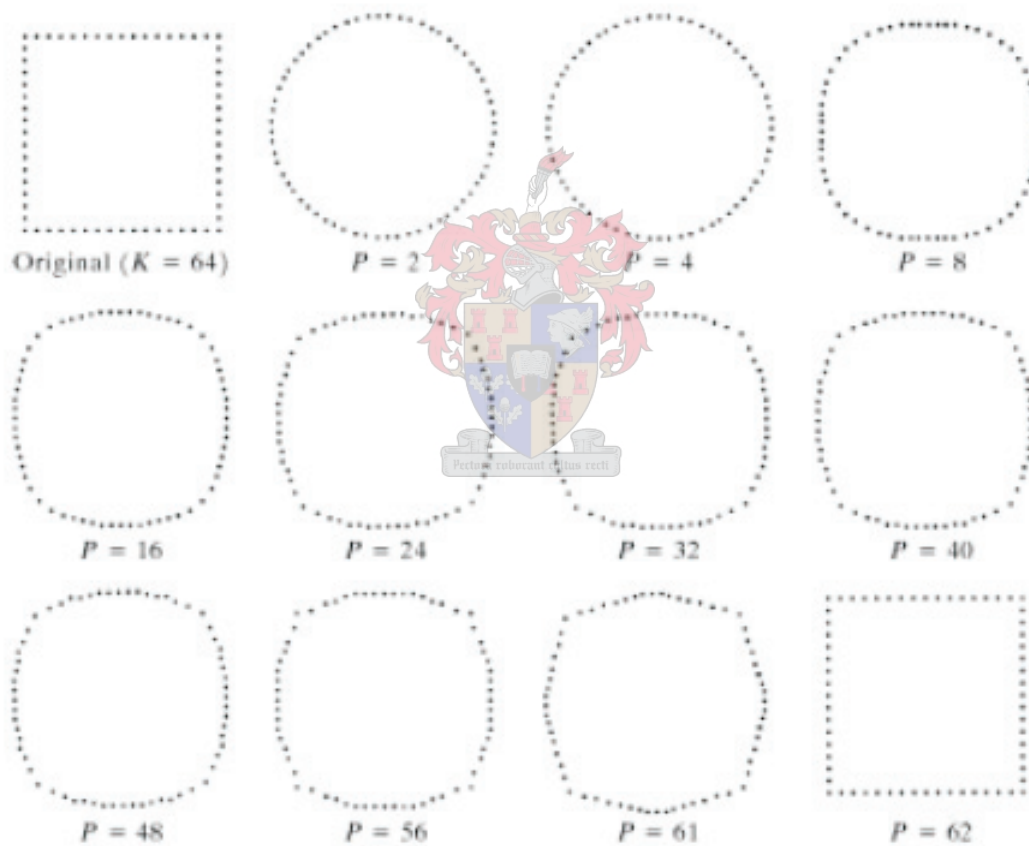


Figure 3.2: Examples of reconstruction from Fourier descriptors. P is the number of Fourier coefficients used in the reconstruction of the boundary. (Adapted for [3].)

consists of $K = 64$ points. We can see that the approximation only becomes more square than circular when P is about 8 and that corner definition only appears until P is about 56 or larger. The lower frequency components form the basis for differentiating between distinct boundary shapes.

<i>Transformation</i>	<i>Boundary</i>	<i>Descriptor</i>
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

Table 3.1: Basic Fourier descriptor properties. (Adapted from [3].)

The Fourier descriptor method for object recognition is invariant to translation, rotation, scale of the boundary, and the order in which points are processed. Although the method is not directly insensitive to these geometrical changes, changes in these parameters can be related to simple transformations on the descriptors. Table 3.1 summarizes the basic properties for the Fourier descriptors $a(u)$ of a boundary sequence $s(k)$ that undergoes rotation, translation, scaling or a change in starting position. Rotation affects all the coefficients equally by a multiplicative constant term $e^{j\theta}$. Translation consists of adding a constant displacement to all coordinates in the boundary, but does not affect any of the descriptors except for $a = 0$, which has the impulse function Δ . A change in scale of the boundary relates to an equal change in scale of the descriptors. Changing the starting point affects all of the descriptors in a different, but known, way. Knowing the effect of each of the geometric changes makes it possible to remove them and achieve invariance to translation, rotation, scale and starting point. Ignoring the first descriptor where $a = 0$, removes the effect of translation, the inverse DFT of only this point is equal to the position of the centre of the boundary or Δ_{xy} in table 3.1. Scaling affects the boundary and the descriptors with the same amount. This constant can easily be calculated and scaling corrections can be made. From table 3.1 we can see that the magnitudes of $a(u)$ are invariant to rotation because:

$$|a_r(u)| = |a(u)e^{j\theta}| = |a(u)| \cdot |e^{j\theta}| = |a(u)| \cdot 1 = |a(u)| \quad (3.3.5)$$

The magnitude of $a(u)$ is invariant to the starting point in the coordinate sequence for the same reason. Having removed all variances, the complex Fourier descriptor sequences can now be compared. Boundaries of similar objects will have a small difference. If the difference between an unknown and known object boundary's descriptors is small, then they must be of the same kind. The comparison and usage of Fourier descriptors are covered in detail in chapter 5.

Removing all of the invariances due to the properties in table 3.1 from the descriptors allows for easy comparison between two different sets of descriptors. Although never implemented in this project, the properties can be used to calculate the rotation of an object or even its position by looking at the change to the descriptors due to translation and scale.

This paragraph explains the significance of each of the descriptors of a boundary. To start, the Fourier descriptors of the boundary in figure 3.3(a) is calculated by taking the discrete Fourier transform of the boundary. Figure 3.3(b) shows the magnitude for the 21 lowest frequency descriptors. As it is the result of a discrete Fourier transform the highest frequency components are found in the middle of the plot and the lowest at the beginning and end. The first Fourier descriptor, $a(0)$, in 3.3(b) represent the centre of the boundary. The inverse Fourier transform of it is the coordinate of the crosshair in 3.3(a), the centre of the boundary. Each

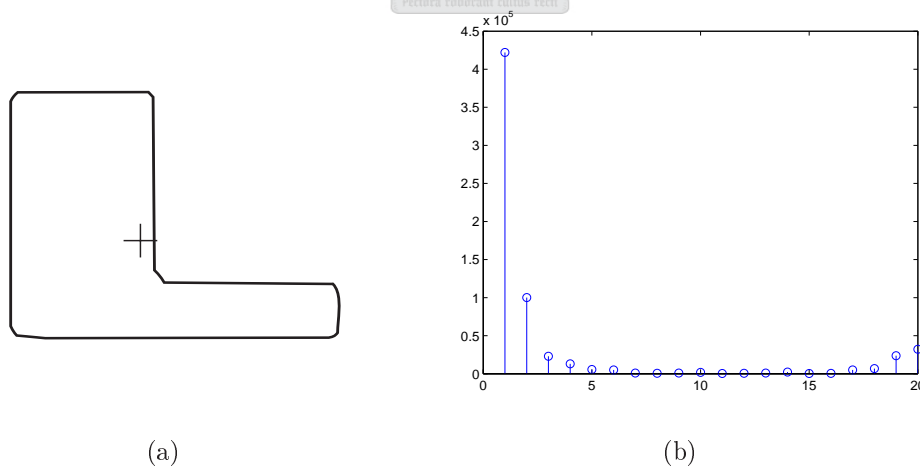


Figure 3.3: Boundary, (a), and its Fourier descriptors, (b).

of the following descriptors represents a vector rotating around the origin with a constant magnitude, forming a circular footprint. The magnitude of the vector is dependant on the magnitude of its descriptor. The vectors represented by the second and the last descriptor form a set with a frequency of one complete rotation when moving around the boundary. The second descriptor rotates anti-clockwise and the last clockwise. The third and second to last descriptor's vectors complete two full rotations when transversing the boundary and the next set will complete three full rotations and so forth. Adding all of the vectors together at a certain point in transversing the boundary will give an approximation of a point on the boundary. Repeating this process for every point an approximation of the complete boundary can be found. Figure 3.4 shows the vectors related to the first and the last descriptors being added together. The first vector, rotating anti-clockwise, relates to the second descriptor and is added to the centre of the boundary as calculated from the very first descriptor in the sequence. The second vector related to the last descriptor, rotating clockwise, is added to the centre and first vector resulting in an approximation of a point on the boundary. As the vectors rotate around their origins while we transverse the full 360 degrees around the origin the complete

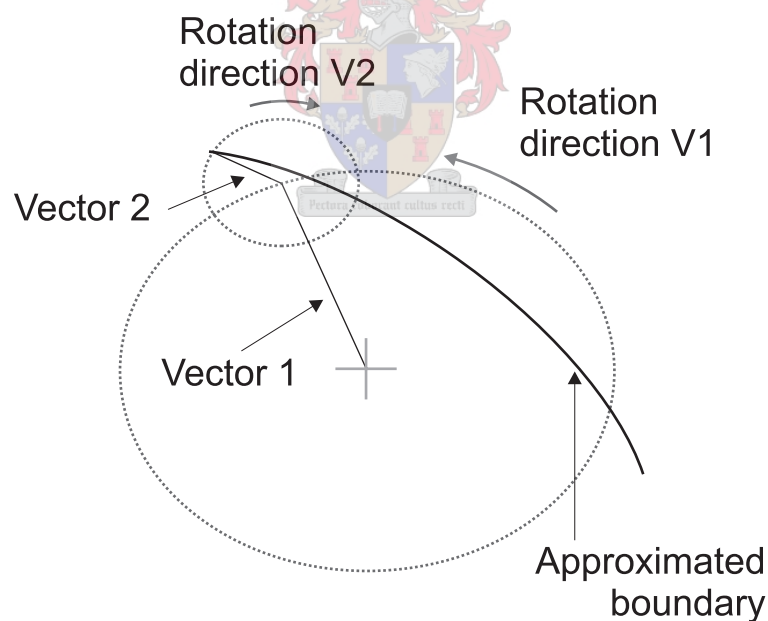


Figure 3.4: The approximation process featuring vectors related to the two lowest frequency components.

approximation of the boundary is calculated. An increase in descriptors used will improve the accuracy of the approximation.

3.3.2 Relationship with the Fourier Series

Similar to the use of a Fourier series to construct a specific time signal using sine waves of different amplitude and frequency, the Fourier descriptor method uses a series of circles with different sizes and frequencies to build up a two dimensional plot of a boundary. Each descriptor coefficient is the frequency representation of a circle in the two dimensional complex plane, XY .

The Fourier series is a mathematical tool which allows us to analyse arbitrary periodic signals by decomposing them to a weighted sum of simpler sinusoidal

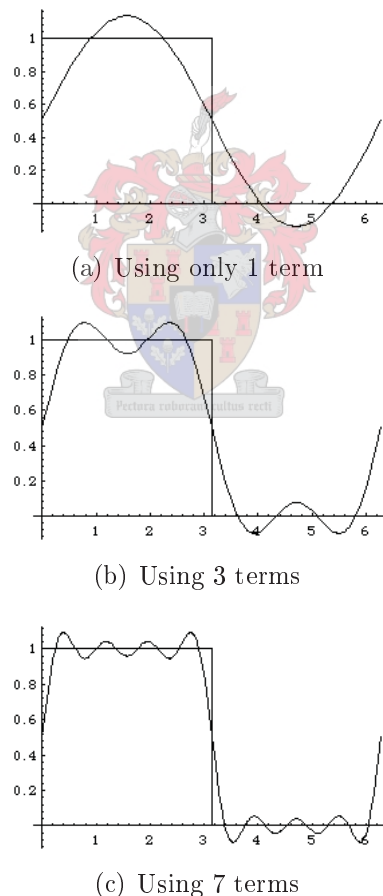
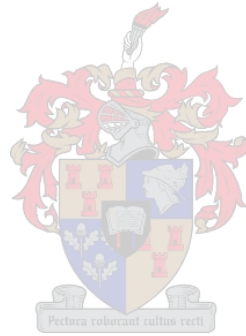


Figure 3.5: Approximating a square wave with different amount of sine waves.

components or terms. The coefficients, or terms, are a one-to-one mapping of the original function. Any periodic signal can be approximated by using only a few of these terms. In figure 3.5 we can see how the accuracy of the approximation increases as the number of terms used in the reconstruction of a square wave increases. With only one term the approximation is, as expected, a single sine wave. At a three term approximation the general shape of the signal becomes apparent and at seven terms the shape is even more visible. With more terms higher frequency components are added which account for the finer details in the signal. A boundary plotted on in the complex XY plane can easily be interpreted as a periodic signal. In transversing the boundary more than once a complex periodic signal can be constructed which can be decomposed to a weighted sum of simpler circles in the complex plane. The original boundary can be approximated in the same way as the periodic signal by using only a few terms. This has already been shown in figure 3.2 where a square boundary is approximated with different amount of terms or sum of circles in the complex plane.



Chapter 4

Robotics

4.1 RTX Geometry

In order to calculate the kinematics of a robot the geometry of the specific robot is needed. The RTX robot from UMI is a SCARA type robot, providing a blend of features of cylindrical and articulated robots. With the help of figure 4.1 the movements of the RTX robot can be explained. The whole robot arm moves up and down along the tower of the robot, while the next three joints are rotary. These joints are the shoulder, elbow and wrist yaw. All three joints rotate around axes parallel to the tower, a familiar feature of SCARA robots, so that movement of the previous joint affects the position of all of the following joints. The wrist of the RTX robot is able to perform roll and pitch movements.

Knowing the geometry of the robot a Cartesian coordinate system is chosen, as in figure 4.1, which will be the reference to express the position of the robot arm and orientation of the gripper. The shoulder rotates around the Z-axis in the XY-plane. The rotation of the elbow is around the Z-axis in the XY-plane if a new coordinate system, with the system centred at the elbow point, is used. With a new coordinate system centred at the wrist point yaw consists of a rotation around the Z-axis, pitch a rotation around the X-axis and roll a rotation around the Y-axis.

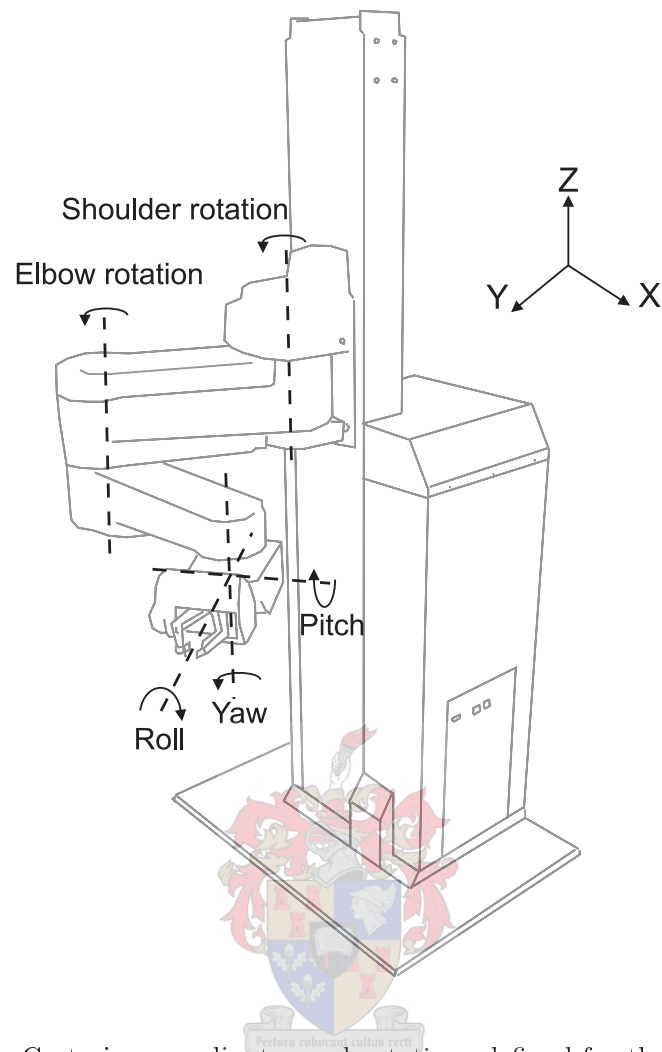


Figure 4.1: Cartesian coordinates and rotations defined for the RTX robot.

4.2 Robot Kinematics

4.2.1 Forward Kinematics of the RTX Robot

The forward kinematics equation is used to calculate the position of the end point of the robotic arm from the angles for the individual joints. An adequate kinematics method is an important tool when working with robotics, as it simulates the movement of the manipulator. The Denavit-Hartenberg method establishes a series of steps leading to a homogenous matrix which relates the angles of the joints to a Cartesian position for the end of the robotic arm. Using the Cartesian coordinate system defined in the previous section and illustrated in figures 4.1 and 4.4 and

the following Denavit-Hartenberg matrices, the forward kinematics equation can be found. The Denavit-Hartenberg matrix for rotation about the Z-axis is:

$$\theta_i : \begin{pmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.1)$$

Around the X-axis the Denavit-Hartenberg matrix is

$$\theta_i : \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_i & -\sin \theta_i & 0 \\ 0 & \sin \theta_i & \cos \theta_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.2)$$

and for a rotation about the Y-axis it is

$$\theta_i : \begin{pmatrix} \cos \theta_i & 0 & \sin \theta_i & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta_i & 0 & \cos \theta_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.3)$$

with a translation along X, Y or Z represented with the matrix

$$xyz : \begin{pmatrix} 0 & 0 & 0 & x_t \\ 0 & 0 & 0 & y_t \\ 0 & 0 & 0 & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.4)$$

where x_t representing the translation along the X-axis, y_t the translation along the Y-axis and z_t translation along the Z-axis.

