CAPACITANCE SENSING

FOR ROBOTIC ARM COLLISION AVOIDANCE

CAPACITANCE SENSING

FOR ROBOTIC ARM COLLISION AVOIDANCE

By

YUE XIAN MA, M. A. Sc

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2007) McMaster University

(Mechanical Engineering) Hamilton, Ontario

TITLE: Capacitance Sensing for Robotic Arm Collision Avoidance

AUTHOR: Yue Xian Ma, M. A. Sc. (Tsinghua University, China)

SUPERVISOR: Dr. Gary M. Bone, Professor

NUMBER OF PAGES: cxxii, 122

ABSTRACT

Existing robotic arms have limited or no ability to avoid collisions with their environment due mainly to the lack of a suitable sensing system. A collision avoidance capability should be incorporated into every robot so that injuries to people and damage to equipment from collisions are prevented. Important applications that could benefit from robot collision avoidance include: manufacturing, robot-assisted surgery, robotic handling of hazardous waste, and personal robots.

Creating a full-coverage, fast, reliable and cost effective sensing system for sensor-based robotic arm collision avoidance is a challenging problem. Capacitive sensors were selected based on their promising potential. Capacitive sensors have the limitations of nonlinearity and being influenced by the environment. In this thesis, their sensing behaviour, and solutions to these limitations, were investigated.

A forward model predicts the capacitance for a given electrode geometry. The conventional method, Method of Moments (MoM) and Finite Element Method (FEM) were investigated and compared. The MoM demonstrated that the fringing electric field ignored by the conventional forward model is significant for the robotic arm application due to the relatively large ratio of electrode gap to electrode area. Two forward modeling cases were simulated by writing macro code for a commercial FEM package. The first consisted of two parallel cylindrical robotic arms. The second consisted of two cylindrical shell electrodes wrapped around a pair of robot links that rotated relative to each other. The results for this case were compared with experimental results. The FEM results were a poor predictor of the experimental results. The failure of the FEM model to

include the true environmental conditions (e.g. air humidity and surrounding electric fields) is the most likely cause of its inaccuracy.

An inverse capacitance model outputs the electrode geometry for a given capacitance. In this research the desired geometric output was the seven robot link pose variables, (x, y, z, q_x, q_y, q_z, q_0), describing the position and the orientation of the link of a robotic arm. A Cerebellar Model Articulation Controller (CMAC) neural network was chosen for the inverse modeling based its ability to model nonlinear behaviour and its efficiency. One CMAC network was trained for each pose variable. The sensor was built using capacitance sensing circuit and a multiplexor board with the potential for 16 by 16 electrode combinations. Note that an n by n combination produces n^2 separate capacitance values.

For the inverse modeling experiments, four aluminum foil electrodes were mounted on a CRS-F3 robotic arm and four aluminum foil electrodes were placed on a wooden box used to simulate a second stationary robotic arm. A pair of reference electrodes was mounted on the back of the CRS-F3 arm. This reference measurement was used to normalize the measured capacitances in order to minimize environmental effects. The normalized capacitance data were used to train and test the CMAC neural networks. The CMAC learning factors were dynamically changed to reduce the training errors. A new fuzzy logic approach was developed that allowed the range of the CMAC input data to be increased without significantly increasing the training error. After evaluating eleven combinations of electrodes, it was determined that only the 3 by 3 and 4 by 4 combinations converged with small training errors. Three methods were used to analyze

the CMAC testing errors: comparison plots, error plots and error metrics. Over a 15 *cm* range, pose variable *y* had maximum absolute errors of 2.1 *mm* for the 4 by 4 electrode combination and 7.2 mm for the 3 by 3 electrode combination. For the 4 by 4 combination the maximum relative errors were less than 3% for the *x*, *y*, and *z* variables, and less than 15% for the quaternion variables. For the 3 by 3 combination, these values increased to 13% and 20%, respectively. The larger relative errors for the quaternion variables were due to their smaller ranges of variation.

Using the same hardware, a simple collision avoidance system was implemented using one pair of electrodes to detect the potential collision between a robotic arm moving in the vertical plane and a second stationary robot. The robot was shown to successfully avoid the potential collision and then continue its motion.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Gary M. Bone, for his excellent supervision and continual support. I am grateful for what I learnt from him and his willingness to share his valuable experience and knowledge all the time. I truly appreciate helpful suggestions from Mr. William Zeng. I would also like to thank technicians, Mr. Joe Verhaeghe, Mr. Jim McLaren, Mr. Ron Lodewyks and Mr. Mark Mackenzie for their help with the test setup. I also thank my colleagues and friends for their advice and encouragement. Finally, I would like to specially show appreciation to my family members for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CMAC	Cerebellar Model Articulation Controller
D-H	Denavit - Hartenburg
EMI	Electromagnetic Interference
FEM	Finite Element Methods
FNN	Fuzzy Neural Network
LMS	Least Mean Squares
MAE	Mean Absolute Errors
MAXAE	Maximum Absolute Errors
MINAE	Minimum Absolute Errors
MoM	Method of Moments
MSE	Mean Squared Errors
NN, RL	neural network, reinforcement learning
PI	proportional-plus-integral controller
RAE	Relative Absolute Error
RME	Relative Mean Error
T & R	Transmitter and receiver

NOMENCLATURE

a, d, A	length, distance, area of the electrode
acv	association cell vector
C, C_{MoM}	capacitance, capacitance calculated by MoM
E, D, ∇	electric field intensity, flux density, gradient operator
$\vec{F}_e, q_e$	electrical force, electrical charge
G, M	the generalization parameter, physical weight memory size in CMAC
h, Ad_i, Δ	pseudorandom hashing function, quantization parameter, physical address for CMAC
Lq	quaternion operator
$\vec{n}$	normal of the surface
N	sensory data input dimension
N_{MoM}	electrode subsection dimension
$p, q, q_x q_y, q_z, q_0$	quaternions
P_x, P_y, P_z, x, y, z	coordinate of electrode center
$\vec{s}, s_1, \ldots, s_N$	CMAC input vector, and elements
T, R	homogenous transfer matrix, rotation matrix
t, b, u	top, bottom, subsection length
V, Q	conductor voltage, charge

W, L_4, M, S	world frame, link 4 frame, moving electrode frame, stationary electrode frame
y_d, y, e	desired output, network output, errors
$\beta_1, \beta_2,$	CMAC learning gains
ε	permittivity in media
θ, d, a, α	D-H parameters
ρ_v, ρ_s	the volume, area charge density

CHAPTER 1

INTRODUCTION

1.1 Preface

Conventional robots have limited or no ability to avoid collisions with their environment. This lack of ability reflects the difficulty of the collision avoidance problem rather than its unimportance. Indeed, a collision avoidance capability should be incorporated into every robot so that injuries to people and damage to equipment are prevented. This applies whether or not the robot is autonomous, pre-programmed (as are most industrial robots) or teleoperated (since humans make mistakes). Important applications that could benefit from robot collision avoidance include: robot-assisted surgery, robotic handling of hazardous waste, personal robots and manufacturing robots.

The largest use of robotic arms is in automotive manufacturing. These robots move quickly, and in close proximity to each other, so collisions are likely to occur. The likelihood of collisions is increased by the large tools (and their associated hoses and cables) carried by these arms.

The current collision avoidance approach is based on CAD models of the arms and their environment and data from the built-in joint angle sensors of the arms. If a collision occurs, it will result in expensive downtime from production and potentially damage to equipment. The CAD-based method assumes complete information about the robot and its environment is available and accurate. First the geometries of the robot links and the objects in the surrounding area have to be known. Then at each time step, the positions of

the links are updated by reading the joint angle sensors. When the closest distance between the arm and its environment reaches a critical value, the speed and/or path of the arm is altered to avoid a collision. The advantage of this method is that the only sensors required are built into all robotic arms. Its disadvantages are it is difficult to adapt to a changing environment and it is computationally expensive (especially for environments with complex geometries). Its lack of adaptability is a critical weakness. Hoses and other flexible objects are very difficult to model in CAD since they change shape often and unpredictably. When the robot or its environment is changed, as can happen in routine maintenance, the CAD model will no longer be accurate, and this can lead to a collision.

The alternative to CAD-based is sensor-based collision avoidance. Sensors are used to determine the distances between the robot and obstacles in its environment. Since no 3D models are involved, this approach is computationally efficient. It also has the advantage of being able to work with flexible objects such as hoses. Its disadvantage is that it requires a reliable proximity sensing system that can cover both the robotic arm and objects in its path. The sensing system provides information on the proximity of the various parts of the robot (i.e. links, tools, hoses, etc) to obstacles in its environment. This proximity data can then be used to modify the commanded trajectory to produce a collision free one leading to the desired destination using a suitable path-planning algorithm. In general, the task of maintaining the desired trajectory would be the primary task, with the avoidance of obstacles being a secondary task [1].

Creating a suitable sensing system is a challenging problem. The sensing system must cover the surface of all parts of the robot that may be involved in collisions as its

primarily requirement. Next, it must provide reliable proximity data while operating in a harsh industrial environment. Thirdly, it must be cost effective. Finally, it must be compatible with the original design and function of the robot.

1.2 Objectives and Organization of the Thesis

As previously mentioned, robotic arms often work closely to each other and other equipment in industry. The scope of this research is limited to the case of a single moving robotic arm and a stationary robotic arm or other single obstacle. The research objectives are to demonstrate the suitability of capacitance sensing for this collision avoidance application, and to implement a simple capacitive sensor-based collision avoidance method. After the literature review in chapter 2, chapter 3 presents forward models of capacitive sensors using analytical and numerical approaches. A neural network based approach to inverse modeling of capacitive sensors is described in chapter 4. The experimental verification of this approach is presented in chapter 5. In chapter 6, a simple capacitive sensor based collision avoidance method is demonstrated. Conclusions and recommendations for future work are given in chapter 7.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A typical sensor-based collision avoidance system includes three components: the sensing hardware, the algorithm to convert the sensor data into a useful form (usually proximity information), and the path planning or control algorithm that employs the processed data. The data processing and control algorithms are often merged into a single algorithm. The sensors can be categorized into those based on reflected signals and those based on electromagnetic radiation. The relevant literature will be reviewed in this chapter.

2.2 Collision Avoidance Sensors Based on Reflected Signal

In this category, the signal sent by the signal source is reflected by the object. The receiver then gets the reflected signal and calculates the distance. In most cases, the signal sources are acoustic, infrared, laser or microwave. Depending on how the distance is calculated, this category includes four different types of systems: intensity of reflection sensor, time-of-flight sensor, triangulation sensor, and vision system.

2.2.1 Intensity of Reflection Sensors

Cheung and Lumelsky studied intensity of reflection sensors for robot collision avoidance [2]. Because infrared light can be easily scattered in all directions and the distance is proportional to the intensity of the light, they chose infrared sensors. The

sensors are modulated at 10 *KHz* frequency to minimize interference with ambient lighting. The sensor consists of printed circuit board modules each containing 16 pairs of transmitters and receivers. These modules are placed onto the surfaces of the robot links. The approximate sensing range is 125 *mm*. They conducted some collision avoidance experiments with their system using lightly colored obstacles [3]. The drawback of infrared sensors is that the intensity of the reflected signal is different for objects with different colors. Dark colored objects may not be sensed at all. Objects having optical mirror like surface are also hard to be detected.