Using equations 4.2.1, 4.2.2, 4.2.3 and 4.2.4 the forward kinematics equation for the RTX robot is found. The column up and down movement of the arm is a simple translation along the Z-axis, z_c , and is represented in the first matrix in equation 4.2.5. The physical length of the shoulder results in a translation along the Y-axis,

y_s , while movement of the shoulder is a rotation about the Z-axis, θ_s . This, and the shoulder translation is also represented by the first matrix in 4.2.5. The movement of the elbow causes a translation, y_e , due to the length of it and a rotation around the Z-axis, θ_e , which the second matrix in 4.2.5 represents. 0A_3 forms the partial matrix for the movement of the robotic arm from the base to the wrist. Multiplying the matrices in equation 4.2.5 leaves equation 4.2.6, the Denavit-Hartenberg matrix for movement from the base of the robot to the base of the wrist.

$${}^0A_3 = \begin{pmatrix} \cos \theta_s & -\sin \theta_s & 0 & 0 \\ \sin \theta_s & \cos \theta_s & 0 & y_s \\ 0 & 0 & 1 & z_e \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_e & -\sin \theta_e & 0 & 0 \\ \sin \theta_e & \cos \theta_e & 0 & y_e \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.5)$$

$${}^0A_3 = \begin{pmatrix} \cos(\theta_s + \theta_e) & -\sin(\theta_s + \theta_e) & 0 & -y_s \sin \theta_s - y_e \sin(\theta_s + \theta_e) \\ \sin(\theta_s + \theta_e) & \cos(\theta_s + \theta_e) & 0 & y_s \cos \theta_s + y_e \cos(\theta_s + \theta_e) \\ 0 & 0 & 1 & z_c \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.6)$$

From the wrist to the gripper the first rotation encountered, θ_w , is that of the wrist yaw around the Z-axis, represented by the first matrix in equation 4.2.7. The second matrix in 4.2.7 represents the pitch rotation, θ_p of the wrist around the X-axis. The last matrix represents the roll rotation, θ_r , around the Y-axis and the gripper length translation, y_w , along the Y-axis. Equation 4.2.8 represents the Denavit-Hartenberg matrix for the movement from the wrist to the gripper of the robotic arm and is derived from equation 4.2.7. In 4.2.7 and 4.2.8 the operator \cos is shortened to c and \sin shortened to s .

$${}^3A_6 = \begin{pmatrix} c\theta_w & -s\theta_w & 0 & 0 \\ s\theta_w & c\theta_w & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta_p & -s\theta_p & 0 \\ 0 & s\theta_p & c\theta_p & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\theta_r & 0 & s\theta_r & 0 \\ 0 & 1 & 0 & y_w \\ s\theta_r & 0 & c\theta_r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.7)$$

$${}^3A_6 = \begin{pmatrix} c\theta_w \cdot c\theta_r - s\theta_w \cdot s\theta_p \cdot s\theta_r & -c\theta_p \cdot s\theta_w & & & & \\ c\theta_r \cdot s\theta_w + c\theta_w \cdot s\theta_p \cdot s\theta_r & c\theta_w \cdot s\theta_p & & & & \\ & -c\theta_p \cdot s\theta_r & s\theta_p & & & \\ & 0 & 0 & & & \\ & & & c\theta_w \cdot c\theta_r - s\theta_w \cdot s\theta_p \cdot s\theta_r & -c\theta_p \cdot s\theta_w & \\ & & & c\theta_r \cdot s\theta_w + c\theta_w \cdot s\theta_p \cdot s\theta_r & c\theta_w \cdot s\theta_p & \\ \dots & & & -c\theta_p \cdot s\theta_r & s\theta_p & \\ & & & 0 & 0 & \end{pmatrix} \quad (4.2.8)$$

The full forward kinematics can be found by multiplying the two partial matrices 0A_3 and 3A_6 . The resulting equation (4.2.9) is the global kinematics matrix for the RTX robot.

$${}^0A_6 = {}^0A_3 \cdot {}^3A_6 \quad (4.2.9)$$

Substituting the values for the angles and translations into the matrix 0A_6 will yield the x,y and z position of the tip of the robotic arm in the last column as illustrated in equation 4.2.10.

$$\begin{pmatrix} * & * & * & x_{position} \\ * & * & * & y_{position} \\ * & * & * & z_{position} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.2.10)$$

4.2.2 Positioning of the Gripper

The positioning of the gripper at the end of the robotic arm is a more complicated problem. The process of finding solutions for the joint angles resulting in a specific Cartesian position for the end of the arm is sometimes called finding the robot's inverse kinematics solution. Whereas the forward kinematics solution is always defined, the inverse kinematics solution is not always solvable. This means that it is not always possible to move a manipulator to a specific position or orientation. Fernandez [2] states that all practical six-axis robots, such as the RTX, usually have a solution for a position within the range of movement of the specific robot, but positions do exist for which an inverse kinematics solution can not be solved, called singularities. The range of movement of the RTX robot is shown in figure 4.2. Movement of the robot arm is covered in chapter 2, while detailed information

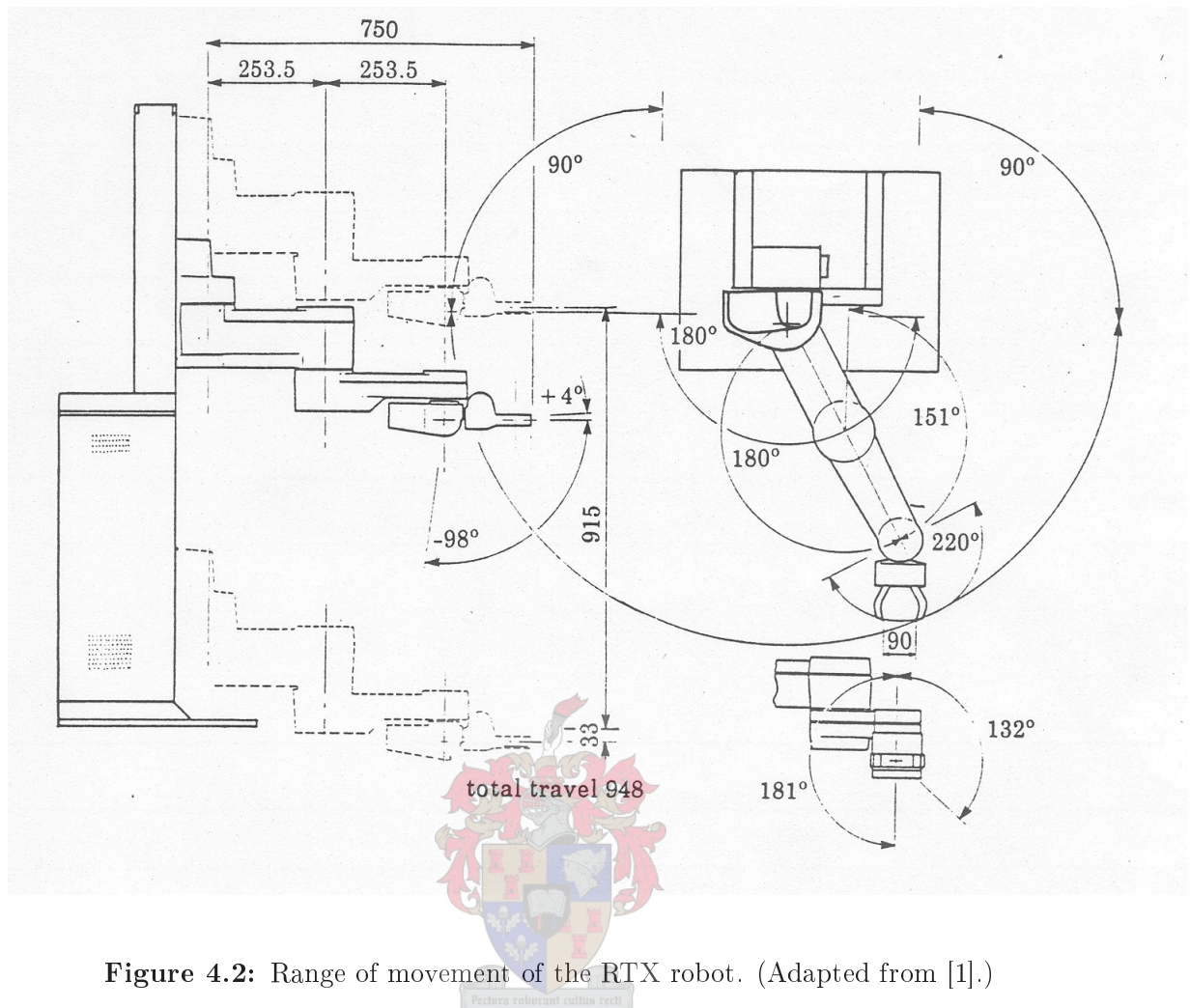


Figure 4.2: Range of movement of the RTX robot. (Adapted from [1].)

can be found in its manuals [1].

Algorithms for solving the joint angles to position simple mechanisms are easily calculated, but for more complicated, multiple axes manipulators it is much more difficult to solve the inverse kinematics equation using only geometry. Reasons include the necessity of relating the orientation of the tool to the corresponding axial position, the existence of more than one solution and the solving of nonlinear equations. Robots with more than six axes can have an infinite number of solutions, but for the RTX and other with six axes the number is finite.

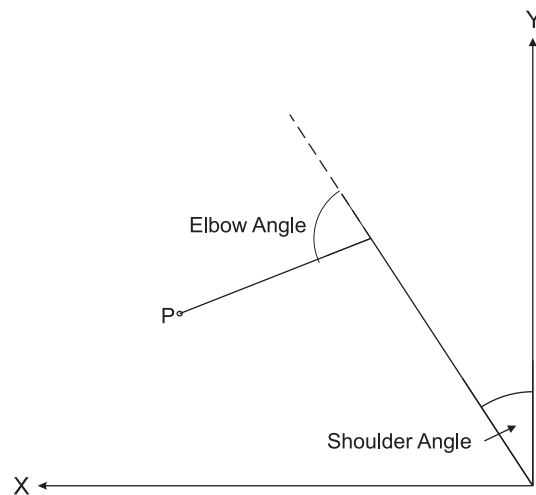
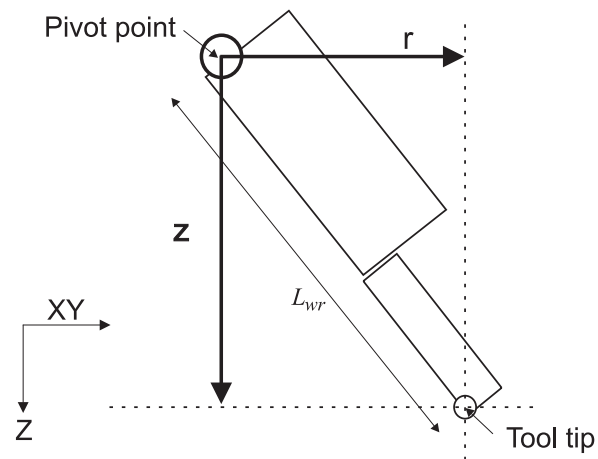


Figure 4.3: Top view of RTX robot with shoulder and elbow.

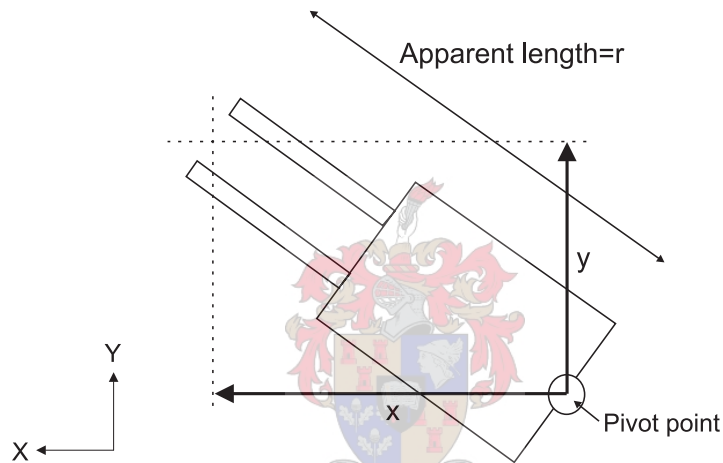
No standard method exists to construct the inverse kinematics of a robot. Different methods have been developed for different robots depending on their particular configuration and characteristics, but even with these methods the problem is complex if all of the axes are taken in consideration. Knowing or choosing the wrist yaw, pitch and roll angles greatly reduces the complexity of the problem and allows for a simpler solution.

When using the RTX robot to handle recognised objects, we have control over where the objects are picked up and put down for sorting. Choosing these positions not to be singular for chosen wrist yaw, pitch and roll orientations, geometry can be used to find solutions for the remaining joints, the shoulder and elbow, of the robot. Using a method from [4] the robot is viewed from above as a two link manipulator from where the law of cosines is used to calculate the shoulder and elbow angles. However, before this can be done the position at the end of the elbow, P in figure 4.3, has to be calculated. This is done using similar methods as those used by Wane in [4]. Images explaining the methods were also adapted from [4].

Calculating the position of point P in figure 4.3 is done by removing the contribution of the wrist due to its pitch and yaw from the required position for the end of the robotic arm. The contribution of the pitch is removed first. Looking at fig-



(a) Side view of wrist showing the contribution of the pitch to x and z . (Adapted from [4].)



(b) Top view of wrist showing the contribution of the yaw to x and y . (Adapted from [4].)

Figure 4.4: Contributions of the wrist to the x , y and z position to the end position of the robot arm.

Figure 4.4(a) the length of the wrist in the XY plane and the z -direction is calculated using:

$$r = L_{wr} \cdot \cos \theta_p \quad (4.2.11)$$

$$z = L_{wr} \cdot \sin \theta_p \quad (4.2.12)$$

The variable L_{wr} resembles the length of the tool, while θ_p is the wrist pitch angle. Z is the contribution of the wrist to the z -position of the arm and r the vector length of the wrist in the XY plane. This vector length is dissected into individual

contributions of the wrist to the x-and y-positions of the end of the robot due to the wrist yaw. This contribution of r in the X and Y direction is calculated next. Looking at figure 4.4(b) and using geometry to find the following equations where θ_y is the wrist yaw angle, these contribution are found.

$$y = r \cdot \cos \theta_y \quad (4.2.13)$$

$$x = r \cdot \sin \theta_y \quad (4.2.14)$$

The vector length of the wrist in the XY plane is r , y is the contribution of r in the y-direction and x the contribution in the x-direction. Roll does not affect the position of the gripper, only its orientation. Knowing the wrist's contributions to the robotic arm's final position and removing them the point P in 4.3 is found. Using the law of cosines the shoulder and elbow angles that result in this position can be calculated.

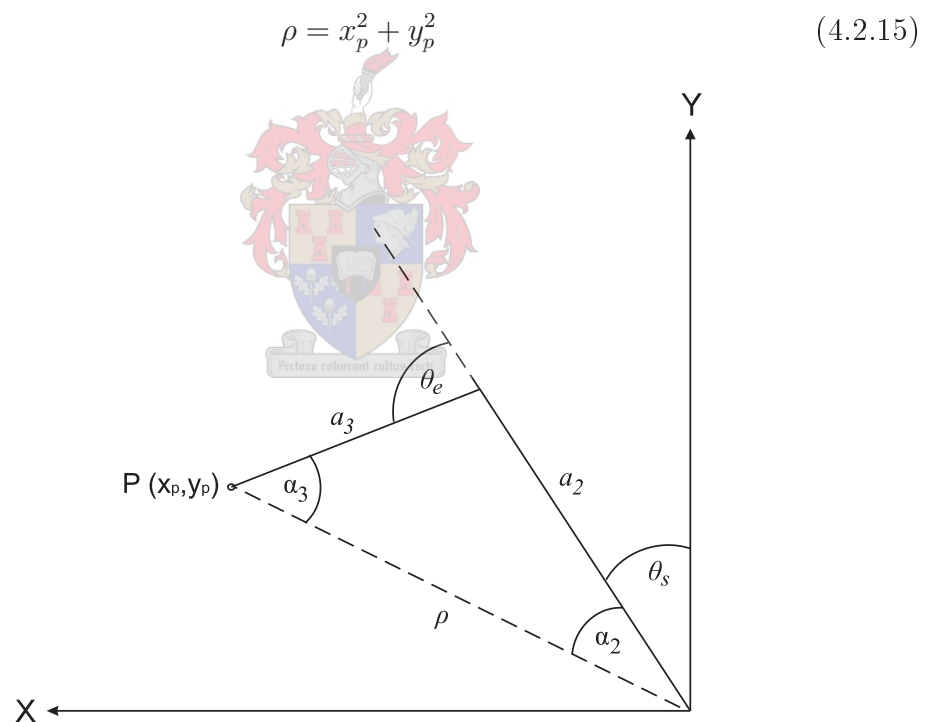


Figure 4.5: Using the law of cosines to find the shoulder angle, θ_s , and elbow angle, θ_e .