Seraji et al. built a sensor-based collision avoidance system using arm-mounted infrared proximity sensors [4]. The infrared sensors are mounted on the robotic arm by employing 12 trapezoidal and 8 octagonal circuit boards. Distributed signal processing is used to solve computationally intensive problem caused by total of 63 infrared emitters and 118 infrared detectors on the robotic arm. A group of sensors, detectors, and a microprocessor forms a "Sensor Cell". The Sensor Cells communicate with the host computer to report the distances. To change the robot trajectory for collision avoidance, a proportional-plus-integral controller is used in the outer loop of the robot controller when the distances are less than user-preset values. The calibration and collision avoidance experiments are performed using only white objects.

Similarly, Gandhi et al. implemented a simple collision avoidance system using a reinforcement learning neural network that outputs the joint motion command depending on the measurement from two infrared range sensors. Their sensors can detect distances ranging from four to thirty centimeters [5]. They use a Q-learning algorithm to learn

appropriate collision avoidance actions. Q-learning employs a trial-and-error approach and in their experiment it requires a few minutes to work with only two sensors. When enough sensors to cover the entire robot are used, it is suspected that their approach would be overly slow and could suffer from interference problems.

2.2.2 Time-of-Flight Sensors

Ultrasonic sensors are a common form of time-of-flight proximity sensor. Cheung and Lumelsky mentioned that ultrasonic sensing could have the problem of specular (mirror like) reflection for some obstacles because the wavelength of the ultrasound is relatively long [3]. In this situation, because the reflected signals are only in one direction the receiver cannot receive the reflected signal most of the time. Besides, the popular Polaroid ultrasonic sensors operate poorly when obstacles are closer than 25 *cm* from the sensor. This minimum distance is too large for robots operating close to each other.

Blanc et al. have developed a 3D camera in which the image sensor receives the reflected sine amplitude-modulated laser signal sent by the camera [6]. The phase shift between the sent and received sine waves is used to estimate the distance. The authors claim a best case accuracy of 5 *mm* but they do not present any distance measurement results. They do include experimental results from simple collision avoidance experiment. performed with a mobile robot. A similar paper can be found from the same group using infrared LEDs as the light source [7].

2.2.3 Triangulation Sensors

Triangulation sensors obtain the distance by using the length of one side and two interior angles of a triangle. Marques et al. developed a triangulation system based on three laser beams and a video camera [8]. The principle is shown in Fig. 2.1. B is the distance between the central point of the lens and the laser beam; α is the angle between the camera optical axis and the laser beam and γ is the angle calculated from the position of the image point and the focal distance. By using B, α and γ, the distance to a single point may be computed.

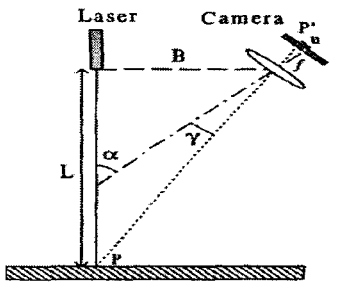

Fig. 2.1 Triangulation system [8]

The three laser beams are used to measure the proximity of three points, allowing both the distance and relative orientation of the object to be measured. They have not used this system for robot collision avoidance.

In fact, triangulation sensors are not well suited for the collision avoidance application. They only measure at a point so they must be scanned to cover the surface of the object, significantly slowing the measurement time. Their accuracy depends on the accuracy of the baseline but the baseline is usually tens of times smaller than the distance they are going to measure. Another problem is the possibility of occlusion on specular

surfaces due to multiple reflections. Furthermore, they suffer from lens distortions, image sensor distortions, and sensor noise [8].

2.2.4 Vision Systems

Ebert et al. presented a specially designed high-speed vision-chip for avoiding collisions between a robotic arm and a person's arm [9]. It can measure at a rate of 500 *Hz*. It distinguishes the robot from the human using a gray scale image sensor and requires the human to wear white gloves. Since it only measures in 2D, it will misinterpret the cases when the person's arm is in front of, or behind, the robotic arm.

Morikawa et al. implemented a visual servo control algorithm for real-time collision avoidance using several cameras mounted on the surfaces of the links of a robotic arm [10]. The optical axis of each camera is parallel to the centerline of its associated link. With this setup the proximity of an obstacle to the robot link can be estimated assuming that the obstacle is distinct from the background and exists somewhere along the link in a narrow range. The distance between obstacle and the camera must be at least 30 *cm*. Otherwise, the robot "motion become radical and dangerous."[10]. They also assume that the lighting can be carefully controlled. These assumptions are not realistic for most applications.

Vision systems can be based with triangulation as well, e.g. stereovision. Tsalatsanis et al. used a stereovision system mounted on a mobile robot to identify the target and to calculate the distance relative to the robot [11]. With their system, the target is identified and the identical point of interest is found in both camera views. Using the location of this point, and the pan and tile angles of both cameras, the distance is calculated. The

color of the object to be detected has to be not similar with the others in the scene. Besides the problems previously mentioned for triangulation sensors, the system has the problem that the point of interest may not be visible by both cameras.

2.2.5 Limitations of Sensors Based on Reflected Signal

For the collision avoidance project, we require that the sensors can cover the entire robot surface. However, this category of sensors measures only at a point or a small area. As a result, large numbers of sensors are needed to protect the robot link surfaces. Using many sensors creates problems with quantity of data, measurement speed, wiring, and interference. This will result in a slow system, and problems with reliability due to the complexity of the system.

Furthermore, even though the system uses hundreds of sensors, in most cases the system still cannot completely cover the whole robotic arm. There are blind spots or occlusions. This is illustrated in Fig. 2.2.

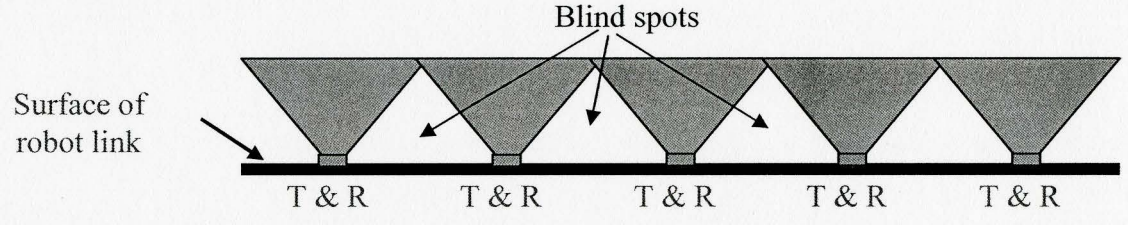

Fig. 2.2 Surface of a robot link covered by reflected signal sensors where T ≡ transmitter and R ≡ receiver

2.3 Collision Avoidance Sensors Based on Electromagnetic Effect

For the second category of sensors, sensors based on electromagnetic field effect, there are inductive sensors and capacitive sensors.

2.3.1 Inductive Sensors

Inductive sensors are widely used for metal sensing in the areas of factory automation and control. They have the advantage of immunity against oil, water and dirt [12][13]. The disadvantages of inductive sensors are: short range and useful for detecting metals only. The detecting range is usually below 10 *cm* [14]. These limitations make them impractical for the robot collision avoidance application.

2.3.2 Capacitive Sensors

There are many advantages of using capacitance-based proximity sensors for the robot collision avoidance, as follows [2][15][16][17][18]:

1. The distribution of the electric field allows whole coverage for the robot surface without using a large number of sensors.
2. The changes of the electric field can be detected almost immediately, unlike the other methods needing to listen to returned echoes.
3. The capacitive sensor is reliable, inexpensive and quite simple, since it uses only two conductive electrodes per sensor. This property greatly increases the reliability while the algorithm is less computationally intensive.
4. Capacitance measurements are insensitive to the color and texture of the approaching obstacle.
5. Capacitive sensors are insensitive to the dirt and oil that commonly occurs in industrial environments.

2.3.3 Limitations of the Capacitive Sensors

Many prior researchers agree that capacitive sensors are a very good solution for robot collision avoidance [2][15][16][17][18]. At the same time, they point out their limitations of nonlinearity, and short measurement range.

This type of sensor is highly nonlinear because of the nature of the electromagnetic field. We know that capacitance depends on the dielectric media, electrode sizes, orientation of the electrodes and distance between the two electrodes. This means even with the same electrode size, different orientations of each electrode with corresponding distances can have the same capacitance. Therefore, one capacitance reading can correspond with infinitely many combinations of distance and orientation.

Environmentally, the capacitance will be affected by the air quality (since air is the dielectric in this application), the electric field generated by motors or other sources, and the material and shape of surrounding objects. Changes to the air humidity and temperature will significantly change the capacitance. The electrodes will cover the robot links, which have joints or motors that draw different currents for different loads. The different currents will generate different electrical fields that will have influence on the capacitance. Even surrounding equipment may radiate electrical noise to the electrodes.

2.4 Modeling of Capacitance

When electrodes are mounted on the robotic arms, the distance between the electrodes and the robotic arm is much smaller than the distance between two electrodes. And the surface of the robotic arm is typically grounded. As a result, there will be a very dense electric field between electrodes and robotic arm. Relative to this field, the field we

are using to measure the distance between two electrodes is very weak. Vranish et al. at NASA developed a capacitive proximity sensor called the "capaciflector" to solve this problem [15]. Their intended application is to prevent a robot from colliding with other objects in the space, particularly human beings. The surface of the sensor is one plate of the capacitor and that of the object is another. Note that the object acts as an electrical ground and is not physically wired to anything. The oscillating frequency changes with the capacitance between robotic arm and the object. The sensor has two elements, the sensing element and the reflector element (or driven shield) on the back of the sensor driven by the same voltage as sensor. This design prevents the electric field from directly traveling into the grounded robotic arm and thus effectively increases the detection distance of the sensor. In addition, a voltage follower is used to reduce the influence of the reflector. They did not test their system with a real robotic arm and have not published anything since 1991.

Novak and Feddema realized a system using a capacitive sensor, and an obstacle avoidance control algorithm [16][17]. The sensor uses two electrodes on a single circular substrate to generate and measure the electric field. One electrode acts as an emitter driven by an oscillator and the other acts as a receiver connected to an amplifier. The substrate is a printed circuit board with three layers. The top layer contains the charge amplifier components; the middle layer is grounded for isolation; and the bottom layer is for the power and signals. A synchronous detection circuit, and phase and frequency locking techniques enabled low noise detection. They mounted 49 of these sensors on a PUMA 560 robotic arm. The position of the sensors was chosen to provide enough

overlap of the sensing fields. They did not accurately correlate the capacitance readings with distance. They presented the results from one collision avoidance experiment where the robot decelerated to a stop to avoid hitting a 60 *mm* diameter metal pipe.

Artificial neural network is one approach for solving highly nonlinear problems. Hoole tried to solve an inverse electromagnetic field problem using an artificial neural network [19]. His multilayer perceptron neural network has four inputs, three outputs and one hidden layer with 15 nodes. The back-propagation algorithm is used for training the network. He concluded that this neural network is feasible only for "a narrow range of performance of a particular class of device."

Marashdeh et al. used a feed forward neural network to solve nonlinear forward problems in electrical capacitance tomography, and then predicted different permittivity distributions represented by different capacitance values [20]. Their neural network is composed of two hidden layers with 30 - 30 neurons. Two figures showing their results are much better than those obtained by traditional linear methods were included.

2.5 Summary

Two categories, sensors based on reflected signals and those based on electromagnetic effect, have been studied and applied to the robot collision avoidance. The former category includes intensity of reflection sensors, triangulation sensors, time-of-flight sensors and vision systems. The later one consists of inductive and capacitive sensors. Both of the categories have their own advantages and disadvantages.

For intensity of reflection sensors, all the researches chose infrared sensors because the infrared light can be easily scattered in all directions and the distance is proportional

to the intensity of the light. However, these sensors respond differently to objects with different colors. Ultrasonic, laser and the infrared sensors are used as time-of-flight sensors. The accuracy of these sensors is affected by the air and surrounding objects, besides ultrasonic sensors cannot detect small distances. Triangulation sensors can only measure one point each time, and suffer from lens and image sensor distortions. Stereovision systems are based on triangulation as well. Besides the problems with triangulation, the stereovision systems need to find a common point of interest in both camera views. The high-speed vision chip using gray scale image sensor required the human to wear white gloves. The vision system using several cameras mounted on the surface of robotic arm required that the obstacle was distinct from the background and not too close to the camera. Because a large numbers of the sensors have to be mounted on the robotic arm, the category of sensors based on reflected signals will have the problems of computational intensity, interference and blind spots.

Inductive sensors can only measure metal objects with closer than 10 *cm*. Capacitive sensors possess promising properties for the robot collision avoidance. Besides their simple, compact and robust properties for this application, they are easily applied to objects of any shape and can protect any kind of materials including conductors or dielectrics. However, they are nonlinear and affected by environment. A few researchers built different prototype to solve this problem. One prototype tried to enhance the electrical field by putting a "capaciflector' between the electrodes and robotic arm. But there was no experiment for the real robot. Another prototype featured a compact sensing circuit design, but they did not model the relationship between capacitance readings and

distance. Two neural network methods were proposed to model the relationship between capacitance and distance. One could not solve complex problems and the other reported better results than the traditional linear methods for tomography. They both agreed that the neural network is a promising way to overcome the limitations of capacitive sensors.

CHAPTER 3

FORWARD MODELING OF CAPACITIVE SENSORS

3.1 Introduction

According to the definition, capacitance exists between two conductors with voltage potential. A "forward model" is used to calculate capacitance from a given electrode geometry and dielectric. Such a forward model could be used to design electrodes for a particular sensitivity for example. The conventional approximate forward model omitting the fringing effect gives a very simple formula for the capacitance calculation. However, in this thesis, the fringing effect dominates the experiments because the distances are large relative to the area of the electrodes and the conductors are usually not parallel. The electrical charge distribution including the fringing effect will be analyzed using the Method of Moments (MoM) [21][22]. Finally, a forward model will be obtained using the Finite Element Method (FEM) and compared with experimental results.

3.2 Analytical Model

A capacitive proximity sensor can be modeled as quasi-static case because the robot velocity is small compared to the excitation frequency. In the quasi-static case, the electric and magnetic fields are not interconnected [22][23]. Maxwell's equations in the electrostatics form are,

$$\nabla \cdot D = \rho_v \tag{3.1}$$

$$\nabla \times E = 0 \tag{3.2}$$

where, D (C/m^2) is the electric flux density, $\vec{E}$ (V/m) is the electric field intensity, $\rho_v (C/m^3)$ is the volume charge density, and ∇ is gradient operator. Its Cartesian form is,

$$\nabla = \frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j} + \frac{\partial}{\partial z}\vec{k} \tag{3. 3}$$

A capacitor is formed when any two electric conductors are separated by a dielectric or insulating media. Fig. 3.1 shows the capacitance definition, one conductor with charge $+Q$, another conductor with charge $-Q$, and the potential (voltage) between them being V. Capacitance of the two conductors is defined as,

$$C = \frac{Q}{V} \qquad \textit{(Coulomb/Volt) or Farad} \tag{3. 4}$$

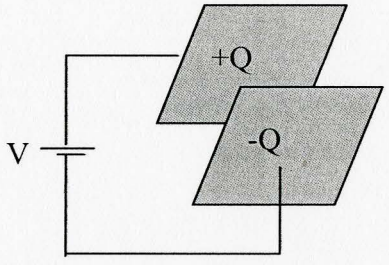

Fig. 3.1 A simple capacitor

3.2.1 Capacitance Calculation

To calculate the capacitance, the analytical form of charge Q and voltage V have to be deduced. The total charge Q for the volume in Eq. (3.4) can be calculated by using Maxwell's equations expressed in Eq. (3.1).

$$Q = \int_v \rho_v dv = \int_v \nabla \cdot \vec{D} dv \tag{3. 5}$$

Eq. (3.5) can further be deduced using the divergence theorem for the volume v enclosed by surface s. The surface s is also called a Gaussian surface.

$$\int_v \nabla \cdot \vec{D} dv = \oint_s \vec{D} \cdot d\vec{s} \tag{3.6}$$

Thus, from Eq. (3.5) and Eq.(3.6), we can have the integral form of Gauss's law,

$$\oint_s \vec{D} \cdot d\vec{s} = Q \tag{3.7}$$

From the definition of the electric field intensity $\vec{E}$, we also have,

$$\vec{D} = \varepsilon \vec{E}_n \tag{3.8}$$

Finally, we have formula of total charge Q calculated by electric field intensity $\vec{E}$.

$$Q = \oint_s \varepsilon \vec{E} \cdot d\vec{s} \tag{3.9}$$

Eq. (3.9) shows that the free charges on the conductor's surface give rise to an electric field $\vec{E}$.

On the other hand, the analytical form of voltage can be obtained from the definition of voltage. Voltage or voltage potential between two points is the amount of work needed to move a unit charge between two points. If $\vec{F}_e$ is the electrical force acting on the charge q_e, the electric field $\vec{E}$ can be defined as,

$$\vec{E} = \frac{\vec{F}_e}{q_e} \tag{3.10}$$

Because of the presence of the electric field $\vec{E}$, the charge q_e is being moved in the opposite electric field direction. The external force needed to counter-act the force $\vec{F}_e$ is,

$$\vec{F}_{ext} = -\vec{F}_e = q_e \vec{E} \tag{3.11}$$

The energy needed to move a vector differential distance $d\vec{l}$ is,

$$dw = \vec{F}_{ext} \cdot d\vec{l} = -q_e \vec{E} \cdot d\vec{l} \tag{3. 12}$$

Therefore, the differential electric potential is

$$dV = \frac{dw}{q_e} = -\vec{E} \cdot d\vec{l} \tag{3. 13}$$

The voltage between any two points p_1 and p_2, can be obtained as,

$$V_{21} = V_2 - V_1 = -\int_{p_1}^{p_2} \vec{E} \cdot d\vec{l} \tag{3. 14}$$

Finally, substituting Eq. (3.9) and Eq. (3.14) into Eq. (3.4), the definition of capacitance gives,

$$C = \frac{\int_s \varepsilon \vec{E} \cdot d\vec{s}}{-\int_l \vec{E} \cdot d\vec{l}} \quad \text{(Farad)} \tag{3. 15}$$

3.2.2 Capacitance of Parallel-Plate Capacitor with Fringing Field Omitted

Fig. 3.2 shows a parallel-plate capacitor. Each surface area is A and the distance between two plates is d. The capacitor is filled with dielectric material with permittivity ε. The upper plate has the electric charge $+Q$ and the lower one has charge $-Q$. The electric field $\vec{E}$ is from upper plate to lower plate.

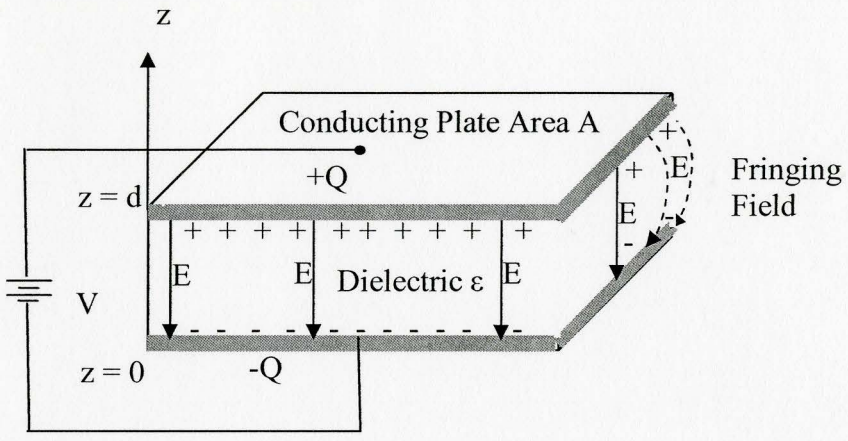

Fig. 3.2 Parallel-plate capacitor

When the plate is much larger than *d*, the fringing field can be omitted and the charge is uniformly distributed. The charge density will be,

$$\rho_s = Q / A \tag{3. 16}$$

Fig. 3.2 also shows that the electric field lines originate on the positive charges and terminate on the negative charges. Because there is no tangential component for the conductor, this field only has a normal component. If we use $\vec{n}$ for the normal of the surface at the point, E is simply,

$$E = \vec{n} \cdot \vec{E} = \frac{\rho_s}{\varepsilon} \tag{3. 17}$$

Therefore,

$$E = \frac{\rho_s}{\varepsilon} = \frac{Q}{\varepsilon A} \quad \text{or, } Q = \varepsilon E A \tag{3. 18}$$

On the other hand, if a coordinate system can be set as in Fig. 3.2 where the *xy* plane is at the lower plate and *z* is perpendicular to the plates. The electric field is then only in the *z* direction, expressed as,

$$\vec{E} = -E\vec{k} \tag{3.19}$$

Hence, $$V = -\int_0^d \vec{E} \cdot d\vec{l} = -\int_0^d (-E\vec{k}) \cdot \vec{k} dl = Ed \tag{3.20}$$

The capacitance is,

$$C = \frac{Q}{C} = \frac{\varepsilon EA}{Ed} = \varepsilon \frac{A}{d} \tag{3.21}$$

Eq. (3.21) shows the capacitance is proportional to the permittivity and the area of the electrodes but inversely proportional to the distance in the simplified situation when the fringing field is omitted.

3.3 Method of Moments Model

The analytical equation (3.15) is hard to solve. Method of Moments (MoM) is a method for reducing Maxwell's equation to a matrix equation and then solving the matrix equation by known techniques [21].

In the collision avoidance application, the electrode area is small compared with the distance, the condition when the fringing field dominates the capacitance. To fully understand the working scheme of a capacitive sensor, the electric charge distribution and the capacitance have to be analyzed. In this section, the air-filled parallel plate capacitor will be analyzed for the large distance case.

Fig. 3.3 shows the specifics of the capacitor. The voltage between the plates is two volts, and the plates are square with the width and length a = 0.5 m, and distance between two plates is d, which will be changing in the study. By using MoM [21][22] the charge distribution and the capacitance will be analyzed.

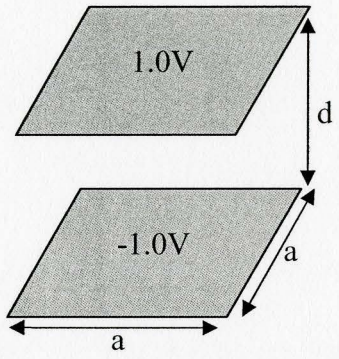

Fig. 3.3 Parallel-plate capacitor specifics

3.3.1 Analysis

Two plates are divided into N_M by N_M square subsections, respectively, giving $2N_M^2$ subsections. Fig. 3.4 shows an example with 10 by 10 square subsections on both plates.