Knowing the distance from the origin to the point P to be ρ we can find a solution for the lengths a_2 and a_3 in figure 4.5 using the law of cosines.

$$a_3^2 = a_2^2 + \rho^2 - 2a_2\rho \cos \alpha_2 \quad (4.2.16)$$

$$a_2^2 = a_3^2 + \rho^2 - 2a_3\rho \cos \alpha_3 \quad (4.2.17)$$

Making α_2 and α_3 the subjects of equation 4.2.16 and 4.2.17 we find:

$$\alpha_2 = \cos^{-1} \left(\frac{\rho^2 + a_2^2 - a_3^2}{2a_2\rho} \right) \quad (4.2.18)$$

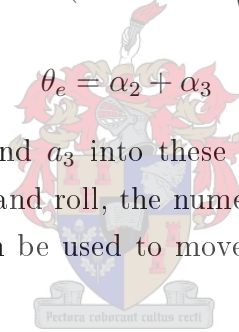
$$\alpha_3 = \cos^{-1} \left(\frac{\rho^2 + a_3^2 - a_2^2}{2a_3\rho} \right) \quad (4.2.19)$$

Using α_2 and α_3 the angles for the shoulder, θ_s , and elbow, θ_e are:

$$\theta_s = 90 - \left(\alpha_2 + \tan^{-1} \left(\frac{y_p}{x_p} \right) \right) \quad (4.2.20)$$

$$\theta_e = \alpha_2 + \alpha_3 \quad (4.2.21)$$

Substituting values for ρ , a_2 and a_3 into these equations and choosing suitable angles for the wrist yaw, pitch and roll, the numerical values for the shoulder and elbow angles are found and can be used to move the tip of the robotic arm to a specific position.



Chapter 5

System Integration

5.1 Hardware and Function Allocation

The integrated system consists of the hardware and the associated functions performed inside each hardware block shown in figure 5.1.

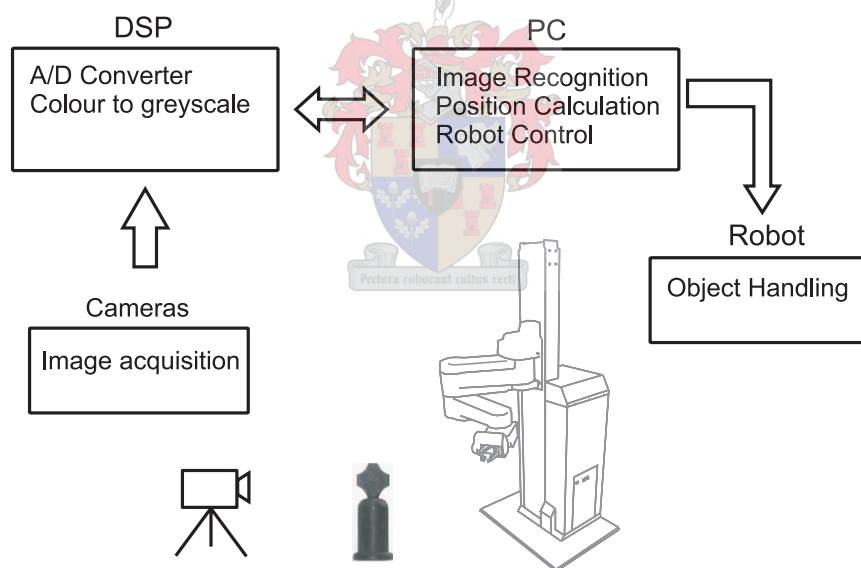


Figure 5.1: Hardware and Function Allocation.

5.2 The Automated Sorting Cell

subsectionEnvironment and System Setup Knowing the sorting environment and

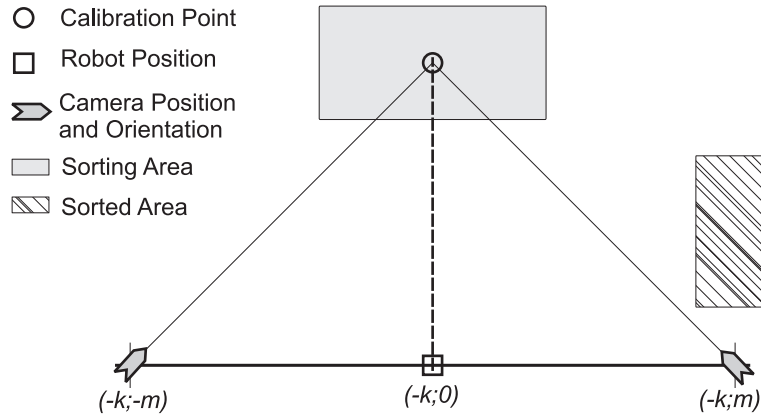


Figure 5.2: Setup of the Sorting Cell Environment.

choosing a suitable setup for the hardware is important to ensure maximum performance. The sorting environment in SENROB is on top of a large working bench with a flat wooden surface. Lighting in SENROB is provided by overhead fluorescent tubes, which provide sufficient lighting for the recognition algorithm. Additional lighting placed above each of the cameras used in the sorting cell, angled at the calibration point, improves the recognition algorithm performance slightly, as the boundaries of objects are more clearly visible. Looking at figure 5.2 the calibration point forms the middle of the setup and all hardware is placed relative to this point. The robot's base is placed at $(-k, 0)$ with $k = 535$ mm. with the robotic arm fully extended directly above the calibration point in its calibration position. With the centre of the coordinate system at the calibration point, objects to be recognised are placed in the rectangular area with corners at (y, x) positions of $(-55, -75)$, $(55, -75)$, $(55, 75)$ and $(-55, 75)$ and taken to the sorted area with (y, x) corners at $(-195, -400)$, $(-195, -440)$, $(-385, -400)$ and $(-385, -440)$ when handled. Each type of object has a specific place in the sorted area where it is placed by the manipulator. The cameras are placed out of the robotic arm's way at $(-320, -535)$ and $(320, -535)$ respectively, resulting in a 60 degree view angle for each of the cameras.

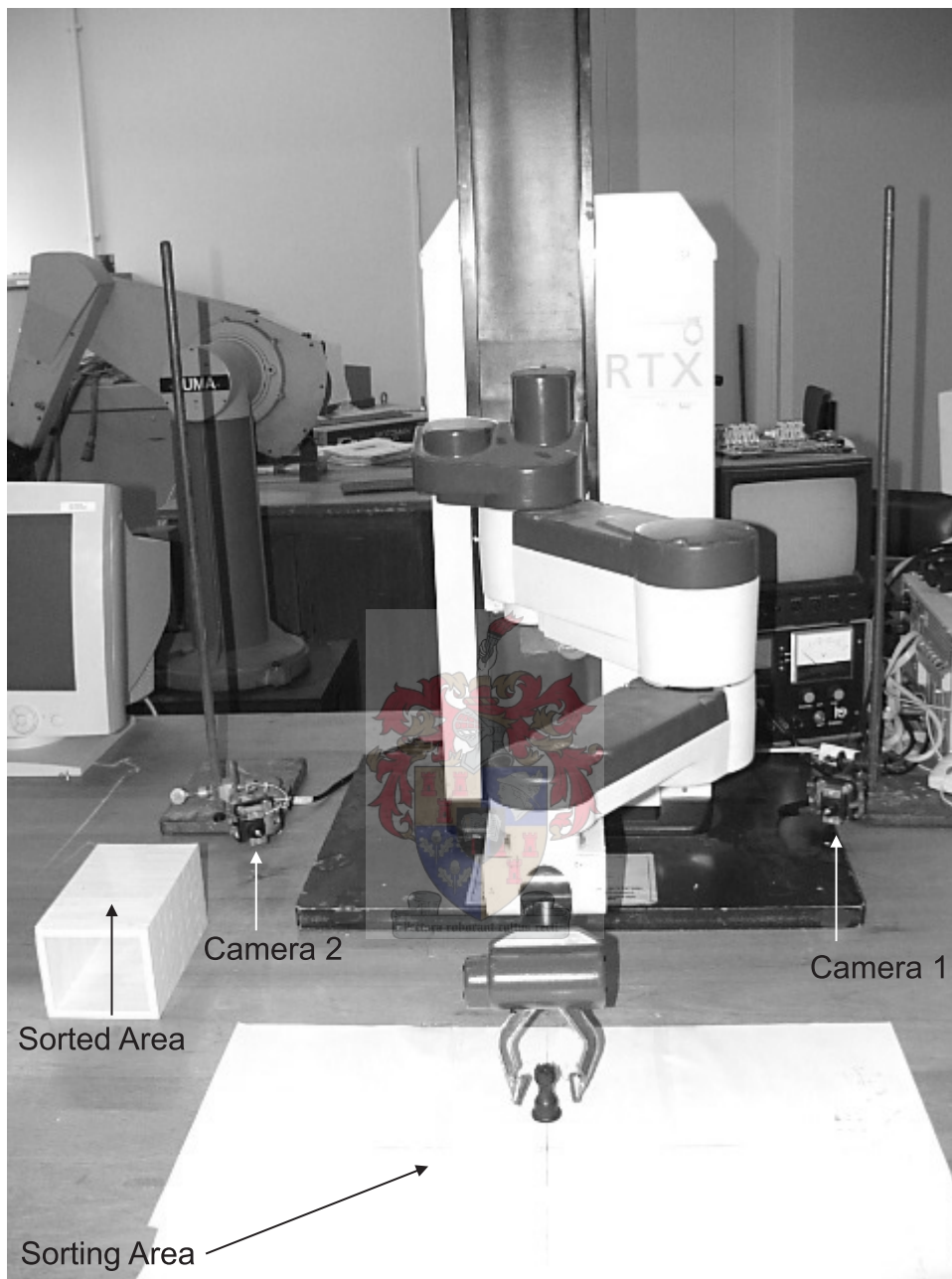


Figure 5.3: Photo of the sorting cell environment.

5.2.1 Sorting Process

The sorting process starts when a new batch of unknown objects arrive at the sorting station. The station is notified that a new batch has arrived and a picture of the batch is taken with each of the cameras. The pictures are sent to the recognition algorithm which recognises the objects in the batch, identify their colour and calculate their positions. The recognition process is discussed in detail in the next section. After the objects have been recognised, they are handled by the robotic manipulator according to strategies described later in this chapter. The sorting process ends with the removal of the tray in which the objects arrived. Any unknown or defective objects will be removed with the tray. The arrival, notification and removal of the tray is done manually until full integration with the rest of the simulated factory environment in SENROB is possible. Full integration was not possible as other projects related to the simulated factory environment in SENROB were not yet completed.

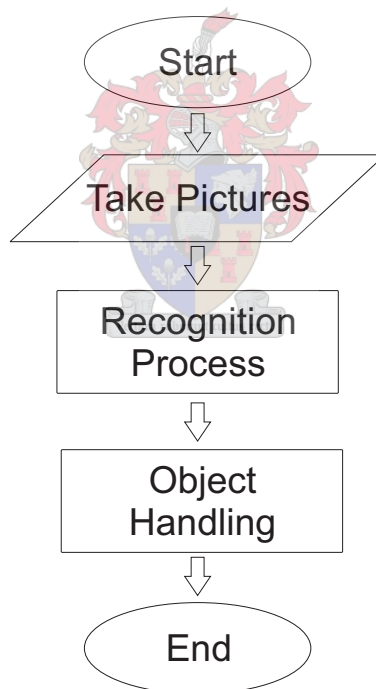


Figure 5.4: Flow diagram of the Sorting Process.

5.3 Hardware

Hardware used in the development of the automated sorting cell includes the RTX robot, two CCD cameras, a Blackfin DSP evaluation kit from Analog Devices and a personal computer. The two analog cameras are connected to the Blackfin evaluation kit which was programmed to act as an analog to digital converter for the camera images. When new images of the sorting environment are required, the personal computer sends a command via a serial interface to the Blackfin. It takes a picture from each of the cameras and send the digital images back to the computer. The computer does the necessary calculations and sends appropriate commands to the robot. The robot and computer are connected by a serial cable. Figure 5.5 illustrates how the hardware is connected.

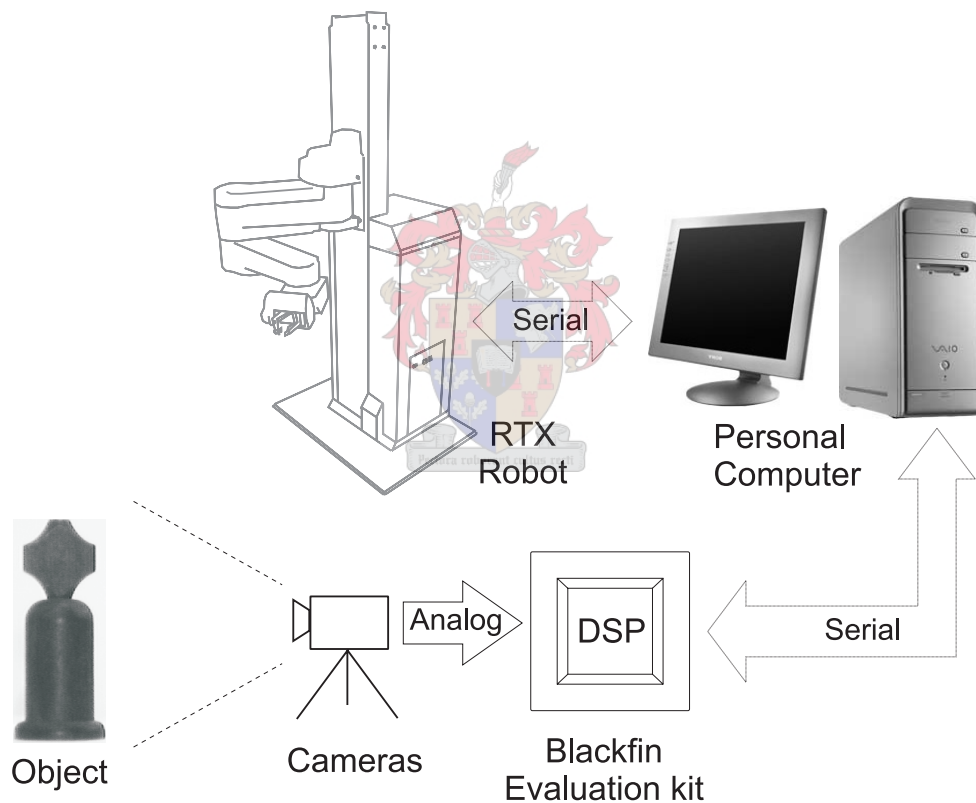


Figure 5.5: Hardware connection for the automated sorting cell.

5.3.1 RTX Robot

The RTX robot was manufactured by Universal Machine Intelligence and is controlled by the IPC protocol developed by the company. The IPC protocol allows status requests and control of the joints at different speeds. Using the JAVA interface written by Wunderlich [10] to interpret commands, the robot was controlled. The interface allows each of the joints to be controlled separately, but the interface is unable to change the speed of movement. Moving the manipulator to a new position consists of converting the joint angles to motor encoder counts and using the interface to send the counts to the robot. Motor encoder counts are calculated using table 5.1 and send via the JAVA interface to the robot with the following command where z , s , e , y , p , r and g represent the encoder counts for the zed, shoulder, elbow, yaw, pitch, roll and gripper motors respectively:

```
InterRob.pushpos(z,s,e,y,p,r,g)
```

<i>Motor</i>	<i>Movement</i>	<i>Encoder Counts</i>
z	1 mm.	3.74953
shoulder	1 degree	29.2227
elbow	1 degree	14.6113
yaw	1 degree	9.73994
pitch	1 degree	13.4862
roll	1 degree	13.4862
gripper	k mm.	$\frac{-0.0585 + \sqrt{0.0585^2 + 4 \cdot 10.7 \cdot 10^{-6} k}}{2 \cdot 10.7 \cdot 10^{-6}}$

Table 5.1: Constants for converting distances and angles to motor encoder counts.

When the RTX is programmed to move to a new position, the RTX's onboard processors calculate a velocity profile for each of the moving motors. The velocity profile of the motors are calculated to move the manipulator to a new position as quickly as possible, moving all of the motors at once, each at its maximum speed. The maximum speed using the JAVA interface is constant.

Aspects such as the physical size, range of movement, design, control and mechanical properties are briefly discussed in chapter 2 and detailed information can

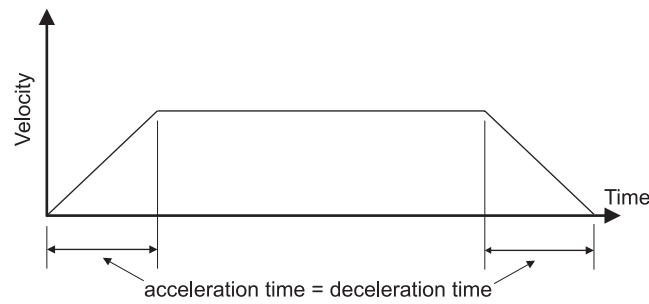


Figure 5.6: Velocity profile for each of the motors. Adapted from [1].

be found in the RTX manuals [1]. The gripper's size is not included in the documentation. The flat contact area on both sides of the gripper is 2 cm. by 2 cm..