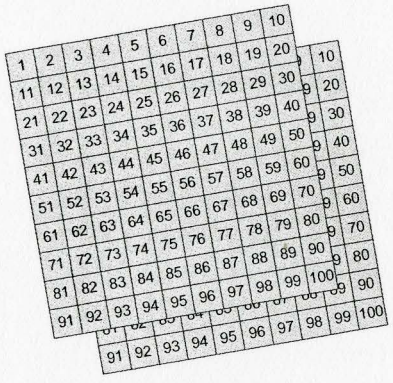

Fig. 3.4 The subsections on the electrodes with $N_M = 10$

Let t denote "top plate" and b denote "bottom plate", the matrix for MoM calculation is

$$[l] = \begin{bmatrix} [l^{tt}] & [l^{tb}] \\ [l^{bt}] & [l^{bb}] \end{bmatrix} \tag{3. 22}$$

where, in the matrix,

$$[l^{tt}] = [l^{bb}]$$

If m is the sequence number of the bottom layer, n is the sequence number of the top plate, d is the distance between two plate, and u is the length of the subsection, which is equal to $\frac{a}{N_M}$, each element in the matrix is expressed as,

$$l_{mn}^{tt} = \int_{\Delta x_n} dx' \int_{\Delta y_n} dy' \frac{1}{4\pi\varepsilon\sqrt{(x_m - x')^2 + (y_m - y')^2}} \tag{3.23}$$

Or,

$$l_{mn}^{tt} \approx \frac{u^2}{\pi\varepsilon\sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}} \tag{3.24}$$

Because the dielectric is air

$$\varepsilon = 8.85\times10^{-12} \qquad coulomb^2 Newton^{-1} m^{-2}$$

For the singularity point when $x_m = x_n$ and $y_m = y_n$, the equation will be,

$$l_{mn}^{tt} \approx \ln(1+\sqrt{2})\frac{2u}{\pi\varepsilon} \tag{3.25}$$

On the other hand,

$$l^{tb} = l^{bt}$$

And,

$$\left[l^{tb}\right] = \left[l^{bt}\right]$$

For $m \neq n$

$$l_{mn}^{tb} = \int_{\Delta x_n} dx' \int_{\Delta y_n} dy' \frac{1}{4\pi\varepsilon\sqrt{(x_m - x')^2 + (y_m - y')^2 + d^2}} \tag{3.26}$$

Or

$$l_{mn}^{tb} \approx \frac{u^2}{\pi\varepsilon\sqrt{(x_m - x')^2 + (y_m - y')^2 + d^2}} \tag{3.27}$$

For $m = n$,

$$l_{mn}^{tb} \approx \frac{0.564u}{\varepsilon}\left(\sqrt{1+\frac{\pi}{4}\left(\frac{d}{u}\right)^2} - \frac{\sqrt{\pi}d}{2u}\right) \tag{3.28}$$

The capacitance of the parallel-plate capacitor is [21][22],

$$C = 2u^2 \sum_{m=1}^{N_M^2}\sum_{n=1}^{N_M^2}(l_{mn}^{tt} - l_{mn}^{tb})^{-1} \tag{3.29}$$

3.3.2 Simulation Results

Fig. 3.5 shows the charge density (C/m^2) along selected subsections of each square electrode. The numbers along the x-axis are the numbers shown in Fig. 3.4. The ten subsections in Fig.3.5 are located on the edge of the electrodes, and the Fig. shows the charge density is large when the subsection is close to the corner of the plate. The maximum value is 2.2×10^{-10} C/m^2, and minimum value is 1.2×10^{-10} C/m^2.

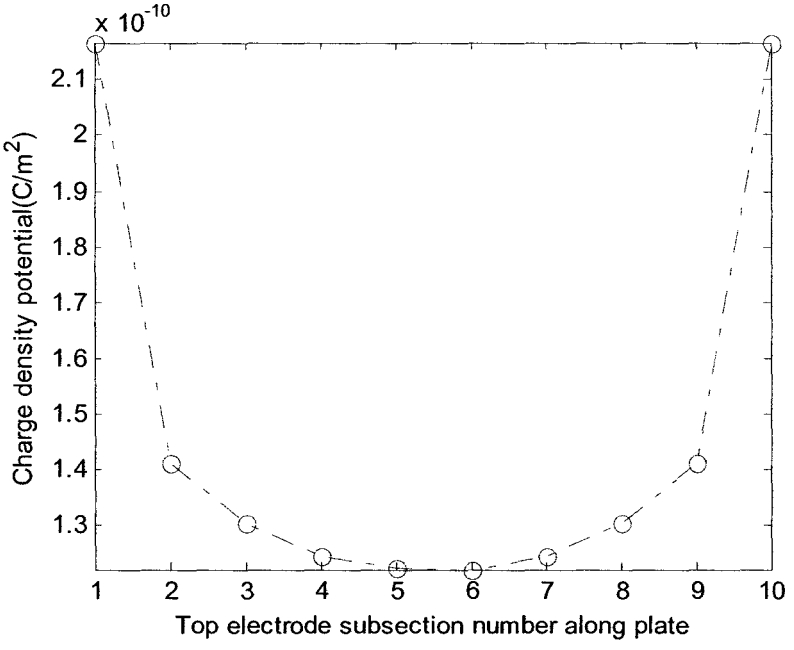

Fig. 3.5 Charge density distribution along edge for $a = 0.5$ m and $d = 5$ m

Fig. 3.6 illustrates the charge density for the entire 10 by 10 subsections of the plate corresponding to Fig. 3.4. The electrode size is 0.5 *m* by 0.5 *m* and the distance between them is 5 *m*. The maximum value in this 3D plot is $2.2\times10^{-10}\ C/m^2$ while the minimum value is $4.4\times10^{-11}\ C/m^2$, a ratio of almost 5. Recall that the approximate model assumes this ratio equals one.

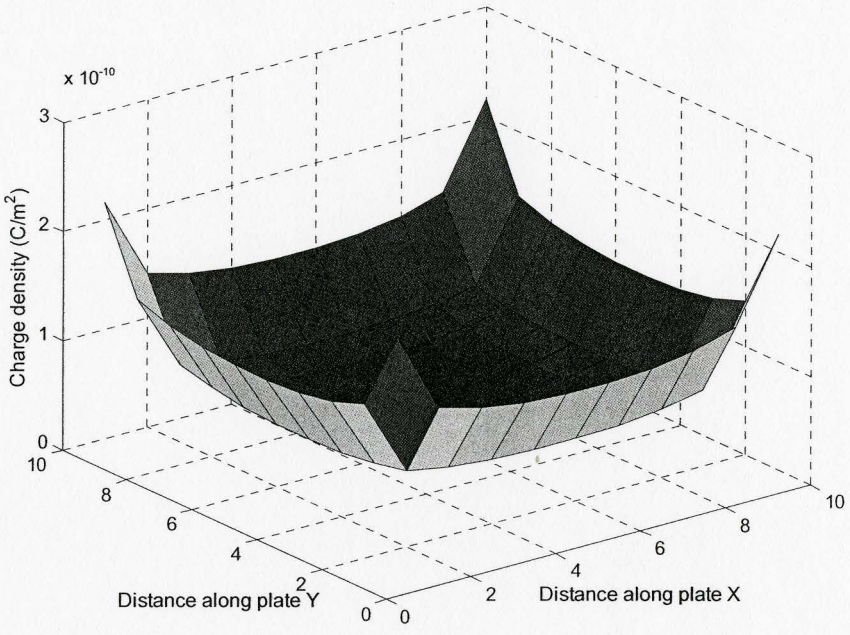

Fig. 3.6 Charge density over entire plate electrode for $a = 0.5\ m$ and $d = 5\ m$

Fig. 3.7 is a plot of the ratio of the capacitances obtained using the MoM to the value from the approximate Eq.(3.21). Fig 3.7 uses the normalized axes: $x = d/2a$ and $y = C_{MoM}\, d/\varepsilon A$. This Fig. clearly reflects the changing ratio of the capacitances when the distance is small compared with the size of the plate ($\leq 0.05a$, where a is the plate length). When d is small, errors due to neglecting fringing are small. However, the errors increase with distance. This means the fringing effect will be larger and larger with the distance

increasing. When the distance between two electrodes is 8 times length of electrodes, the MoM capacitance is 10 times larger than the approximate value.

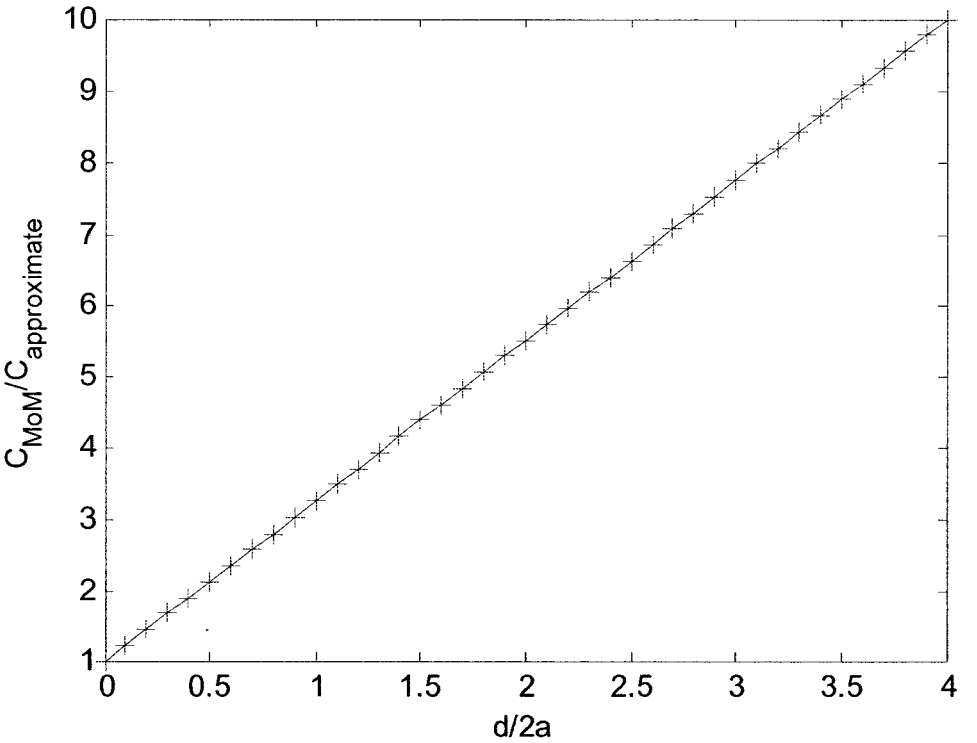

Fig. 3.7 Comparison of the capacitances calculated by MoM and the approximate equation, the errors of approximate formula increase proportionally with the increase of the distance

3.4 Finite Element Method (FEM)

The capacitance depends on the geometry of the electrodes (distance, size, shape and orientation), and the material filled in between two electrodes, which is air in this research. The forward model uses the geometry of the electrodes to calculate the capacitance. The Finite Element Method (FEM) is another technique for solving the complex equations [24].

A commercial FEM package, ANSYS, has been successfully applied in various areas. ANSYS Multiphysics can be used for this application. Three electrode geometries have been studied in this thesis by creating ANSYS macro programs, namely: sphere, cylinder and cylindrical shell. Two of the macro programs are listed in Appendix A.

Sphere and cylinder are two widely used geometric primitives used in collision avoidance research. The capacitance between two spheres and the capacitance for each of the spheres was solved first, after verifying FEM capacitances for two spheres with different distances with analytical calculation results, two parallel cylinders and two cylindrical shell electrodes were simulated using FEM. Finally, the FEM forward model results for the last case were compared with the experimental results. The details will now be presented.

3.4.1 Ground Capacitance and Lumped Capacitance

In the experiment, the electrodes will be mounted on robotic arms. Fig. 3.8 is an equivalent electrical schematic. In the Fig., conductor 1 and conductor 2 represent two robotic arms. And the ground can be expressed as conductor 3. There exist two types of capacitance, the ground capacitance and the lumped capacitance. The ground or self-capacitance is the capacitance between a conductor and ground. The lumped or mutual capacitance values exist between two conductors.

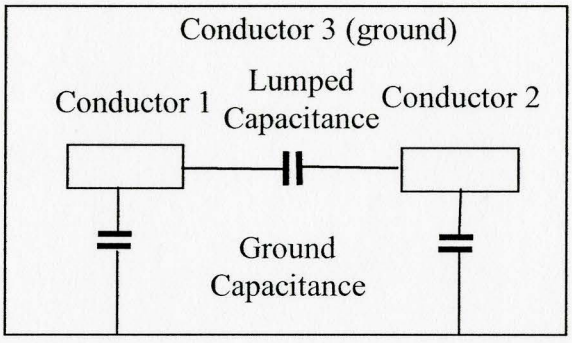

Fig. 3.8 Equivalent circuit for a two robotic arm system

3.4.2 Two Parallel Cylinders

In this section, two parallel cylindrical robotic arms are simulated with various gaps. The radius, 0.1 *m*, and the length, 0.4 *m*, of the arms are fixed. The gap or distance is changed from 10 *mm* to *1 m*. Fig. 3.9 shows the two kinds of distances used to represent the distance between two arms, surface-to-surface distance, d_1, and center-to-center distance, d_2.

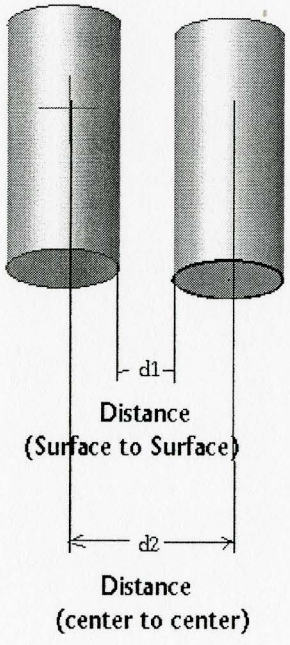

Fig. 3.9 Robotic arms represented as cylinder primitives

Table 3.1 and Fig. 3.10 are the values and the plot of the corresponding capacitances. These results demonstrate that the relationship between Mutual Capacitance and d_2 becomes more nonlinear as the distance decreases. It is also worthwhile noting that decreasing the distance by a factor of 100 only increased the capacitance by a factor of 15.3.

Table 3.1 ANSYS simulation results of capacitance for two parallel cylindrical robotic arms

d_2 (*mm*)	210	240	250	300	400	550	700	1200
d_1 (*mm*)	10	40	50	100	200	350	500	1000
Mutual Cap. (*pF*)	35.7	18.4	16.4	11.3	7.5	5.23	4.04	2.33

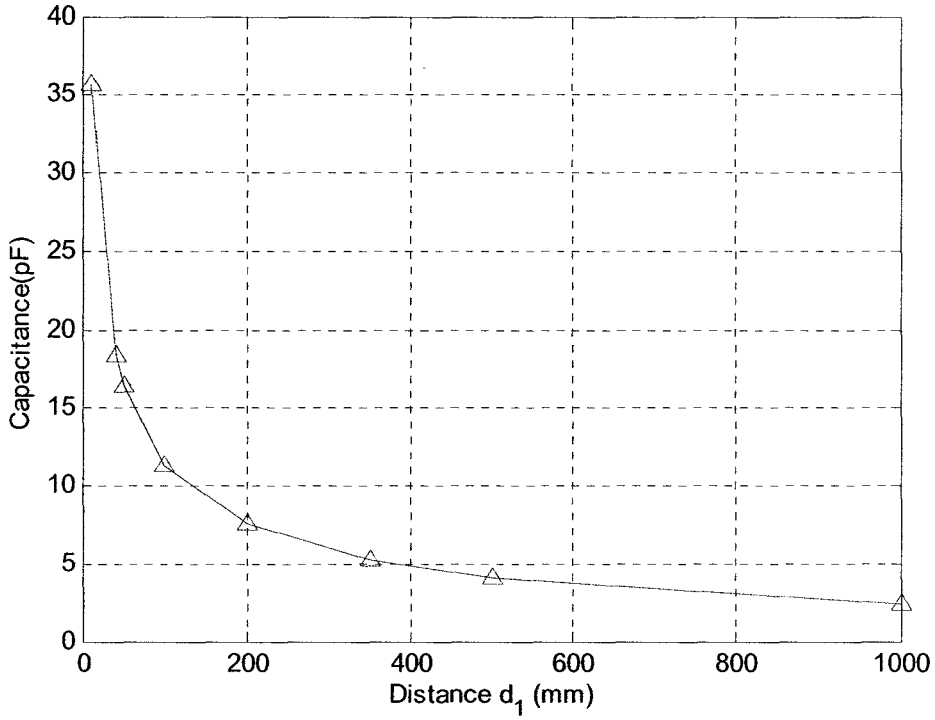

Fig. 3.10 Capacitance of cylindrical robotic arms

3.4.3 Cylindrical Shell Electrode Simulation and Experiment

The case studied in this section is shown in Fig.3.11. Two shell electrodes are wrapped around a pair of robot links that rotate relative to each other, as would occur in the some real applications.

The same geometry was built in the experiment. In the experiments and the ANSYS simulations, the capacitance was simulated/measured over a 0 to 25 degrees range of rotation (ref. angle ϕ in Fig.3.11).

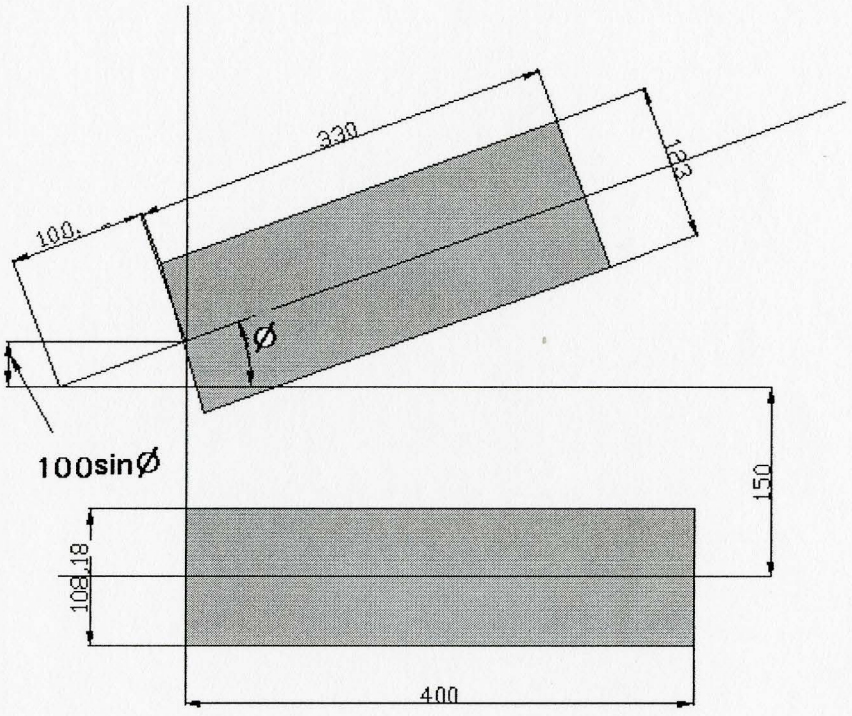

Fig. 3.11 Geometry of the two cylindrical shell electrodes used in the simulation and experiment (unit: *mm*)

The finite element model is shown in more detail in Fig. 3.12. The elements are 10-node-tetrahedral. The small dots in the Fig. are the tetrahedral elements. The program

chose the optimal meshing. The left half of the Fig. shows *xz* plane view of the FEM elements. The right half shows *xy* plane view of the FEM elements.

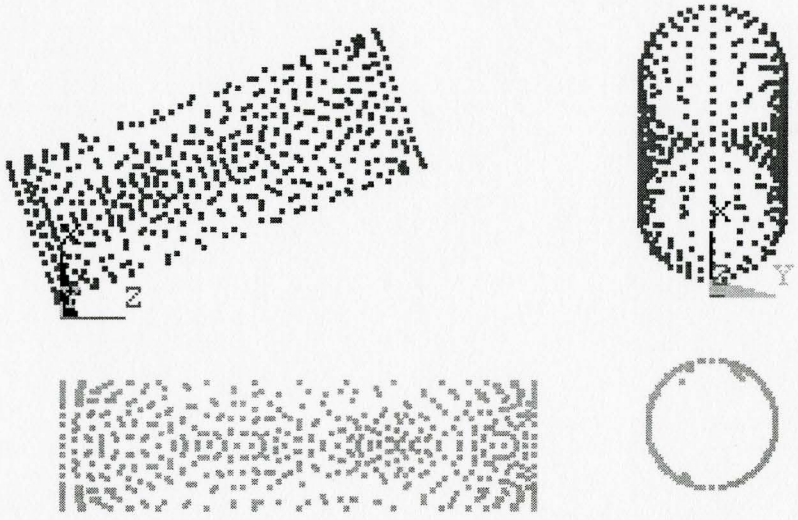

Fig. 3.12 FEM elements used in the simulation

Fig. 3.13 shows the comparison of the experimental results and the FEM results. They do not agree well with each other. For example, when the angle is 15°, the FEM result is 6.8 *pF* while experimental value is 4.8 *pF*. The FEM result is 1.4 times the experimental value.

As discussed previously in Chapter 2, the experimental capacitive value is affected by not only the electrode geometry, but also by the environmental conditions such as changing air temperature or humidity, and surrounding electric fields. Because the FEM model does not include these environmental conditions, it cannot correctly predict the capacitance. As a result, a FEM model unfortunately cannot be used to accurately estimate capacitances for the robot collision avoidance application.

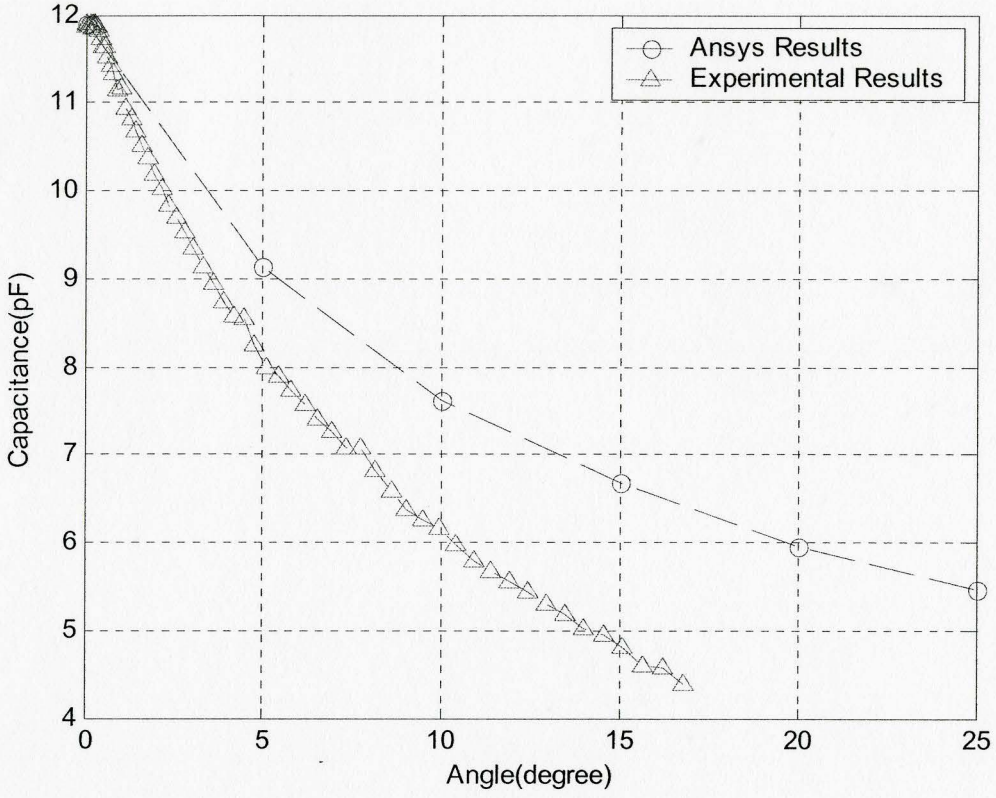

Fig. 3.13 Comparison of FEM and experimental results for the two-link test

3.5 Conclusions

A forward model computes capacitance for a given electrode shape and distance. Capacitance can be calculated using Maxwell's equation. Because this equation is very complex, the approximate formula is often used to calculate the value when parallel flat electrodes are very large compared with the distance between them. The MoM model results suggest that the actual capacitances will be much larger than the values from the approximate formula for this application where the distances are relatively large. The calculation of FEM forward models were done using the commercial software ANSYS. The FEM results were found to be a poor predictor of the experimental results. The

failure of the FEM model to include the true environmental conditions (e.g. air humidity and surrounding electric fields) is the most likely cause of its inaccuracy.