5.3.2 Camera hardware

Two AVC526LN mini colour CCD cameras are used to capture images of the sorting area. As they are analog, an analog to digital converter is implemented using a Blackfin DSP evaluation kit. The colour images are also reduced to 8-bit greyscale images with the Blackfin DSP, as they convey enough information and reduce the complexity of the images. Digital images are sent to a personal computer which uses them in the recognition process. The AVC526LN camera can accommodate a variety of lenses with different focal lengths. For this project, the cameras are



Figure 5.7: AVC526LN Mini CCD Camera.

needed to focus at the calibration point in figure 5.2 which is $\sqrt{535^2 + 320^2} = 623$ mm. away from each of the cameras. This is the required working distance, WD. A field of view (FOV) of about 190 mm. is required at this distance. Using equation 5.3.1 with the CCD width for the AVC526LN equal to 3.2 mm., it is found that a 10.5 mm. focal length (FL) lens is needed. The closest available lens, a 12 mm., was acquired for both of the cameras.

$$FL = CCD \cdot \frac{WD}{FOV} \quad (5.3.1)$$

5.3.3 Blackfin Evaluation Kit

In order to use the available analog CCD cameras for the project, an analog to digital converter was needed. An available Blackfin 533 evaluation kit was programmed in C to interface with both the cameras through the AV ports and the personal computer using a serial link. The Blackfin 533 evaluation kit includes the ADV7171 analog to digital converter, which was programmed to accept signals from the cameras and store the digital image in the kit's memory when the evaluation kit receives the appropriate command from the control computer. The DSP then converts the digital colour images to 8-bit greyscale images and send the image data over the serial link to the personal computer which uses it to recognise objects, identify their colours and calculate their positions.



5.4 Recognition

5.4.1 The Process

The recognition process is the most important step in the sorting process. Unknown objects must be recognised accurately, their colours identified and their positions calculated. The process starts when pictures of the sorting area are received. The digital boundary of the objects in the images are found and interpolated to a set number of points. From here the boundaries are send to a guard algorithm that decides if a specific boundary belongs to the object set based on the size, orientation and the size of the object's circumference. This process is less computational intensive than the recognition process and avoids running the recognition algorithm

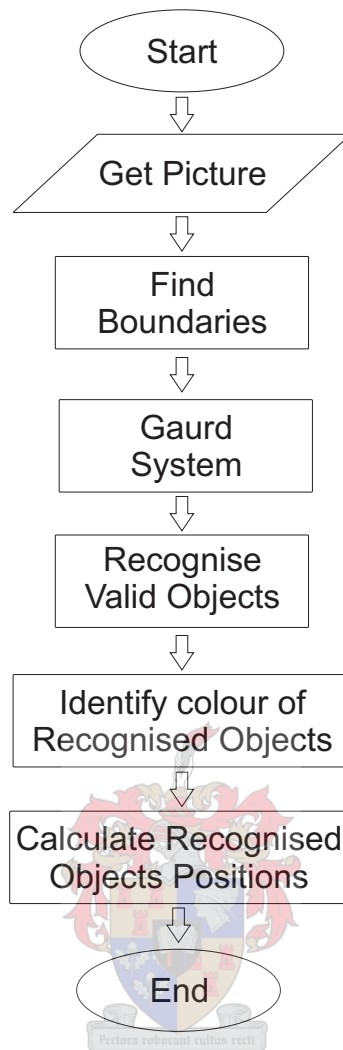


Figure 5.8: Flow diagram of Recognition Process.

on boundaries that are definitely not known. Valid objects' boundaries are sent to the recognition algorithm which decides on the boundaries type and assigns the appropriate label to the boundary. The positions of the recognised objects are then calculated and stored after which the information about the recognised objects are available for use in the handling algorithm.

5.4.2 Objects to be recognised

To verify the performance of the automated sorting cell and its individual components, a standard set of objects are needed. The test objects must be composed of objects with similarities but also some distinct differences to test the recognition algorithm and demonstrate the capabilities of the method. The pieces must also be of a size and shape to be handled by the robotic arm. For these reasons chess pieces were chosen. In figure 5.9 all of the chess pieces are shown with their different profiles from different angles. In image (a) and (b) is the bishop, (c) and (d) is of the castle, next is the king in (e) and (f) followed by the rook in (g) and (h), then the pawn in (i) and lastly the queen in (j) and (k). Each of the test pieces is available in a dark or light brown colour. The size of the pieces varies from about 4cm high with 1.8cm diameter for the pawn to about 5.5cm high with 2.1cm diameter for the king. All of the other pieces fall between these values.

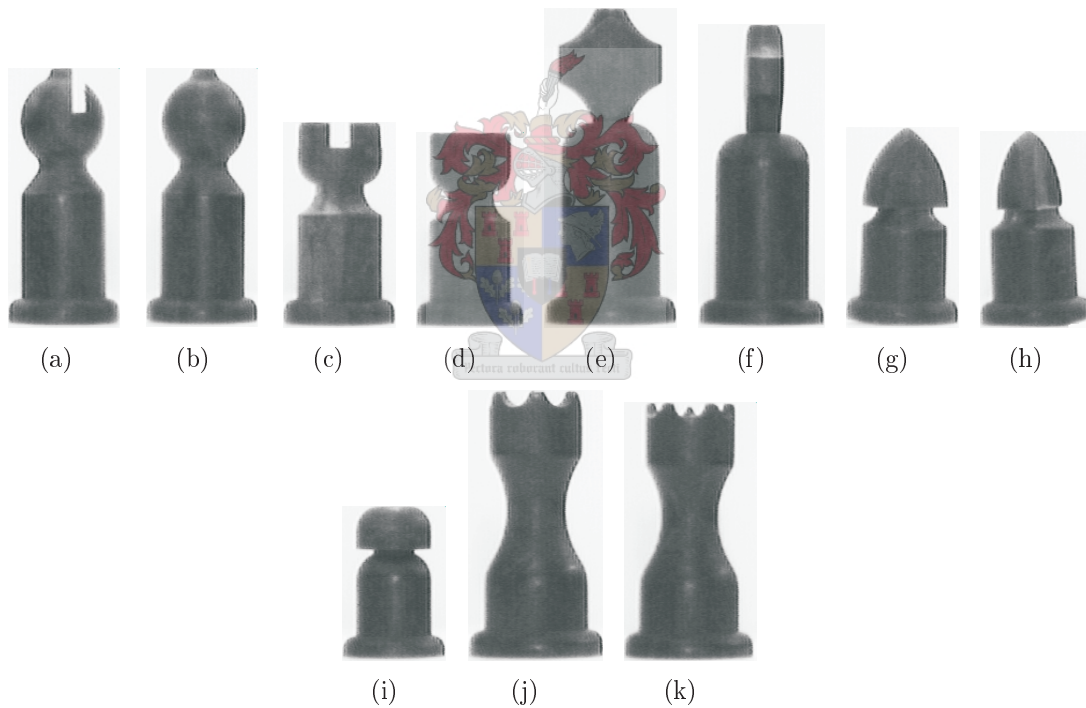


Figure 5.9: Pieces to be recognised from different angles.

5.4.3 Guard System

The objective of the guard algorithm is to optimise the recognition and handling process by reducing the recognition process time. This is done by avoiding the computation intensive Fourier descriptor process for boundaries that are definitely not known to the system. Considering the height, width, circumference, shortest and longest distances from the centre of the boundary to the boundary the algorithm decides if an object falls into the chess pieces class and if the objects orientation allows it to be handled. When an objects boundary is received by the algorithm the boundary is scaled to have a set height from where the width, circumference, shortest and longest distance from the centre is calculated. The orientation of the object is found easily by looking at the height to width ratio. If a piece is not standing, handling will not be possible and it is designated to be invalid. For an object to belong to the chess piece class, its dimensions properties must fall within the boundaries stated in table 5.2 when scaled to a height of 200 pixels. Furthermore,

70	<	Width	<	140
525	<	Circumference	<	700
15	<	Shortest distance	<	50
100	<	Longest distance	<	200

Table 5.2: Valid regions for dimension properties to fall between to belong to the chess piece class.

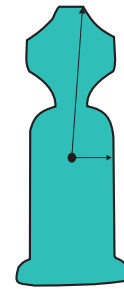


Figure 5.10: Object boundary with arrows showing the longest and shortest distances from the centre to the boundary.

when the values for the width and shortest distance to the centre of an object are plotted against each other, the resulting point must fall within one of the regions summarised in table (5.3) and shown in figure (5.11). In the figure some sample valid points are shown within the regions and some invalid points outside. These regions are defined by equation 5.4.1.

$$R^2 = (w_p - w_c)^2 + (d_p - d_c)^2 < \text{radius}^2 \tag{5.4.1}$$

In this equation the root of R^2 is the distance from the point (w_p, d_p) , the width

<i>Region centre</i> (w_c, d_c)	<i>radius</i> ²
(100,23)	40
(82,30)	53
(112,38)	59
(135,49)	65
(131,23)	70

Table 5.3: Table of valid regions with centre point and radius of each region.

and the shortest distance of the current boundary, to the middle of a region with centre (w_c, d_c) . If R is smaller than the radius of a region, the boundary belongs to a valid object. Other kinds of object classes will have different bounding values and different valid regions.

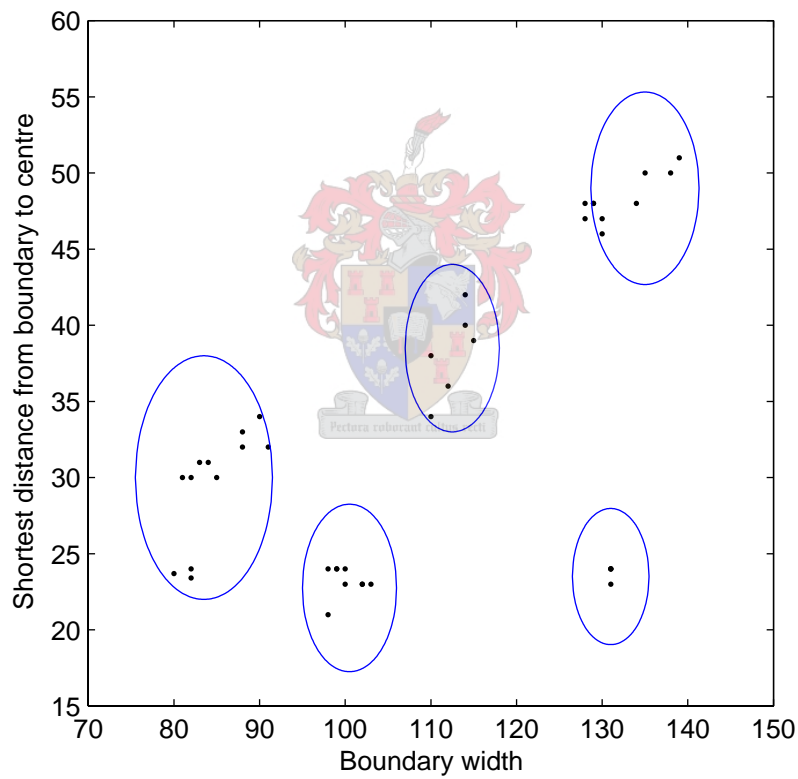


Figure 5.11: Valid regions with sample points of width vs. shortest distance to centre.

5.4.4 Using Fourier Descriptors for Recognition

The Fourier descriptors of a boundary uniquely describe it. Object recognition takes place by comparing the Fourier descriptors of known and unknown objects' boundaries with each other. The first step in building a reliable algorithm with Fourier descriptors is to calculate and index the descriptors of known objects for comparison later. The indexed or calibration objects for this project are shown in figure 5.9 and the Fourier descriptors of these objects are calculated using the methods described in chapter 3. The digital boundaries of the objects are found, interpolated to set number of points, in this case 1000 points, and the discrete Fourier transform are performed on the boundaries resulting in their Fourier de-

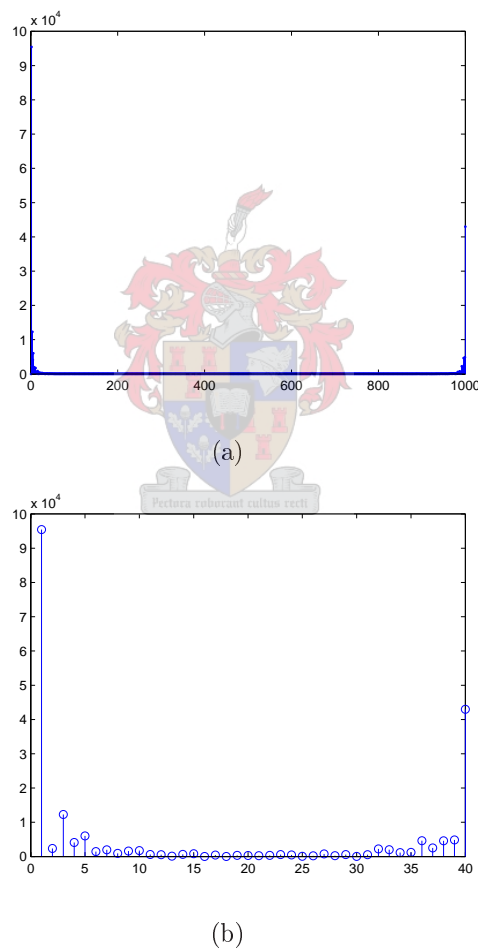
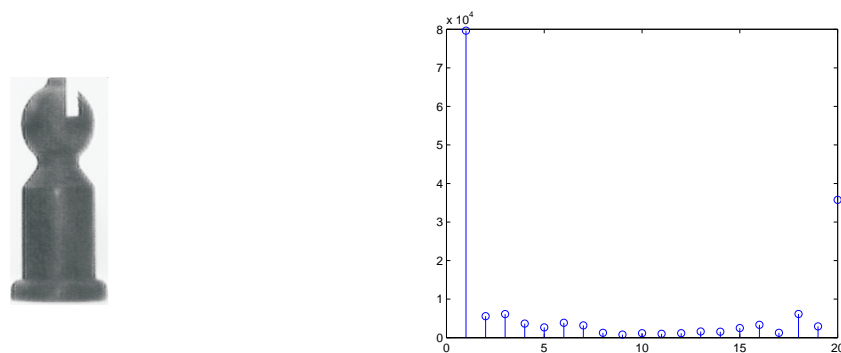


Figure 5.12: Full Fourier descriptors sets of the king, (a), and subset containing 40 lowest frequency components, (b).

criptors. As discussed in chapter 3, a smaller subset of the descriptors is sufficient for comparison. Figure 5.12(a) is of all the descriptors for the king and it is clearly visible that the contribution of most of the middle components, the higher frequency components, is negligible. Figure 5.12(b) shows the forty lowest frequency components of the same object, ignoring the less contributing components. Even here the contribution of the higher frequency components is small. Throughout the project the forty lowest frequency components were used for comparison. The amount of components used for comparison varies according to the type of objects that need to be recognised. The right amount can only be found by trial and error. A general rule is to use more descriptors for comparison for objects with much detail, where higher frequency components will play a bigger role, and less descriptors for simple objects with less high frequency components. Looking at a plot of all the descriptors of the objects to be recognised, helps finding a limit from where the higher frequency components play a negligible role.

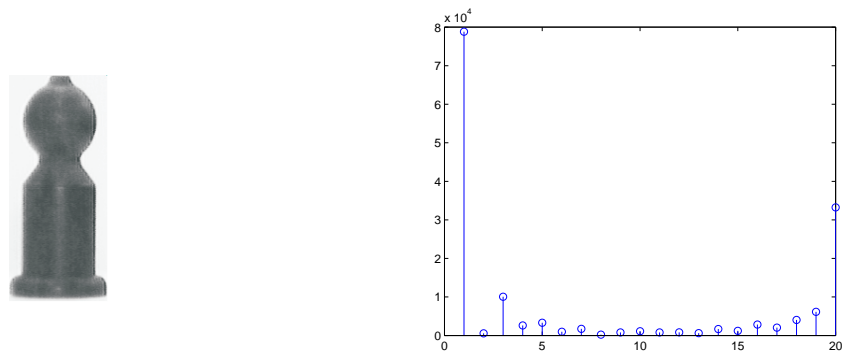
5.4.4.1 Descriptors of Chess Pieces

To illustrate the differences between the Fourier descriptors of the objects, the magnitude of the twenty lowest frequency components with the corresponding objects are plotted in figures 5.13 to 5.16.

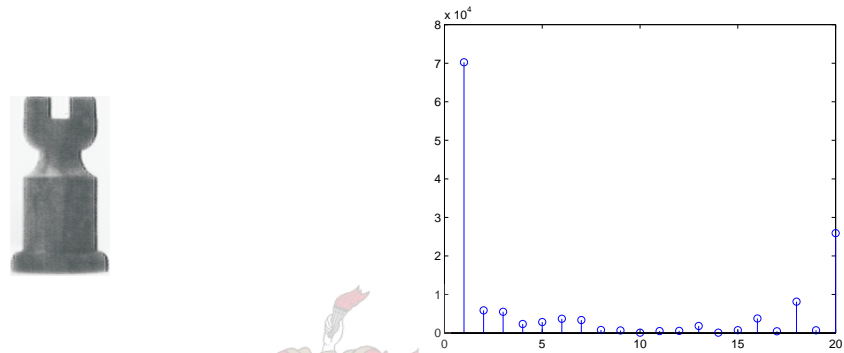


(a) Bishop (angle 1) and its 20 lowest frequency descriptors.