CHAPTER 4

INVERSE MODELING OF CAPACITIVE SENSORS

4.1 Introduction

An "inverse model" of capacitive sensors outputs the geometry of the electrodes given the capacitance values. For the collision avoidance application, the inverse model will be used to estimate the robot link pose from the capacitive sensor readings. This is shown in Fig. 4.1. The robot link pose will be expressed as seven variables: the x, y, and z coordinates for the position and four quaternions (q_0, q_x, q_y, and q_z) for the orientation.

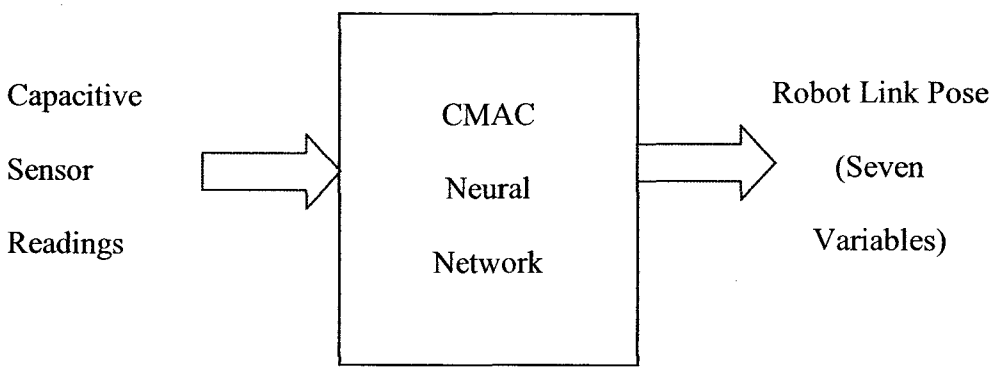

Fig. 4.1 Inverse model

Because of the coupled and highly nonlinear relationship between the capacitance values and the pose variables, the problem cannot be solved by linear methods. The Cerebellar Model Articulation Controller (CMAC) neural network was chosen as the solution approach in this research.

Introduced by J.S. Albus, the CMAC is an artificial neural network modeling the structure and the function of the part of the brain know as cerebellum. CMAC is known to have fast learning and good generation properties [25][26].

Invented by Hamilton in 1843, quaternions are applied in many areas like mechanics, aerospace, theory of relativity and virtual reality [27][28][29]. They are the extension of complex numbers represented as algebraic pairs to triplets and one of the most effective ways to describe the orientation of an object. Although they require an additional variable compared to Euler angles (i.e. 4 vs. 3), quaternions do not suffer from singularity problems. However, when using Euler angles, 90 degrees of the second Euler angle will result in losing of one degree of freedom, the singularity situation. Quaternions will be used to represent the robot orientation for this reason.

4.2 CMAC Neural Network

4.2.1 Introduction

Fig. 4.2 shows the CMAC neural network structure [25][26][30]. The input of CMAC is the set of vectors and the output is a response cell. One element in the association cell vector corresponds to one element in the weight vector.

CMAC neural network performs two subsequent mappings as shown in Fig.4.2. The first nonlinearly maps the input into an association vector, which is a sparse binary vector. The second map calculates the product of the association vector and the weight vector [31].

In CMAC, each input point maps to a unique association vector. The bit is active if and only if the input value is within the support of the corresponding basis function. The first layer encodes the quantized input data and is fixed. With the second layer, trainable weights are updated using the simple least mean squares (LMS) rule. Fig. 4.2 also shows this process, in which only acv_2, acv_4 and acv_6 in the association cell vector are activated, and the corresponding *weights*, $weight_2$, $weight_4$, and $weight_6$, will be trained. For this situation, the output of the neural network will be the average of the three weights.

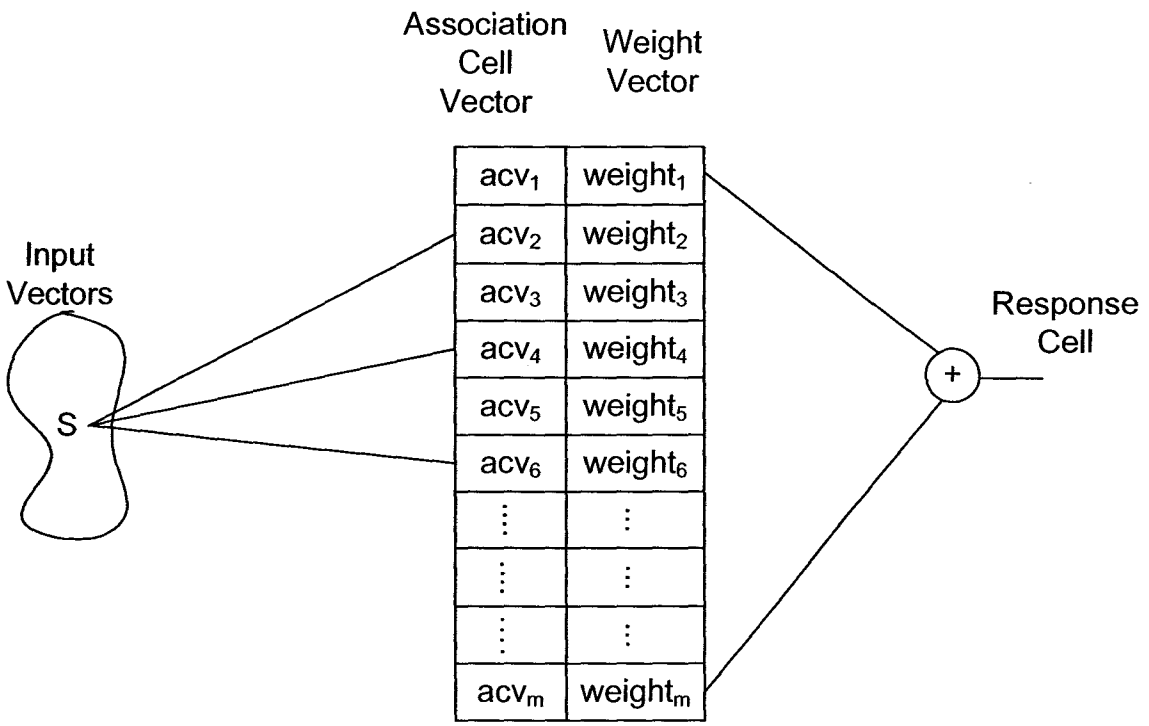

Fig. 4.2 The CMAC structure

Fig. 4.3 depicts the organization of the fields of typical CMAC neural network. Four identical layers represent the generalization parameter, G, which equals four in the Fig. G has to be 2^n, where n is any positive integer. The input is two-dimensional. Each

receptive field in a CMAC is an on-off type entity. If it is excited, the response will be the magnitude of the corresponding weight. If it is not excited, the response is zero. In the training stage, only the weights corresponding to the excited receptive fields will be trained. In Fig. 4.3, only the black square representing one receptive field in each layer is activated. Therefore, the output is the average of the corresponding four weights, and only these four weights will be trained for the training stage.

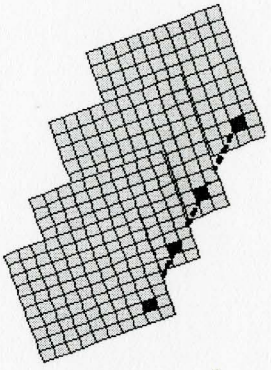

Fig. 4.3 Albus CMAC receptive field distribution for 2D input with generalization parameter $G = 4$

The characteristics of CMAC include [25]:

1) Accepts quantized inputs and gives quantized or continuous outputs.
2) Have a built-in local generalization, the closer two inputs in the input state space is, the more shared elements in the association cell vector.
3) High training speed: Each input only changes the activated G receptive fields, and the output is only contributed by the corresponding G weights.

4) The LMS adaptation rule guarantees a unique minimum in contrast with the relative minimum for the Backpropagation (BP) rule used with multi-layer perceptron neural networks.
5) CMAC can learn a wide variety of functions and obeys superposition in the output space.

4.2.2 CMAC Algorithm

For a real-valued sensory input vector [25][26][30],

$$\vec{s} = [s_1, s_2, \cdots s_N]$$

The normalized integer input vector is obtained by dividing by the quantization parameter Δ,

$$\vec{s}' = [s_1', s_2', \cdots s_N'] = [\text{int}(\frac{s_1}{\Delta_1}), \text{int}(\frac{s_2}{\Delta_2}), \cdots \text{int}(\frac{s_N}{\Delta_N})] \qquad (4.1)$$

For G parallel layers of receptive fields, the normalized virtual N-dimensional address is then

$$Ad_i = [s_1' - ((s_1' - i)\%G), s_2' - ((s_2' - i)\%G), \cdots, s_2' - ((s_N' - i)\%G)]$$

$$= [a_{i1}, a_{i2} \cdots a_{iN}] \qquad \text{i} = 1,2,\cdots,\text{G} \qquad (4.2)$$

where % is the modulus operator, and G is called the generalization parameter.

Pseudorandom hashing function, h, is applied to produce uniformly distributed scalar addresses in the physical weight memory of size, M. The physical address is,

$$Ad_i' = h[a_{i1}, a_{i2} \cdots a_{iN}] \qquad \text{i} = 1,2,\cdots,\text{G} \qquad (4.3)$$

The scalar output $y(s)$ is the average of the addressed weights,

$$y(\vec{s}) = \frac{1}{G}\sum_{i}^{G} W(Ad_i') \tag{4. 4}$$

where $W(Ad_i')$ is the weight corresponding to physical address Ad_i'. It is real type in this application. In the training process, we have sensory input, $\vec{s}$, and the desired output, $y_d(\vec{s})$, and the weight adjustment equation,

$$\Delta W = \beta(y_d(\vec{s}) - y(\vec{s})) \tag{4. 5}$$

where, β is the training gain and $y_d(\vec{s}) - y(\vec{s})$ is the residue errors. After the CMAC has converged, the ongoing errors and the noise from the sensory inputs can cause small adjustments to the weights. The accumulating adjustment can make some of the weights drift to a large positive number and others to a negative number. Then the training system may become numerically unstable. Weight magnitude normalization can solve this problem. This approach places a penalty on large weight magnitudes during the training process. Since the output is an average of the multiple weights, the weight adjustment equation will be changed to,

$$\Delta W = \beta_1(y_d(\vec{s}) - y(\vec{s})) + \beta_2(y(\vec{s}) - W(A_i')) \tag{4. 6}$$

where β_1 and β_2 are the training gains.