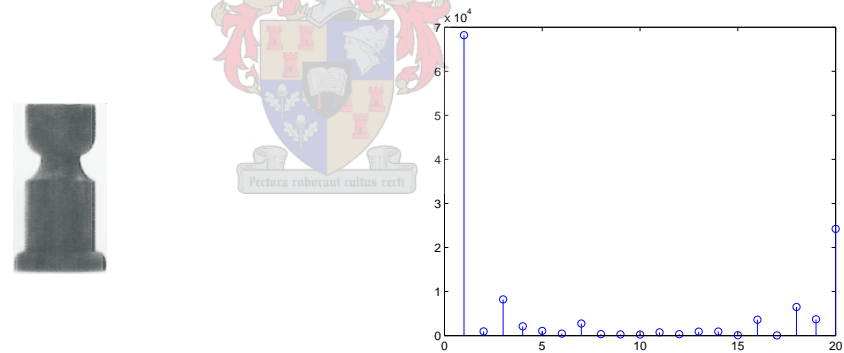
Figure 5.13: Fourier descriptors of a bishop's profile.



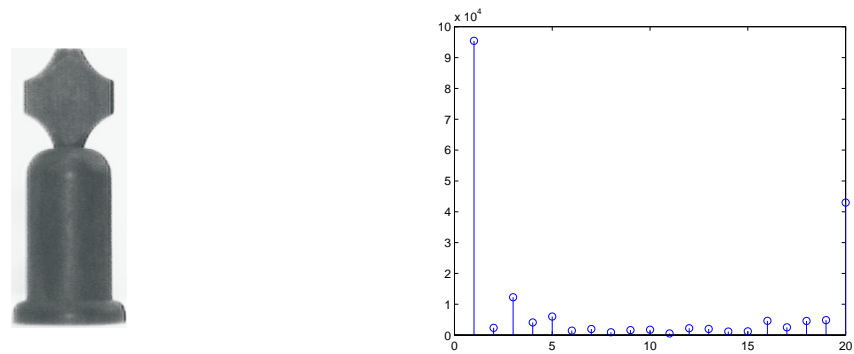
(a) Bishop (angle 2) and its 20 lowest frequency descriptors.



(b) Castle (angle 1) and its 20 lowest frequency descriptors.

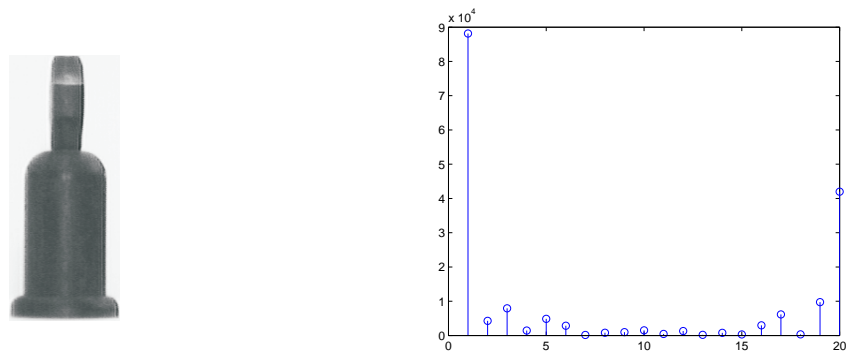


(c) Castle (angle 2) and its 20 lowest frequency descriptors.

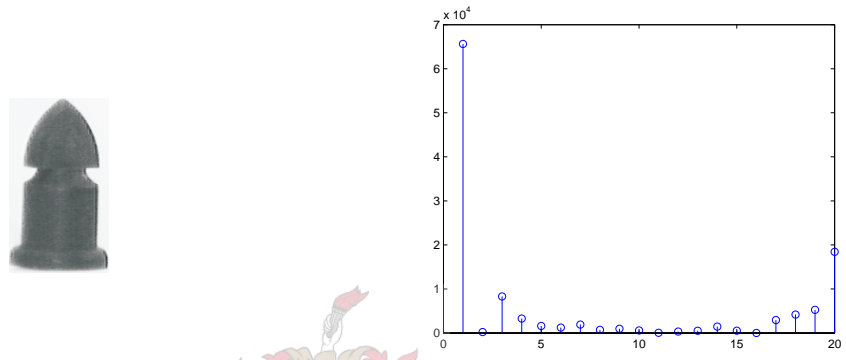


(d) King (angle 1) and its 20 lowest frequency descriptors.

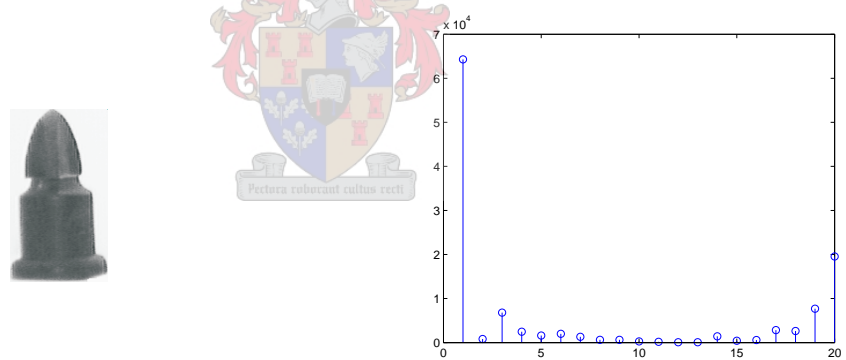
Figure 5.14: Fourier descriptors of a bishop's, castle's and king's profiles from different angles.



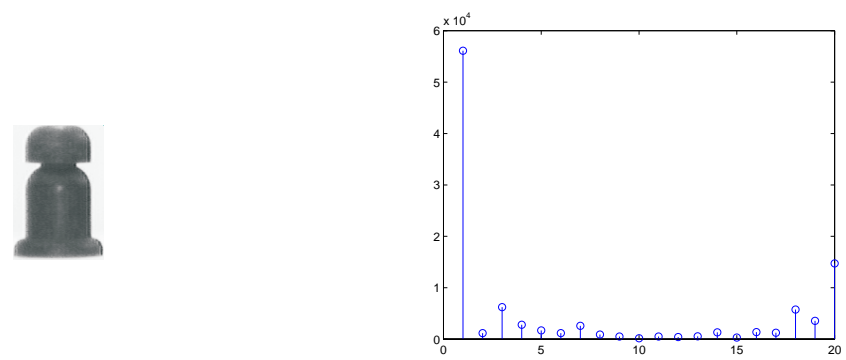
(a) King (angle 2) and its 20 lowest frequency descriptors.



(b) Rook (angle 1) and its 20 lowest frequency descriptors.

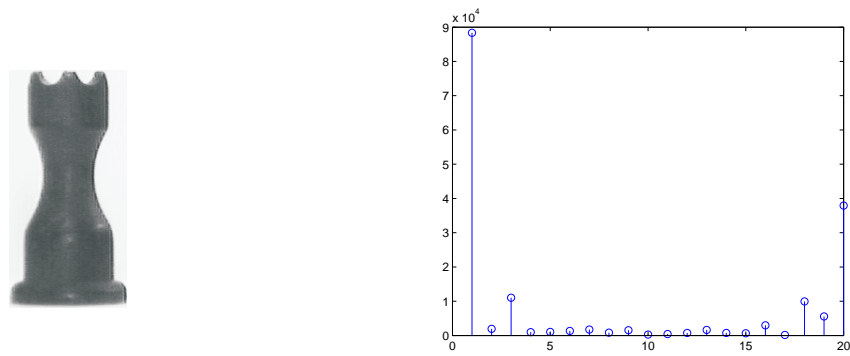


(c) Rook (angle 2) and its 20 lowest frequency descriptors.

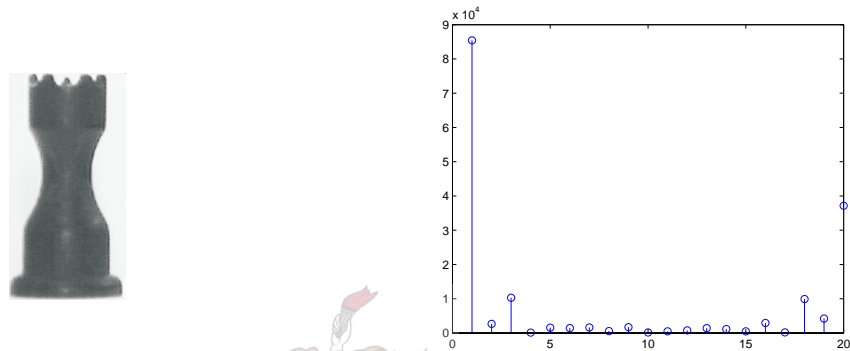


(d) Pawn and its 20 lowest frequency descriptors.

Figure 5.15: Fourier descriptors of the king's, rook's and pawn's profiles from different angles.



(a) Queen (angle 1) and its 20 lowest frequency descriptors.



(b) Queen (angle 2) and its 20 lowest frequency descriptors.

Figure 5.16: Fourier descriptors of the queen's profile from different angles.

The first descriptor, $u(0)$, responsible for the translation of an object is usually very large. It has been removed in figures 5.13 to 5.16 to plot the remaining descriptors to a scale allowing the magnitudes of the smaller descriptors to be more clearly visible. The similarities between descriptors of the same objects photographed from different objects is evident, especially in those of the queen, where the profile of the piece does not change much with a change in perspective.

5.4.4.2 Comparison of indexed objects

The difference between the descriptors of unknown and calibration images is determined by using the Euclidean metric function. If the difference between the two descriptor arrays is below a chosen threshold, then the unknown object must be the same as the one it is compared to. Choosing $M = 40$ significant descriptors, the difference between the calibration descriptor array, a_c , and the unknown object's

descriptor array, a_t , is

$$err = \|\vec{a}_c - \vec{a}_t\| = \sqrt{\sum_{u=1}^M (a_c(u) - a_t(u))^2} \quad (5.4.2)$$

The arrays a_c and a_t comprise only of the significant descriptors; u starts at one as the first place in the array, zero, holds $a(0)$ which is influenced by translation. Other variances must be removed as described in chapter 3 before comparison can take place. Using equation 5.4.2 to compare the calibration images with each other, knowledge regarding their similarities and suitable thresholds for recognition are gained. Table 5.4 shows the result of this operation.

	<i>B1</i>	<i>B2</i>	<i>C1</i>	<i>C2</i>	<i>K1</i>	<i>K2</i>	<i>R1</i>	<i>R2</i>	<i>P</i>	<i>Q1</i>	<i>Q2</i>
<i>B1</i>	0	8846	7374	9380	9284	12464	13488	12473	11512	9850	8793
<i>B2</i>	8940	0	10944	6736	5432	10080	9942	8755	9835	7454	7908
<i>C1</i>	8361	12279	0	7880	14563	18578	11744	11894	8548	11475	10567
<i>C2</i>	10958	7787	8118	0	12213	17240	7634	8077	6086	7864	8298
<i>K1</i>	4479	4486	10719	8726	0	9845	12385	11493	11564	8867	8635
<i>K2</i>	11261	9011	14802	13334	10658	0	14903	12256	14419	14323	13949
<i>R1</i>	16369	11939	12569	7931	18009	20019	0	3996	3755	15418	15830
<i>R2</i>	15454	10734	12996	8567	17062	16807	4079	0	5821	15042	15518
<i>P</i>	16339	13813	10700	7394	19666	22651	4392	6688	0	15786	15916
<i>Q1</i>	8636	6649	9123	6069	9577	14290	11453	10944	10026	0	2395
<i>Q2</i>	8197	7295	8688	6622	9645	14391	12160	11676	10454	2477	0

Table 5.4: Calibration objects Fourier Descriptor comparison. Smallest difference to object of another type in bold.

5.4.4.3 Determining the Thresholds

The differences between objects vary greatly. Thus, choosing a separate threshold for each of the calibration images will increase reliability. Taking the first calibration bishop for example, the smallest difference is 7749 when compared to the first castle. Picking a threshold of 25% of the smallest difference to another image, 1800, will ensure accurate recognition of the bishop even under imperfect conditions. Choosing a similar threshold of 25% for the pawn will lead to problems as the smallest difference to its descriptors is only 3755 when compared to the first

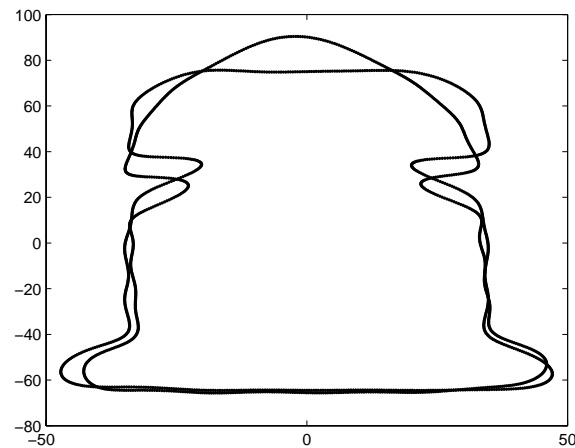


Figure 5.17: Comparison between the boundaries of the first rook and the pawn.

rook. A reliable threshold for the rook can still be found at 50% of the smallest difference, but the chance of a wrong recognition for the rook is greater than that for the bishop. The small difference between the rooks and the pawn can be explained when we look at their boundaries shown in figure 5.17. The objects share a very similar boundary, with only a large difference at the top.

The smallest difference in table 5.4 is between the two queens. Mistaking the first queen for the second calibration queen does not affect the performance of the system and the small difference between these two can be ignored. In fact, mistaking any object for the same object viewed from another angle does not affect the performance of the system. Using the smallest difference to a another boundary that is not of the same type as a guide line, thresholds for each of the calibration object boundaries can be chosen as in table 5.5. The percentage of the smallest difference to the boundary is shown in the margin column. A lower percentage increases the likelihood of a correct recognition.

The choice of thresholds is further influenced by the average difference when compared to images of objects taken in non ideal environments. For example, the difference between the calibration pawn and a pawn in an imperfect environment is always relatively small. The same is not true for more complex objects such as the

<i>Object</i>	<i>Threshold</i>	<i>Margin</i>
Bishop (angle 1)	2200	29%
Bishop (angle 2)	1800	40%
Castle (angle 1)	2200	30%
Castle (angle 2)	2000	33%
King (angle 1)	1900	35%
King (angle 2)	2500	25%
Rook (angle 1)	2000	50%
Rook (angle 2)	2000	50%
Pawn	1900	51%
Queen (angle 1)	2400	32%
Queen (angle 2)	2400	30%

Table 5.5: Thresholds for each of the calibration boundaries.

king. This is due to the change of the visible boundary of an object under different light conditions. Taking this in consideration the threshold for the various images can be fine tuned by trail and error.

5.4.4.4 Recognition Example

Using the Fourier descriptors of an object with these thresholds for recognition is illustrated in the following example. The boundary of the object in figure 5.18(a) is shown in figure 5.18(b). The Fourier descriptors of the boundary is calculated and the 20 lowest frequency coefficients, without the large $a(u)$ responsible for translation, are shown in figure 5.19. The difference between the 40 lowest frequency components of this boundary and each of the indexed calibration boundaries are

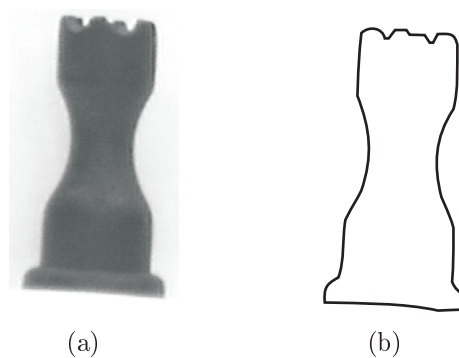


Figure 5.18: Object for comparison, (a), and its resulting boundary (b).

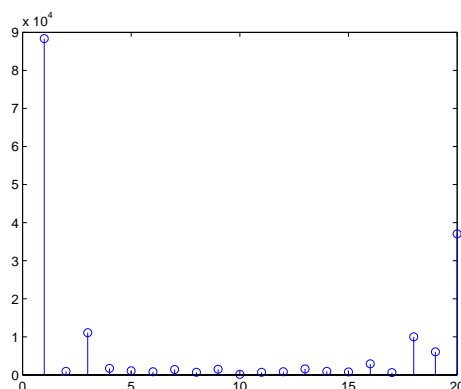


Figure 5.19: The 20 lowest frequency components of the example queen's boundary.

<i>B1</i>	<i>B2</i>	<i>C1</i>	<i>C2</i>	<i>K1</i>	<i>K2</i>	<i>R1</i>	<i>R2</i>	<i>P</i>	<i>Q1</i>	<i>Q2</i>
9337	6164	9566	5448	9707	14507	10684	10261	9478	1996	3496

Table 5.6: Difference between descriptors of the test object and each of the calibration objects.

calculated and are shown in table 5.6. The lowest difference, 1996, are to the first calibration queen. The difference is also below the threshold for the queen and the boundary is defined as that of a queen chess piece.

5.4.4.5 Limitations

The Fourier descriptor method for object recognition does hold limitations. The most important one for this project's purposes comes from the overlapping of objects' silhouettes. If two objects are placed close together or if both of them fall in the same line of sight, recognition of the individual objects will not be possible. Figure 5.20 illustrates the problem. When the silhouettes of objects overlap, the resulting boundary that is calculated and sent to the object recognition algorithm is that of the combined boundary for both pieces. This boundary will be interpolated and its Fourier descriptors will be calculated. The difference between the descriptors of the combined boundary and those of the indexed calibration boundaries will be large as the boundary's shape differs a lot from all of them. The algorithm will label the boundary as unknown even though both pieces contributing to the boundary are known. Plotting the differences in the 40 lowest frequency descriptors of a

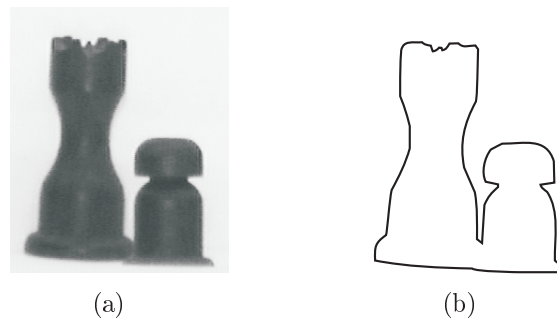
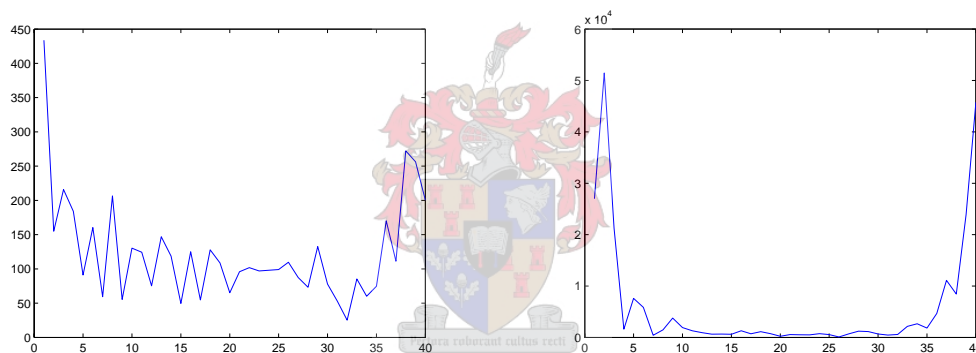


Figure 5.20: Two chess pieces in the same line of sight, (a), and the resulting boundary (b).

test queen and one of the indexed calibration queen in figure 5.21(a) illustrates the small differences between the descriptors. Comparing this to the plot in differences for the combined boundary and the same indexed queen in figure 5.21(b), it is clear that it is not possible to recognise individual pieces in combined boundaries.



(a) Difference between Fourier coefficients of a queen-queen comparison.

(b) Difference between Fourier coefficients of the queen-combined boundary comparison.

Figure 5.21: Differences in Fourier descriptors.

5.4.5 Colour Identification

The chess pieces do not only differ in shape but also in colour. An algorithm is needed to identify the colour of a recognised chess piece. As the chess pieces are either of light or dark brown colour, as shown in figure 5.22, the problem is simplified. The colour of an object can be determined by taking the average value

of the pixel colour values inside the boundary of an object. The average value will be between 0 and 255, as it is a greyscale image where 0 represents black and 255 white. Differentiation between colours is done by placing a threshold in the middle of the expected average colour values for light and dark brown pieces. If the average value is below the threshold, the piece is dark brown, and if the value is above the threshold it is light brown. Expected colour values close to the extremes, 0 or 255, can be achieved by increasing the contrast of the image, making colour identification more reliable as expected values are removed further from the threshold.

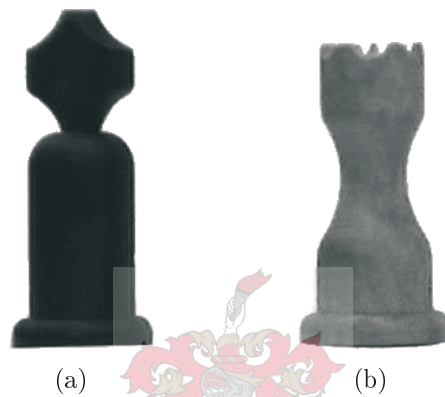


Figure 5.22: Chess pieces with different colours, a black king, (a), and white queen (b).

5.4.6 Position Calculation

To enable a robotic arm to handle recognised objects, it is necessary to determine the position of the objects relative to the robotic arm. Various techniques exist to calculate the positions of objects in images, including 3D reconstruction and stereoscopy. 3D reconstruction is more computational intensive than stereoscopy. Complex calculations is made resulting in unnecessary information for the purposes of this project. Stereoscopy only finds the position of an object and was used.

With more than one camera and basic geometry, the position of an object is calculated through stereoscopy and simple surveying techniques. This is done by using a calibration point and setting up the camera and work area as illustrated in figure 5.23. Extracting the angles φ_1 and φ_2 from the images, the x and y position

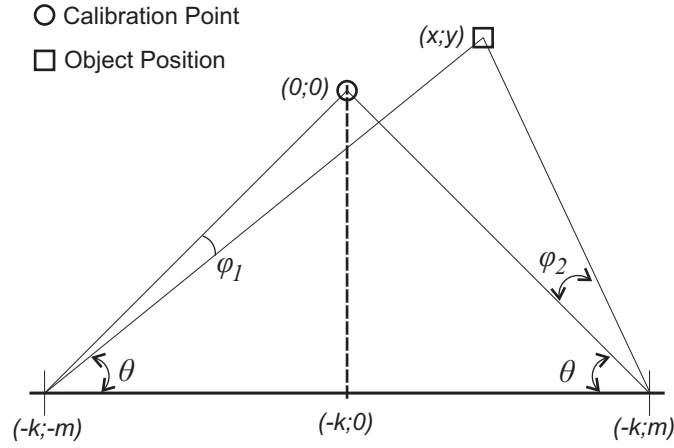


Figure 5.23: Environment setup for position calculations.

of the object can be found in relationship to the cameras. This position is converted to be relative to the robot knowing the placement of the cameras relative to the robot. The equations used to calculate x and y in figure 5.23 are deduced as follows. From figure 5.23 we find:

$$\theta = \tan^{-1} \frac{k}{m} \quad (5.4.3)$$

Defining ϕ_1 and ϕ_2 as

$$\phi_1 = \theta - \varphi_1 \quad (5.4.4)$$

$$\phi_2 = \theta + \varphi_2 \quad (5.4.5)$$

we have:

$$\phi_1 = \tan^{-1} \frac{k + y}{m + x} \quad (5.4.6)$$

$$\phi_2 = \tan^{-1} \frac{k + y}{m - x} \quad (5.4.7)$$

Making x the subject in equation 5.4.6 and y the subject in equation 5.4.7 we get:

$$x = \frac{k + y}{\tan \phi_1} - m \quad (5.4.8)$$

$$y = (m - x) \tan \phi_2 - k \quad (5.4.9)$$

Substituting x in equation 5.4.9 with equation (5.4.8) we find y as:

$$y = \frac{2m \tan \phi_2 - k \left(\frac{\tan \phi_2}{\tan \phi_1} + 1 \right)}{\frac{\tan \phi_2}{\tan \phi_1} + 1} \quad (5.4.10)$$

Using equation 5.4.8 and 5.4.10 the values of x and y can be found and the position of an object relative to the cameras will be known.

5.4.6.1 Extracted Angles

The angles φ_1 and φ_2 are found from the amount of pixels between the center of an object in an image and the calibration point in the image. The center of an object is calculated from the first descriptor, $a(0)$, of the boundary, as explained previously. The amount of pixels between the centre of the boundary and the calibration point is converted to a specific angle, knowing roughly the distance between the camera and the object. This amount of pixels is calculated for each of the objects in both of the images, converted to angles and the position calculated. A constant conversion factor between pixels and angles was used, leading to some inaccuracy in the calculated position. With the non-linearity of the camera lenses and varying distance between the object and the cameras, the true pixels to angle conversion factor changes constantly. The constant factor that is used is an average of the true conversion factors encountered.



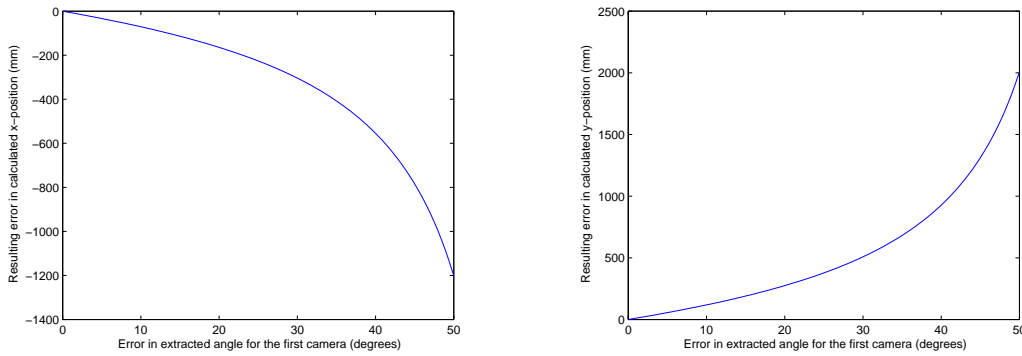
5.4.6.2 Sensitivity

The sensitivity of the position calculation algorithm to errors in the extracted angles is a measurement of the method's usability for this project. Rewriting equations 5.4.4 and 5.4.4 to include errors Δ_1 and Δ_2 , the measurement errors on the extracted angles, and using equations 5.4.8 and 5.4.10 the effect of the errors on the positions can be calculated.

$$\phi_1 = \theta - (\varphi_1 + \Delta_1) \quad (5.4.11)$$

$$\phi_2 = \theta + (\varphi_2 + \Delta_2) \quad (5.4.12)$$

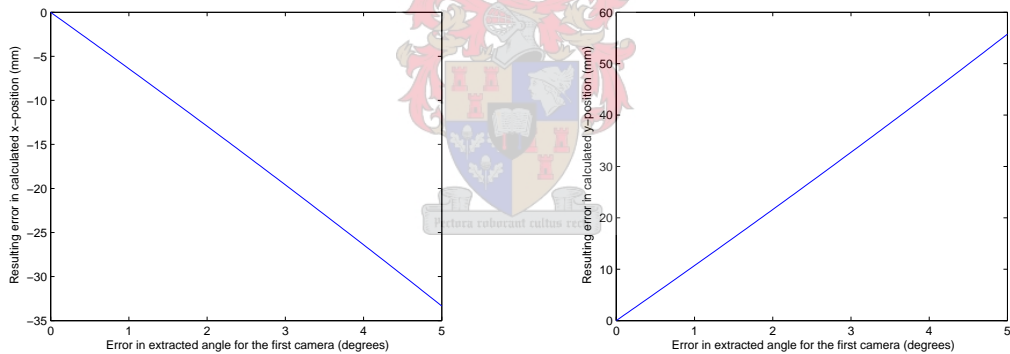
Assuming an error on only one of the extracted angles for a object standing at the calibration point, the error on the x and y positions are calculated and shown in figures 5.24(a) and 5.24(b). It is clear that the error on the calculated position does



(a) Error in x-position for Δ_1 ranging from 0° to 50° . (b) Error in y-position for Δ_1 ranging from 0° to 50° .

Figure 5.24: Position errors due to errors in the extracted angle.

not relate to the extracted angle error in a linear way. A maximum error of 50° is however very unlikely. Assuming a maximum error of 5° , still relatively large and unlikely, the relationship between position error and extracted angle error can be approximated to a linear function, as figures 5.25(a) and 5.25(b) illustrates.



(a) Error in x-position for Δ_1 ranging from 0° to 5° . (b) Error in y-position for Δ_1 ranging from 0° to 5° .

Figure 5.25: Position errors due to errors in the extracted angle.

From the figures it is approximated that for every error of 1° in the extracted angle the calculated x-position will differ 6.67 mm. from the actual x-position. The y-position will differ 11.15 mm. from the actual y-position for every 1° error. The calculated y-position is almost twice as sensitive as the x-position for an error in the

extracted angle. Expected errors due to a slight incorrect pixels to angle conversion factor are low, below 0.25° . Small errors on both of the extracted angles will either cancel each other out or increase the sensitivity of the algorithm depending on their direction. With small errors this method for position calculation still provides enough accuracy for handling and can be used.

5.4.6.3 Limitations

Stereoscopy presents some limitations when the objects in the images are not unique. Figure 5.26 shows the sorting area from above, with three objects represented by \circ , \square and \triangle and the positions of the cameras. Looking at an image taken by camera 1 from left to right the sequence of the objects will be $\circ \square \triangle$. For camera 2 the same sequence will differ and will be $\square \circ \triangle$. If the objects \circ , \square and \triangle are unique by shape or colour their position can be calculated without any problem, but if this is not the case, stereoscopy will not be able to calculate their positions. The problem is with the extraction of the angles needed for the position calculations. If the \square and \triangle are both black pawns there is now way to know which angle from camera 1 correspond to which angle in camera 2. The only way to guarantee reliable results from the stereoscopy algorithm is to ensure that all the objects in the sorting area are unique.

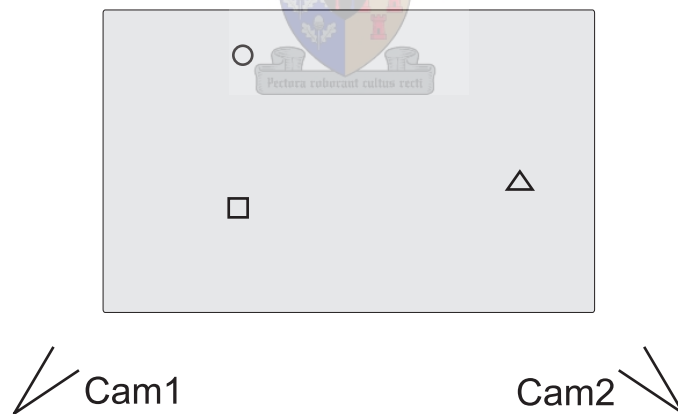


Figure 5.26: Illustrating limits of stereovision algorithm.

5.5 Handling

The last step in the sorting process is the handling of the objects in the sorting area. To handle an object the manipulator is moved from its resting position, 230 mm. above the calibration point, to directly above the specific object. The robotic arm is then lowered with the gripper open until the object is in the middle of the open gripper. The gripper is then sufficiently closed to grab the object and the arm is raised straight up to a set height. From here the arm is rotated at this height to the positions in the sorted area for the specific piece where it is lowered into its position or dropped into a container. The arm returns to its resting position, from where the same process starts for the next piece.

Moving the manipulator to a new position involves deciding on a wrist yaw, pitch and roll angle and calculating the inverse kinematics function for the remaining two

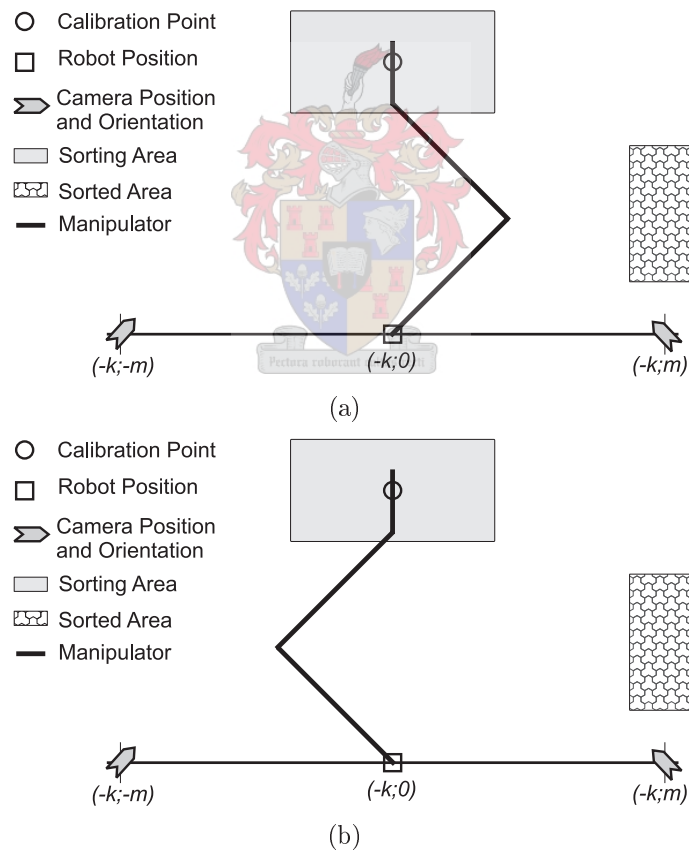


Figure 5.27: Different solutions to the inverse kinematic equation.

joints. For the inverse kinematics function for the RTX there will be two solutions, a right and a left arm solution as illustrated in figure 5.27. Throughout the left arm solution in figure 5.27(b) was used as it is the only possible solution for the RTX in the sorted area. Keeping with the left arm solution reduces movement time, as moving from a right arm pose in the sorting area to a left arm pose in the sorted area takes more time as bigger angle movements at the joints are needed.