4.2.3 CMAC Programs

The CMAC programs were obtained from Dr. W.T. Miller from University of New Hampshire [32]. In this section, several important programs and their integer parameters will be described. They are as follows:

1) ***train_cmac*** **(*cmac_id*, **state*, **respns*, *beta1*, *beta2*)**

This procedure is used to train a previously allocated CMAC. The first parameter is the CMAC handle. The second parameter is a pointer to the vector containing the CMAC training input. The third parameter is a pointer to a vector containing the target (desired) response. The last two parameters set the two training gains of weight adjusting equation. The training gain parameters in the program are bitwise right shift factors (*beta1* = 1 means $\beta_1 = 0.5$, beta1 = 2 means $\beta_1 = 0.25$, etc.).

2) *allocate_cmac* (*num_state*, **qnt_state*, *num_resp*, *num_cell*, *memory*, *field_shape*, *collide_flag*)

This procedure is used to allocate a new CMAC with the specifications given in the parameters. The first parameter is the number of dimensions, *N,* for the CMAC input vectors. The second parameter is a pointer to a vector of dimension, *N,* that defines the quantization parameters (Δ_j in Eq. (4.1)). The third parameter is the dimension of the CMAC output vectors. The fourth parameter is the generalization parameter *G*. This value must be equal to an integer power of 2 (1, 2, 4, 8, 16, ...). The fifth parameter is the total size *M* of the CMAC vector memory (the total number of weights is memory multiply by *num_resp*). The sixth parameter sets the design of the CMAC receptive fields, using the following predefined constants, such as *ALBUS*, *RECTANGULAR*, *SPLINE* etc. *ALBUS* creates conventional on-off receptive fields in a hyperdiagonal arrangement. *RECTANGULAR* creates on-off receptive fields in a uniform arrangement.

3) *cmac_response*(*cmac_id*, **state*, **respns*)

This procedure returns the CMAC vector response to the input vector specified. The first parameter is the CMAC handle. The second parameter is a pointer to the vector

containing the CMAC inputs. The third parameter is a pointer to a vector to receive the CMAC response.

4.3 Quaternions

4.3.1 Mathematical Properties

A quaternion can be defined as [27][28],

$$q = q_0 + \vec{q} \tag{4. 7}$$

where q_0 is a scalar, and vector $\vec{q}$ is called a pure quaternion.

$$\vec{q} = \vec{i}q_1 + \vec{j}q_2 + \vec{k}q_3 \tag{4. 8}$$

For pure quaternions, the basis is, $\vec{i}$ =(1,0,0), $\vec{j}$ =(0,1,0), $\vec{k}$ =(0,0,1)

We can also consider the basis of entire quaternions in the 4D situation as [27],

1=(1,0,0,0), $\vec{i}$ =(0,1,0,0), $\vec{j}$ =(0,0,1,0), $\vec{k}$ =(0,0,0,1)

The orthogonal bases have the properties:

$$\vec{i}^2 = \vec{j}^2 = \vec{k}^2 = ijk = -1 \tag{4. 9}$$

$$\vec{i}\vec{j} = -\vec{j}\vec{i} = \vec{k} \tag{4. 10}$$

$$\vec{k}\vec{i} = -\vec{i}\vec{k} = \vec{j} \tag{4. 11}$$

The Multiplication rules are almost the same as ordinary rules of algebra, like commutative, associative over addition. For a given quaternion,

$$q = q_0 + \vec{i}q_1 + \vec{j}q_2 + \vec{k}q_3 = (q_0, q_1, q_2, q_3) \tag{4. 12}$$

If c is a scalar,

$$cq = cq_0 + \vec{i}cq_1 + \vec{j}cq_2 + \vec{k}cq_3 = (cq_0, cq_1, cq_2, cq_3) \tag{4. 13}$$

The complex conjugate of q is given by

$$\begin{aligned} q^* &= q_0 - \vec{q} \\ &= q_0 - \vec{i}q_1 - \vec{j}q_2 - \vec{k}q_3 = (q_0, -q_1, -q_2, -q_3) \end{aligned} \tag{4. 14}$$

And also

$$qq^* = |q|^2 \tag{4. 15}$$

For two quaternions, $p = (p_0, \vec{p})$, $q = (q_0, \vec{q})$ $q = (q_0, \vec{y})$

$$pq = (p_0 q_0 - \vec{p} \cdot \vec{q}, p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q}) \tag{4. 16}$$

$$p + q = (p_0 + q_0, \vec{p} + \vec{q}) \tag{4. 17}$$

The multiplication of quaternion is associative and distributes over addition [27]. For three quaternions, q_1, q_2, q_3

$$(q_1 q_2) q_3 = q_1 (q_2 q_3) \tag{4. 18}$$

$$(q_1 + q_2) q_3 = q_1 q_3 + q_2 q_3 \tag{4. 19}$$

$$q_1 (q_2 + q_3) = q_1 q_2 + q_1 q_3 \tag{4. 20}$$

For two pure quaternions, $\vec{p} = (p_1, p_2, p_3)$, and $\vec{q} = (q_1, q_2, q_3)$, the scalar product is

$$\vec{p} \cdot \vec{q} = p_1 q_1 + p_2 q_2 + p_3 q_3 \tag{4. 21}$$

And the cross product is

$$\vec{p} \times \vec{q} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix} \tag{4. 22}$$

4.3.2 Quaternion Rotation Operator

The discovery of the quaternion rotation operator made it possible to relate the quaternion to rotation in 3D [27][33]. For any unit quaternion

$$q = q_0 + \vec{q} = \cos\theta + \vec{u}\sin\theta \tag{4. 23}$$

For any vector, $\vec{e} \in R^3$, the action of the operator may be interpreted geometrically as a rotation of the vector $\vec{e}$ through an angle 2θ about $\vec{q}$ as the axis of rotation.

$$L_q(\vec{e}) = q\vec{e}q^* \tag{4. 24}$$

where $L_q()$ is the rotation operator. Suppose $\vec{e}$ is a vector defined in the electrodes frame, and $\vec{w}$ is the same vector defined in the world coordinate frame. It follows that,

$$\begin{aligned}\vec{w} &= L_q(\vec{e}) \\ &= qeq^* \\ &= (q_0 + \vec{q})(0 + \vec{e})(q_0 - \vec{q}) \\ &= (2q_0^2 - 1)\vec{e} + 2(\vec{q}\cdot\vec{e})\vec{q} + 2q_0(q_0 \times \vec{e})\end{aligned} \tag{4. 25}$$

This can be rewritten as

$$\vec{w} = \mathrm{R}\vec{\mathrm{e}}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = R \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{4. 26}$$

where

$$R = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_0q_z & 2q_xq_z + 2q_0q_y \\ 2q_xq_y + 2q_0q_z & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_0q_x \\ 2q_xq_z - 2q_0q_y & 2q_yq_z + 2q_0q_x & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{4. 27}$$

The equations for obtaining the q variables from a given R matrix are presented in Appendix B.

4.4 Robot Link Pose Equations

4.4.1 Introduction

In this research the pose of the 4th link surface of a CRS-F3 robotic arm relative to a stationary robotic arm surface will be modeled. The robot frames are shown in Fig. 4.4. In the Fig., $\{W\}$ denotes the world frame, $\{L4\}$ denotes the link 4 frame, $\{M\}$ denotes the moving electrode frame, which is the surface of link 4, and $\{S\}$ denotes the stationary electrode frame, which is the surface of stationary robot link. The goal is to obtain the seven pose variables from the transformation matrix ${}^{S}T_{M}$. This involves several steps as described in the following sections.

The motion of a robot link includes a 3D translation and a 3D rotation. From [34], if we attach a coordinate frame named M to the link and denote the orthogonal axes as normal, x_M, orientation, y_M, approach, z_M, and the origin of the M frame is P_x, P_y, P_z, then the transformation defining the M frame relative to the coordinate frame S has the form,

$$ {}^{S}T_{M} = \begin{bmatrix} x_{Mx} & y_{Mx} & z_{Mx} & P_x \\ x_{My} & y_{My} & z_{My} & P_y \\ x_{Mz} & y_{Mz} & z_{Mz} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28} $$

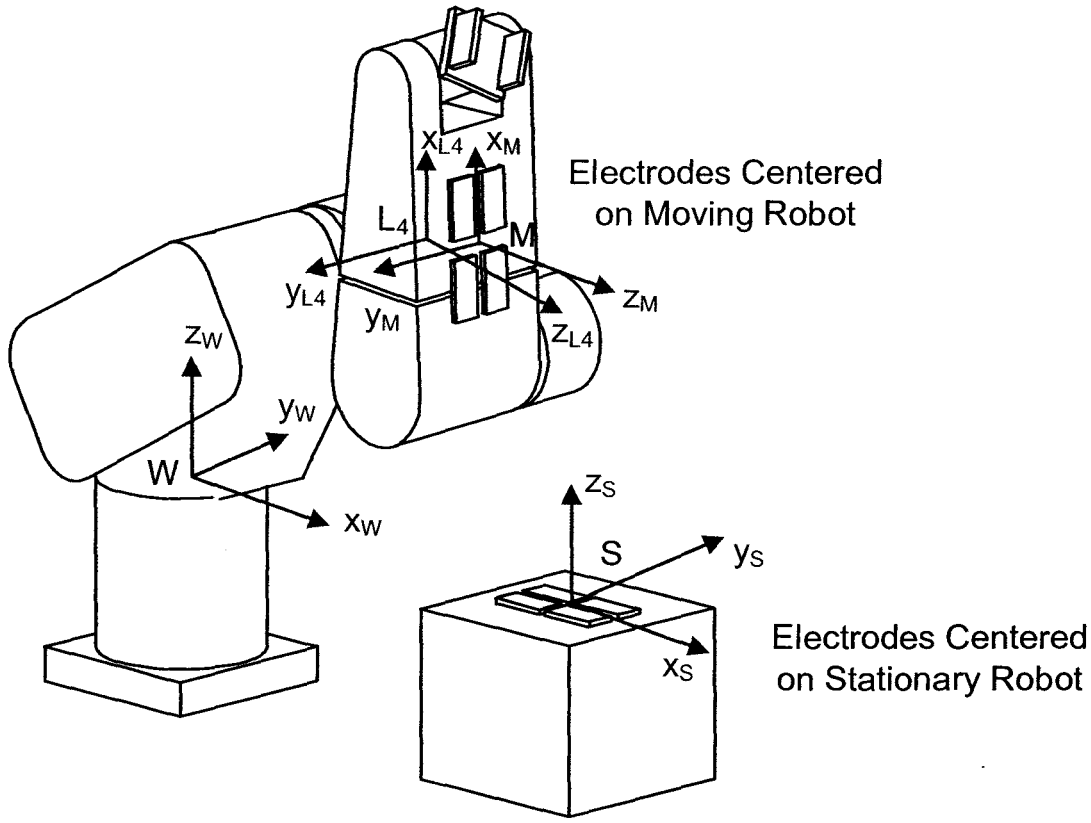

Fig. 4.4 Definitions of world frame, link 4 frame, moving electrode frame and stationary electrode frame

We can write ${}^{S}T_{M}$ in another form:

$$ {}^{S}T_{M} = \begin{bmatrix} & & & P_x \\ & R & & P_y \\ & & & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.29} $$

where, in the matrix ${}^{S}T_{M}$, R is the rotation matrix and P_x, P_y, P_z are the coordinates of the origin of the sensor frame in the world frame.