Deciding on the wrist yaw, pitch and roll angles involves looking at the position of the manipulator when objects are picked up or put down. To increase the range of the robotic arm in the sorting area, the yaw angle is kept at 0° relative to the base coordinate system of the robot, as illustrated in both figures in 5.27. At the sorted area the yaw angle is at -90° , as shown in figure 5.28, to increase the reach within this area. The roll of the wrist is always kept at 0° . The pitch of the wrist is influenced by the chosen strategy for picking up recognised pieces, while the roll is kept at 0° throughout.

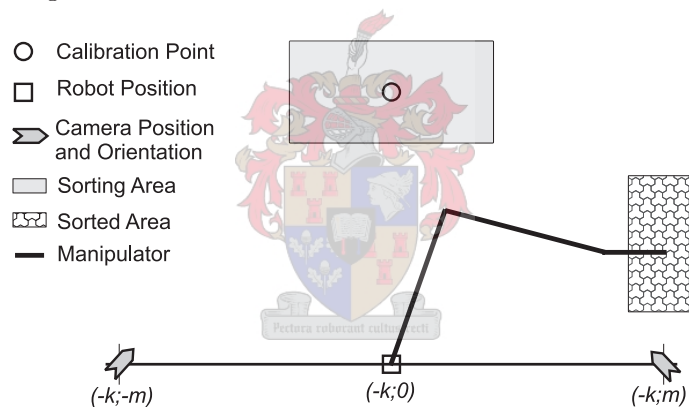


Figure 5.28: Manipulator pose in sorted area.

The simplest and quickest strategy for handling is to assume that all of the objects in the area have been identified and to start with the object closest to the robot and working towards the back of the sorting area. For maximum grip the pitch of the gripper is at 0° , level with the surface of the area. This strategy works well for the ideal situation, but provisions must be made for other situations. Taking in consideration that all of the objects in the area might not have been recognised

or that an operator may wish the objects to be handled in another sequence the following handling strategies have been devised:

- Strategy 1: Handling sequence starts at the front of the sorting cell and works towards the back of the sorting cell assuming all the pieces have been recognised.
- Strategy 2: Handling sequence starts at the front of the sorting cell and works towards the back of the sorting cell assuming unknown objects are in the area.
- Strategy 3: Handling starts at the back of the sorting cell and works towards the front.
- Strategy 4: Operator chooses a handling sequence manually.
- Strategy 5: Operator chooses to handle only a specific piece.

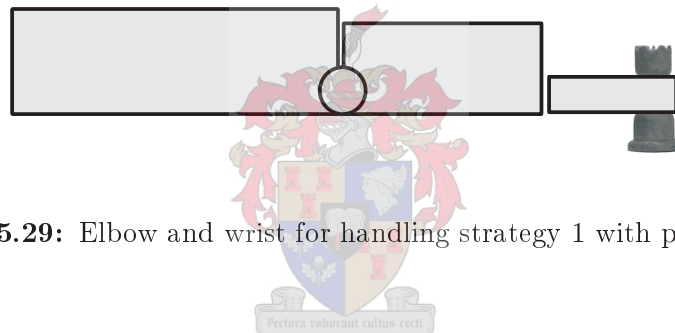


Figure 5.29: Elbow and wrist for handling strategy 1 with pitch at 0° .

Strategy 1 assumes every object has been recognised. This has already been explained in a previous paragraph of this section. Strategy 2 is a development of strategy 1 but makes provision for unidentified objects. Objects might not be recognised due to defects in the objects, lighting conditions or simply because they are not chess pieces. In such a situation, handling objects from the front of the sorting area with the gripper level with the surface will cause problems. If an unidentified object is under the base of the wrist when the arm moves down to pick up another piece, the unidentified object might be knocked over, causing the position of other objects to change. Or the robotic arm may not be able to move down sufficiently to grip a required object as the unidentified object may cause an obstruction. Changing the pitch of the gripper from level with the surface to -60° ,

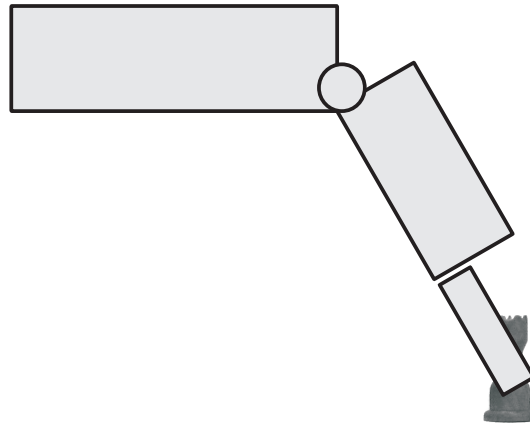


Figure 5.30: Elbow and wrist for handling strategies with pitch at -60° .

as shown in figure 5.30, will keep the base of the gripper above other objects that may be in the way, avoiding knocking them over. The pitch angle is maintained throughout the rest of the handling manoeuvres.

Strategies 3 and 4 are similar to strategy 2. The pitch of the gripper is kept at -60° to avoid knocking over other pieces, but the sequence in which objects are handled are different. Strategy 3's sequence is the opposite of strategy 2 while strategy 4 can be any sequence as defined by the operator. Strategy 5 allows the operator to choose a specific piece leaving the rest behind, also using a pitch angle of -60° .

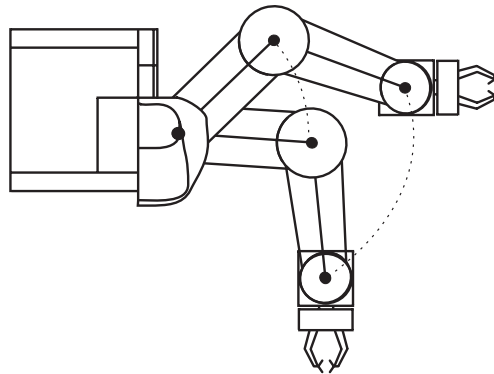


Figure 5.31: Movement of the robot arm from the sorting to sorted area.

Chapter 6

Results

The performance of the automated sorting cell can be measured through the performance of the subsystems and the performance of the integrated system. The subsystems include the guard, shape recognition, colour identification and positioning systems and were tested in the SENROB environment using the chess pieces. The integrated system is tested by evaluating the handling aspects of the sorting cell.

6.1 Guard Algorithm

Testing the guard algorithm is a simple procedure. Boundaries of objects are sent to the algorithm and it decides whether a boundary belong to the chess piece set or not.

Results are presented with a picture of the test object followed by the attributes on which the guard system bases its decision. All attributes are calculated after the object's boundary has been scaled to a height of 200 pixels. Attributes contributing to an invalid classification are shown in bold. First some of the indexed calibration boundaries of the chess pieces under ideal conditions are tested with results in table 6.1. As expected, none of the calibration images produce an attribute outside the allowable regions.




	Width = 108.88 Circumference = 637.47 Shortest distance from centre = 36.88 Longest distance from centre = 112.59 Valid = Yes
	Width = 81.08 Circumference = 586.33 Shortest distance from centre = 30.43 Longest distance from centre = 110.02 Valid = Yes
	Width = 99.16 Circumference = 636.97 Shortest distance from centre = 23.97 Valid = Yes

Table 6.1: Results for the some of the chess piece profiles.

Testing the performance of the algorithm for images taken in a non-ideal setup is important. The first test case is a queen where the camera is slightly out of focus, and the second is a pawn in a non ideal illuminated environment, both shown in figure 6.2. In both of these test cases the objects were still valid. Although the queen is out of focus, its boundary does not change and is still that of a queen, The

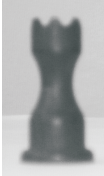
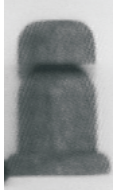
	Width = 98.42 Circumference = 647.03 Shortest distance from centre = 23.70 Longest distance from centre = 110.04 Valid = Yes
	Width = 134.45 Circumference = 642.70 Shortest distance from centre = 49.39 Longest distance from centre = 110.38 Valid = Yes

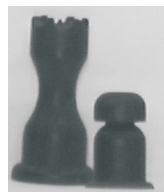
Table 6.2: Attributes for object in non ideal conditions.

boundary of the pawn, however, does change, but not significantly and it is still allowed through to the Fourier descriptors recognition algorithm. Stopping invalid boundaries from being passed to the recognition algorithms is just as important as letting valid objects through.

The most likely situation where this might happen is the combined boundary of overlapping objects in images. As explained in chapter 5, if the silhouettes of objects overlap the resulting boundary will be of the combined silhouettes of the objects, a boundary that will definitely not be recognised. Looking at the following test cases, we see that none of their boundaries are deemed valid by the guard system.



Width = **144.8**
 Circumference = **737.40**
 Shortest distance from centre = **52.39**
 Longest distance from centre = 117.96
 Valid = No



Width = **165.58**
 Circumference = **933.74**
 Shortest distance from centre = **3.8**
 Longest distance from centre = 129.08
 Valid = No

Table 6.3: Attributes for invalid objects.

6.2 Recognition with Fourier Descriptors

Performance with overhead lighting: The recognition algorithm is tested by calculating the differences in the 40 lowest frequency coefficients of test and calibration objects. The thresholds chosen in table 5.5 are then used to determine the accuracy of the algorithm. Accuracy is expressed as the percentage of correct recognitions that have been made. The first set of tests that were run, involved all of the chess pieces under normal conditions in SENROB using only the overhead lights for illumination. Table 6.4 holds the results.

Object	$B1$	$B2$	$C1$	$C2$	$K1$	$K2$	$R1$	$R2$	P	$Q1$	$Q2$	Classification
<i>White bishop</i>	7203	1384	9104	6132	4732	9522	8534	7793	8884	6599	7039	bishop
<i>White bishop</i>	7426	1445	9285	6099	4806	9809	5541	7896	8924	6702	7159	bishop
<i>Black bishop</i>	7281	1630	9239	6338	4824	9319	8605	7765	8983	6692	7139	bishop
<i>Black bishop</i>	7327	1505	9265	6268	4587	6519	8628	7891	8987	6681	7177	bishop
<i>White castle</i>	8365	5711	6351	794	10280	15986	6839	7580	5849	6768	7223	castle
<i>White castle</i>	8608	5945	6355	502	10595	16310	6641	7463	5609	6953	7372	castle
<i>Black castle</i>	6200	8961	1682	5913	12033	16598	9496	9836	7233	8737	8402	castle
<i>Black castle</i>	8998	5779	6837	988	10732	16136	6606	7214	5694	7161	7604	castle
<i>White king</i>	6425	4589	9160	7769	1822	9125	10501	9956	10278	7417	7570	king
<i>White king</i>	6715	4609	9444	7927	996	8988	10543	10079	10392	7848	7995	king
<i>Black king</i>	9284	7232	12073	11130	7596	3001	11891	10141	12110	11391	11496	(king)
<i>Black king</i>	6670	5028	9317	8091	1281	9402	10717	10254	10451	7884	8005	king
<i>White rook</i>	11737	8228	10286	8166	14478	14792	4598	2118	5766	12985	13491	(rook)
<i>White rook</i>	11311	7414	9581	6575	13590	15829	2323	2770	4396	12067	12590	(rook)
<i>Black rook</i>	12909	9415	13133	9001	15782	16108	4412	2299	5779	14203	14783	(rook)
<i>Black rook</i>	12887	9343	10315	7430	15871	18361	1807	3118	3514	13511	14103	rook
<i>White pawn</i>	14613	12445	10219	8335	19210	22554	4479	6173	1616	15548	16023	pawn
<i>White pawn</i>	13695	11461	9348	7150	17973	21795	4104	6089	650	14384	14815	pawn
<i>Black pawn</i>	14118	11934	9752	7700	18605	22161	4372	6101	1306	14891	15364	pawn
<i>Black pawn</i>	13578	11665	9437	7713	18085	21459	4524	5967	1939	14570	15060	(pawn)
<i>White queen</i>	7884	6068	8612	6278	8364	13252	10494	10197	9743	2045	2385	queen
<i>White queen</i>	8180	5892	8842	6053	8444	13498	10221	9990	9590	2383	3137	queen
<i>Black queen</i>	7925	5731	8324	5612	8605	13498	9915	9618	9172	2234	3003	queen
<i>Black queen</i>	7600	5778	8074	5654	8590	13461	10044	9804	9227	1374	2234	queen

Table 6.4: Fourier descriptor coefficient differences for test objects under normal conditions. Smallest difference in bold.

<i>Object</i>	<i>Threshold</i>	<i>Margin</i>
Bishop (angle 1)	2200	29%
Bishop (angle 2)	1800	40%
Castle (angle 1)	2200	30%
Castle (angle 2)	2000	33%
King (angle 1)	1900	35%
King (angle 2)	2500	25%
Rook (angle 1)	2000	50%
Rook (angle 2)	2000	50%
Pawn	1900	51%
Queen (angle 1)	2400	32%
Queen (angle 2)	2400	30%

Table 6.5: Thresholds for each of the calibration boundaries.

For easy comparison, the thresholds first defined in table 5.5 are repeated in table 6.5. With these thresholds, and looking at table 6.4, an accuracy of 79.16% was achieved for these results. The classification column of table 6.4 holds the identity of the indexed piece with which the difference is below the threshold, if not the identity for the indexed piece with the lowest difference is shown in brackets. The objects whose smallest difference was not below the particular threshold, were a king, three rooks and a pawn.

The problem with the recognition of the king can be explained by looking at its image in figure 6.1. The rotation of the object results in a boundary not similar



Figure 6.1: Image of unrecognised king angle.

to those in the indexed calibration set. Two options exist to ensure that a king with this rotation is recognised. Either the thresholds for the calibration kings are increased in order for the difference to be below them, or an extra calibration boundary is added which correlates with this rotation. Seeing that the smallest difference between the second calibration king and another type of object is 9845 in table 5.4, raising the threshold for this calibration boundary is a viable option. A threshold of 3200 or 32% will ensure that this profile of the king will be recognised. Raising the threshold increases the chances that another type of object might be recognised as a king. Adding another calibration boundary will increase the reliability of the algorithm without changing the thresholds.

The inability of the algorithm to recognise the rooks and the pawn is also caused by a threshold problem. Thresholds of 2000 (50%) and 1900 (51%) were chosen for the rooks and pawn respectively. At 50% they are the closest to their smallest difference to another object. Coincidentally, the closest boundary for the rooks is the pawn and vice versa. Increasing the threshold might lead to a pawn being recognised as a rook and a rook as a pawn. No other option exists, however, under the current conditions. The shapes of the objects are just very similar. Adjusting the threshold for the rooks to 2400 (60%) and the pawn's to 2000 (53%) increases the accuracy of the current algorithm to 96%, with the king being the only unrecognised object.



Performance with added directional lighting: Changing the conditions under which recognition takes place might improve the algorithm's performance. If the boundaries of unknown objects have more definition, especially those of the rook and pawn, their differences will become more apparent. The easiest way to improve the definition of a boundary is by removing any shadows that may dilute the edges of an object in an image. This is done by adding directional lighting, placed directly above each of the cameras and angled towards the camera calibration point, to the sorting cell. Figure 6.2 demonstrates the significant improvement in boundary definition. The accuracy tests were reconducted and the results are shown in table 6.6.

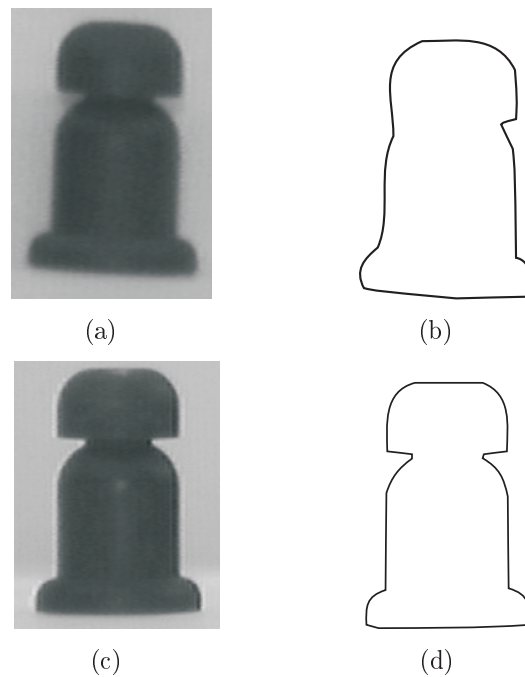


Figure 6.2: Image and resulting boundary with overhead lighting, (a) and (b) and with added directional lighting, (c) and (d).

Looking at the results, we immediately see that the difference between test pawns and the calibration pawn are significantly lower. Throughout, the use and testing of the sorting cell the difference between pawns were always low when directional lighting was added. Knowing this, the threshold for the calibration pawn can be adjusted to a much smaller value of 1000. The differences between the other test and calibration images are also lower, but not to the same extent as those of the pawn comparison. The queen comparison also yields notably better results. The only piece that could not be identified under the improved conditions was a rook. The rook piece is the most complex of all the test pieces, delivering a slightly different profile almost from every angle. Improving the reliability for the recognition of a rook can only be done by adding a number of calibration boundaries to the index of the rook's profile from different angles.

Object	B1	B2	C1	C2	K1	K2	R1	R2	P	Q1	Q2	Classification
<i>White bishop</i>	7517	1294	9270	6186	4891	9398	8315	7548	8770	6908	7371	bishop
<i>Black bishop</i>	7786	1281	9296	6098	4864	9781	8480	7787	8874	6759	7208	bishop
<i>White castle</i>	9356	6371	6766	856	11454	16923	6260	7127	5190	7646	8084	castle
<i>Black castle</i>	8621	5905	6419	1094	10557	16196	6536	7400	5560	7189	7637	castle
<i>White king</i>	6130	4218	8662	7259	1756	9807	10100	9801	9813	7208	7317	king
<i>Black king</i>	7233	4828	9909	8356	1493	8398	10727	10081	10644	8273	8427	king
<i>White rook</i>	12077	9143	9410	6930	15221	18021	2642	4602	3511	13007	13493	(rook)
<i>Black rook</i>	12858	8850	11098	8340	11522	16188	3772	824	5385	13783	14331	rook
<i>White pawn</i>	13780	11763	9321	7368	18153	22116	4234	6534	475	14710	15124	pawn
<i>Black pawn</i>	12948	10883	8725	6603	17116	21191	3950	6162	541	13739	14151	pawn
<i>White queen</i>	7197	6441	7352	5987	8658	13510	10437	10208	9347	1584	1212	queen
<i>Black queen</i>	7803	5985	8012	5634	8714	13688	9985	9762	9144	858	2050	queen

Table 6.6: Fourier descriptor coefficient differences for sorting area under directed lighting. Smallest difference in bold.

6.3 Colour Identification

The colour identification algorithm was tested in normal lighting conditions in SENROB. A threshold of 100 was used to differentiate between the dark and light brown chess pieces. If the average colour value of an object was below the threshold it is black. Above the threshold it is white. A total of 34 tests were run with a result of 100% accuracy. Table 6.7 shows some of the results. Under different lighting the average colour values for the objects will differ and the threshold should be adjusted accordingly.

<i>Object</i>	<i>Average colour value</i>	<i>Classification</i>
White bishop	125	White
White castle	163	White
White king	184	White
Black king	1	Black
Black rook	19	Black
White pawn	177	White
Black pawn	32	Black
Black queen	40	Black

Table 6.7: Colour identification results.

6.4 Position Calculation

The performance of the position calculation algorithm is tested by placing a known item at a specific position in the sorting area and comparing it to the calculated position. The castle chess piece was placed at the positions stated in table 6.8 and the resulting positions, also in the table, were calculated. It must be taken into account that the manual placement of the object at a specific position is not 100% accurate. The error in position was calculated as the direct distance between the actual position and the calculated position. Inaccurate placement of the chess piece was to blame for maximum error of 4.47 mm., an extreme when compared to the other values. The average error without the extreme value of 4.47 is 1.83 mm., sufficiently accuracy with which to handle the objects with the robot arm. Taking

<i>Placed Position $\pm (y, x)$</i>	<i>Calculated Position (y, x)</i>	<i>Error in mm.</i>
(-40,-40)	(-41.81,-40.68)	1.93
(-40,-40)	(-41.17,-40.85)	1.44
(-40,0)	(-41.4,-0.07)	1.40
(-40,0)	(-41.10,-0.05)	1.10
(-40,40)	(-40.34,42.23)	2.26
(-40,40)	(-39.80,42.51)	2.52
(-20,0)	(-21.91,-0.06)	1.91
(-20,-20)	(-21.75,-20.52)	1.83
(-20,20)	(-19.85,20.85)	0.86
(-20,20)	(-19.20,22.50)	2.62
(0,-40)	(-1.70,-42.34)	2.89
(0,-40)	(-1.98,-41.03)	2.23
(0,-20)	(-1.18,-21.15)	1.64
(0,-20)	(-1.90,-21.53)	2.44
(0,0)	(0.08,0.10)	0.13
(0,40)	(0.57,40.09)	0.58
(0,40)	(1.81,41.19)	2.17
(20,-20)	(17.90,-23.95)	4.47
(40,-40)	(38.42,-41.10)	1.92
(40,-40)	(37.64,-41.76)	2.94
(40,0)	(37.89,-1.16)	2.41
(40,0)	(38.80,-2.07)	2.39
(40,40)	(40.68,42.10)	2.20
(40,40)	(39.92,40.36)	0.37

Table 6.8: Position algorithm results and accuracy.

into account the small contribution the slight inaccurate placement of the object makes to the error, the accuracy is even better.

The source for the slight inaccuracies lies at the pixels to angle conversion. As an average of the pixels to angle ratio for the whole sorting area is used, the conversion factor will be slightly off for most positions in the area. As mentioned in chapter 5, the pixels to angle ratio is dependant on the distance of object from the camera and the shape of the lens. Getting more accurate results will require using better lenses and restricting the distance from the object to the camera. Both of which is unnecessary as accuracy is sufficient. An inaccurate placement of the cameras will also cause inaccuracies.

6.5 Handling

Handling aspects of the sorting cell are tested in two ways. The first is concerned with the accuracy of the movements of the robot and the second with the functioning of the different handling strategies.

Movement accuracy of the robot was tested by placing an object in the sorting area. The accuracy of the gripper position relative to the object at pick-up, the time from first movement from the resting position until drop-off and the repeatability of the movement are measured. After the object is recognised and its position is calculated, the testing procedure starts. The robot moves from its calibration position to its resting position as defined in chapter 5, 230 mm. above the camera calibration point. From here it moves at the same height to directly above the recognised piece while opening the gripper 40 mm.. The gripper is lowered around the object to about 8 mm. above the surface. The accuracy of the gripper position relative to the object is measured at this point. The object is gripped by the manipulator and raised to a height of 350 mm. from where it is moved at this height to the sorted area. Here it is lowered into the sorted area, but not dropped. The timed period for the movement stops here. To test repeatability, the object is taken back to its original position via the same route and placed. The difference in original position and the placed position acts as a measurement of repeatability. Test results are shown in table 6.9.

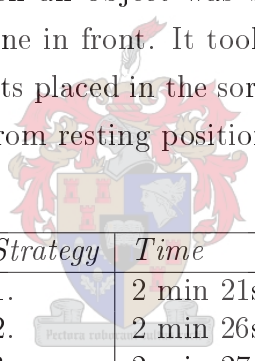
<i>Test</i>	<i>Placed position (y,x)</i>	<i>Calculated position (y,x)</i>	<i>Gripper accuracy (y,x)</i>	<i>Repeatability accuracy (y,x)</i>	<i>Movement time</i>
1.	(-20,-20)	(-20.7,18.3)	(0,1.2)	(0,0)	36s
2.	(-40,40)	(-40.6,36.7)	(0,2)	(0,1)	37s
3.	(-20,-20)	(-23,-21.5)	(1.8,1.5)	(0,2)	37s
4.	(-40,-40)	(-41.4,-41.4)	(0,0)	(0,1)	39s
5.	(0,-40)	(-0.2,-41)	(0,0)	(0,0.5)	40s
6.	(40,0)	(42.4,2.8)	(1,2)	(0,2)	39s

Table 6.9: Handling accuracy results.

From the results we can see that the positioning of the gripper is very accurate when compared to the calculated position. The gripper's placement is dependant

on the calculated position, and the improvement of its accuracy relative to an object is directly connected to the improvement of the positioning calculations accuracy. For any practical use the current accuracy for both the gripper positioning and the positioning calculation is sufficient. Movements are repeated accurately. Every object was returned to its original position with errors only due to the original gripper position being inaccurate at pick-up, slightly moving an object while closing the gripper.

Each of the different strategies was tested by placing multiple objects in the sorting area and sorting them, each time using a different strategy. When there were no unrecognised pieces the first strategy worked flawlessly, handling three pieces in 2 minutes and 21 seconds, measured from resting position until the last piece was placed in the sorted area. All of the other strategies performed without error and objects could be handled in sequence from the front to the back of the sorting cell or vice versa. When an object was behind another one it could be handled without touching the one in front. It took all of the strategies about the same time to handle three objects placed in the sorting cell, as can be seen in table 6.10. The time was measured from resting position until all pieces were placed in the sorted area.

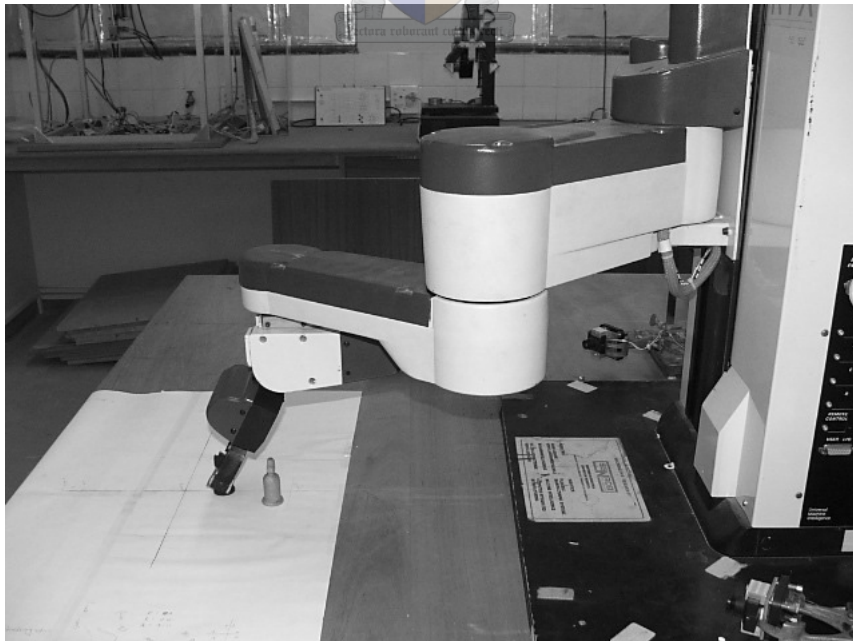


<i>Strategy</i>	<i>Time</i>
1.	2 min 21s
2.	2 min 26s
3.	2 min 27s
4.	2 min 27s

Table 6.10: Sorting time for each of the strategies.



(a) Handling viewed from the front.



(b) Handling viewed from the side.

Figure 6.3: Handling an object behind another.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

This thesis was an investigation into and development of an intelligent, automated sorting cell in the manufacturing line, using object recognition as sorting method.

The automated sorting cell was constructed with the RTX robot, two CCD cameras, an analog to digital converter in the form of a Blackfin evaluation kit and a personal computer. The cameras supplied images of the sorting area in the cell to the image processing algorithms responsible for recognising objects, identifying their colour and calculating their position within the cell. A shape-based object recognition algorithm was written using Fourier descriptors for comparison between the boundaries of test and calibration chess pieces. A separate function was created to determine the colour (light or dark brown) of a recognised object based on the average greyscale colour value in an area around the centre of the object. The position of an object was calculated using stereoscopy and simply surveying techniques. When objects in the sorting cell have been recognised, their colours identified and their positions calculated, they are sorted according to a strategy chosen by the user.

The separate components contributing to a working automated sorting cell were evaluated as well as the integrated system. It is concluded that:

- Object recognition can be used as sorting method within a sorting cell. Fourier descriptors allows for reliable recognition of objects, especially those symmetrical around their longest axis as their profile does not change when viewed from a different angles. Reliability for asymmetrical objects are achieved by extending the indexed calibration boundary set to include the boundaries of objects' profiles from other angles.
- Sufficient accuracy for the position calculation of an object is possible using only stereoscopy. Inaccuracies are caused by an inaccurate environment setup, the varying distance between the object and the camera and camera lens distortion. Stereoscopy limits the sorting ability of the cell to unique objects. Non-unique items cause confusion when pairing extracted angles from the two camera images for position calculation.
- The robot can be precisely moved and movements can be repeated accurately, allowing for reliable handling of objects. Slight precision errors occur due to a slight inaccurate calculated position sent to the robot and misalignment of the robot within the sorting environment.
- Objects are handled accurately using any of the available strategies. The strategies allow the user to choose in which sequence objects are handled.

In conclusion, with object recognition by Fourier descriptors a reliable and relatively accurate automated sorting cell was created. Items unique by shape and colour placed in the sorting area are sorted to the sorted area in a chosen sequence. The sorting cell will form an integral part of the simulated factory environment within SENROB when completed.

7.2 Recommendations

The results reveal that a reliable and accurate automated sorting cell has been developed. However, there are a number of additions or improvements that can be investigated in the future with the aim to increase accuracy, reliability and overall performance of the sorting cell.

7.2.1 Object Recognition Improvements

Although it has been proved that object recognition through the use of Fourier descriptors can be used as sorting method, it is not optimal in its current form. As explained in chapter 5, if the silhouettes of objects overlap the resulting boundary will be unknown to the algorithm and none of the objects will be recognised, even if they are known to the system. Complementing the current recognition algorithm with a method capable of recognising only parts of the boundary will increase the usability of object recognition in the sorting cell.

Various techniques exist that are capable of matching partial boundaries to indexed boundaries, including the landmark-based method and curve moments method. Being able to recognise part of the boundary, the indexed boundary can be used to guess the position of the object enabling the cell to handle it.

A more complex solution using only Fourier descriptors also exists for the problem. If the cameras are moved around the sorting area in a known way, view angles will be found for each object where it is the only object in the specific line of site. Or more simply, the sorting area can consist of a turntable. Objects placed in the area will be recognised and handled. When all of the recognised objects are handled, the turntable will turn, changing the line of site at any unrecognised objects from the first attempt. They will then be recognised and handled. Any unrecognised left on the board will most likely be unknown to the recognition algorithm.

7.2.2 Position Calculation and Gripper Positioning

It was concluded that the current position calculation and gripper positioning for pick-up are sufficiently accurate for the purposes of the project. However, great care

was taken in the setup of the environment and calibration of the algorithms and robot to ensure this. With continuous use of the sorting cell, vibrations caused by the movement of the robot and human interaction will cause the system to become less and less calibrated. A method of calibrating the robot and algorithms for precise position calculation and gripper positioning without the need for a precise positioned environment is proposed.

More accurate handling of objects can be achieved by using closed loop control of the gripper with the cameras as primary sensors. The current method can be used to move the gripper within the area of an object from where closed loop control can take over, guiding the gripper to the exact position relative to the object.

If the position of non-unique items in the sorting cell can be calculated the usability of the cell will be extended. An alternative method for position calculation in the form of 3D environment reconstruction can be investigated. 3D reconstruction will allow the position of non-unique objects to be calculated and even might lead to improved methods for recognising the objects. The method will, however, be far more computational intensive than the current one, but the robustness of the sorting cell will be improved greatly.

7.2.3 Automated Sorting Cell Hardware

It is recommended that two digital cameras, connected directly to the personal computer, be used. This eliminates the need for an analog to digital converter and allows for the full control of the system from one computer. Another option is to embed all the control algorithms on the evaluation board removing the need for the computer. For integration into the simulated factory environment within SENROB, full control from a personal computer is the best option.

7.2.4 RTX control

Control of the RTX through the JAVA interface was successful, but it is recommended that the interface is extended to include speed control over the robot and allow for the different drive modes. This will allow more precise control, needed when complex manoeuvres is necessary.

List of References

- [1] *RTX manuals*. Universal Machine Intelligence, 1st edn, 1985.
- [2] Fernández, A.: C++ library for umi rtx robot. Final Project, March 2004.
- [3] Gonzalez, R. and Woods, R.: *Digital image processing*. 2nd edn. Prentice Hall, 2002.
- [4] Wane, S.: Kinematic solutions of the rtx. Internet, August 2005.
- [5] Joubert, J.: Digital image recognition for machine intelligence. Final Year Project, November 2000. Stellenbosch.
- [6] Joubert, W.: Digital image recognition for rotot vision. Final Year Project, June 2001.
- [7] Petkovic, T. and Krapac, J.: Shape description with fourier descriptors. Tech. Rep., FER Zagreb and FSB Zagreb, 2002.
- [8] Greeff, A.: *The RTX robot programming manual*. University of Stellenbosch, February 1987.
- [9] Alberts, A.: The internet control of a robotic work station. Final Year Project, November 2000.
- [10] Wunderlich, T.: *ComRob-RTX Manual*. University of Stellenbosch, March 2006.
- [11] Pierpaloi, F. and Urbani, A.: Manufacturing service providing in mass customization. In: Teti, R. (ed.), *Intelligent computation in manufacturing engineering 3*, pp. 31–36. CIRP, July 2002.
- [12] Farooq, J. and Osadebey, M.: Content based image retrieval and shape as feature of image. Online Presentation.

- [13] Janse van Rensburg, F.J., Treurnicht, J. and Fourie, C.J.: The use of fourier descriptors for object recognition in robotic assembly. In: Teti, R. (ed.), *Intelligent Computation in Manufacturing Engineering 5*, pp. 509–512. CIRP, July 2006.



Appendices