4.4.2 Equation for Transformation Matrix ${}^{S}T_{M}$

${}^{W}T_{M}$ can be obtained using the formula,

$$ {}^{W}T_{M} = {}^{W}T_{L4} \cdot {}^{L4}T_{M} \qquad (4.30) $$

where ${}^{W}T_{M}$ is the transformation matrix of M relative to world frame, ${}^{W}T_{L4}$ is the transformation matrix of link 4 relative to world frame, and ${}^{L4}T_{M}$ is the transformation matrix defining moving electrodes relative to link 4.

The transformation matrix can also be obtained using the formula,

$$ {}^{W}T_{M} = {}^{W}T_{S} \cdot {}^{S}T_{M} \qquad (4.31) $$

where ${}^{W}T_{S}$ is the matrix defining stationary arm frame relative to the world frame, and ${}^{S}T_{M}$ is the matrix of moving electrodes relative to stationary electrodes. Equating (4.30) and (4.31) gives the desired result,

$$ \begin{aligned} {}^{S}T_{M} &= {}^{W}T_{S}^{-1}\,{}^{W}T_{M} \\ &= {}^{W}T_{S}^{-1}\,{}^{W}T_{L4}\,{}^{L4}T_{M} \end{aligned} \qquad (4.32) $$

Eq. (4.32) is the matrix expressing the sensor system detecting the relative position and orientation between the moving robotic arm and the stationary robotic arm. It can be calculated by using Eq. (4.32) after obtaining ${}^{W}T_{L4}$, ${}^{L4}T_{M}$ and ${}^{W}T_{S}$.

4.4.3 Equation for Transformation Matrix ${}^{W}T_{L4}$

The frames of the sensor system are shown in three views in Fig.4.5. From the world frame (frame 0) to link 4 frame, the homogeneous transformation matrix can be

obtained using the Denavit - Hartenburg (D-H) method. The D-H parameters are found using the procedure in [34] and the parameters are listed in Table 4.1.

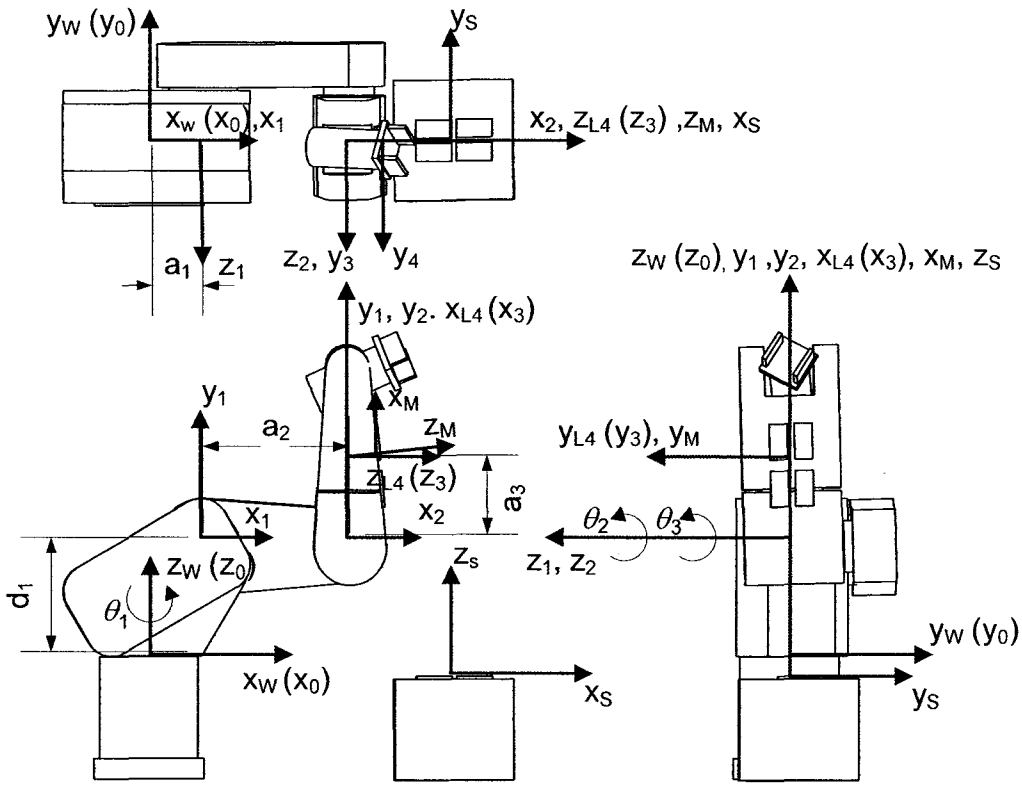

Fig. 4.5 The frames of the sensor system

Table 4.1 The D-H parameters for CRS robot

$n+1$	θ	d	a	α
1	θ_1	d_1=134 *mm*	a_1= 100 *mm*	+90°
2	θ_2	0	a_2= 265 *mm*	0°
3	θ_3	0	a_3= 85 *mm*	+90°

The joint variables are θ_1, θ_2 and θ_3. If $C\theta$ stands for *cos(θ)* and $S\theta$ stands for *sin(θ)*, substituting the D-H parameters into equation in [34] gives the A matrices.

$$A_1 = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 100C\theta_1 \\ S\theta_1 & 0 & -C\theta_1 & 100S\theta_1 \\ 0 & 1 & 0 & 134 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.33)$$

$$A_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & 265C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & 265S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.34)$$

$$A_3 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & 85C\theta_3 \\ S\theta_3 & 0 & -C\theta_3 & 85S\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.35)$$

$$\begin{aligned} {}^{W}T_{L4} &= A_1 A_2 A_3 \\ &= \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 100C\theta_1 \\ S\theta_1 & 0 & -C\theta_1 & 100S\theta_1 \\ 0 & 1 & 0 & 134 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & 265C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & 265S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & 85C\theta_3 \\ S\theta_3 & 0 & -C\theta_3 & 85S\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} C\theta_1 C(\theta_2+\theta_3) & S\theta_1 & C\theta_1 S(\theta_2+\theta_3) & C\theta_1(85C(\theta_2+\theta_3)+265C\theta_2+100) \\ S\theta_1 C(\theta_2+\theta_3) & -C\theta_1 & S\theta_1 S(\theta_2+\theta_3) & S\theta_1(85C(\theta_2+\theta_3)+265C\theta_2+100) \\ S(\theta_2+\theta_3) & 0 & -C(\theta_2+\theta_3) & 85S(\theta_2+\theta_3)+265S\theta_2+134 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (4.36)$$

4.4.4 Equation for Transformation Matrix ${}^{L4}T_M$

From L_4 to moving electrode center, there were a translation about z-axis and a rotation about y-axis. This can be solved using homogeneous transfer matrix. First, frame L_4 was translated along z by 51.5 *mm*. Then it was rotated about its current y-axis by φ = 3.72° to M frame, the electrode coordinate system. The matrix will be,

$${}^{L4}T_M = Trans(0,0,51.5) \times Rot(y_4, \varphi)$$

$$= \begin{bmatrix} C\varphi & 0 & S\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -S\varphi & 0 & C\varphi & 51.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.37}$$

4.4.5 Equation for Transformation Matrix ${}^{W}T_{S}$

From stationary frame to the world frame, it is a pure translation. Because the origin of the stationary robotic arm is located at $[424.5, 0, -29.5]^{T}$ in world frame, the transformation matrix is,

$${}^{W}T_{S} = Trans(424.5,0,-29.5) = \begin{bmatrix} 1 & 0 & 0 & 424.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -29.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.38}$$

The inverse of this homogeneous matrix can be obtained as,

$${}^{W}T_{S}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -424.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 29.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.39}$$

4.4.6 Substituting the Angles from Robot Controller into the D-H Equations

Because the frame of CRS-F3 robot was different from the frame used for D-H, when substituting the angles from robot controller into the angles from D-H method, the transformation function was needed. The joint 1 and 3 angles are the same. Joint 2 angle in D-H is 90 degrees more than that for the controller. If θ_{forD-H} represents the angle in D-H and $\theta_{fromController}$ represents the angle sent from the robot controller, their relationship can be described in Eq. (4.40) to Eq. (4.42).

$$\theta_{1\,forD-H} = \theta_{1\,fromController} \tag{4.40}$$

$$\theta_{2\,forD-H} = \theta_{2\,fromController} + 90^{\circ} \tag{4.41}$$

$$\theta_{3\,forD-H} = \theta_{3\,fromController} \tag{4.42}$$

4.4.7 Summary of Link Pose Solution

The pose matrix can be obtained from the matrixes solved above as follows,

$${}^{S}T_{M} = {}^{W}T_{S}^{-1}\,{}^{W}T_{L4}\,{}^{L4}T_{M}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -424.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 29.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} C\theta_1 C(\theta_2+\theta_3) & S\theta_1 & C\theta_1 S(\theta_2+\theta_3) & C\theta_1(85C(\theta_2+\theta_3)+265C\theta_2+100) \\ S\theta_1 C(\theta_2+\theta_3) & -C\theta_1 & S\theta_1 S(\theta_2+\theta_3) & S\theta_1(85C(\theta_2+\theta_3)+265C\theta_2+100) \\ S(\theta_2+\theta_3) & 0 & -C(\theta_2+\theta_3) & 85S(\theta_2+\theta_3)+265S\theta_2+134 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} C\varphi & 0 & S\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -S\varphi & 0 & C\varphi & 51.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta_1 C(\varphi+\theta_2+\theta_3) & S\theta_1 & C\theta_1 S(\varphi+\theta_2+\theta_3) & \begin{array}{l} C\theta_1(85C(\theta_2+\theta_3)+51.5S(\theta_2+\theta_3) \\ +265C\theta_2+100)-424.5 \end{array} \\ S\theta_1 C(\varphi+\theta_2+\theta_3) & -C\theta_1 & S\theta_1 S(\varphi+\theta_2+\theta_3) & \begin{array}{l} S\theta_1(85C(\theta_2+\theta_3)+51.5S(\theta_2+\theta_3) \\ +265C\theta_2+100) \end{array} \\ S(\varphi+\theta_2+\theta_3) & 0 & -C(\varphi+\theta_2+\theta_3) & \begin{array}{l} 85S(\theta_2+\theta_3)-51.5C(\theta_2+\theta_3) \\ +265S\theta_2+163.5 \end{array} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4.43)

Next this matrix is subdivided into a rotation matrix, R, and three position variables, P_x, P_y and P_z, as shown by Eq. (4.29). The remaining four pose variables (q_x, q_y, q_z, q_0) are solved from R using the equations given in Appendix B.

CHAPTER 5

EXPERIMENTAL RESULTS FOR INVERSE MODELING

5.1 Introduction

In Chapter 4, our inverse modeling method was introduced. In this chapter, the method will be experimentally verified. After describing the experimental setup and procedure, numerous test results are presented and discussed.

5.2 Experimental Setup

The experimental setup consisted of four main components: PC, capacitive sensor, robot controller and CRS-F3 robot. The software for the PC was written in the C language while the software for the robot was written in the RAPL-3 language.

5.2.1 Setup Block Diagram

Fig.5.1 shows the setup block diagram. There were four electrodes on the moving robotic arm and four electrodes on the stationary robotic arm. Each electrode is made from aluminum foil on a foam board backing. One pair of electrodes was used as a "reference pair", which will be discussed later. All of the electrodes are connected to a multiplexor board and then two of them are selected to connect to the capacitance sensing circuit. The PC communicated with the capacitance sensing circuit using the parallel port, which was also used to send a collision avoidance signal to the general purpose input-output (GPIO) port of the robot controller in Chapter 6. A serial communication link

between the PC and robot controller was used to upload robot programs, command joint angles and read the actual joint angles.

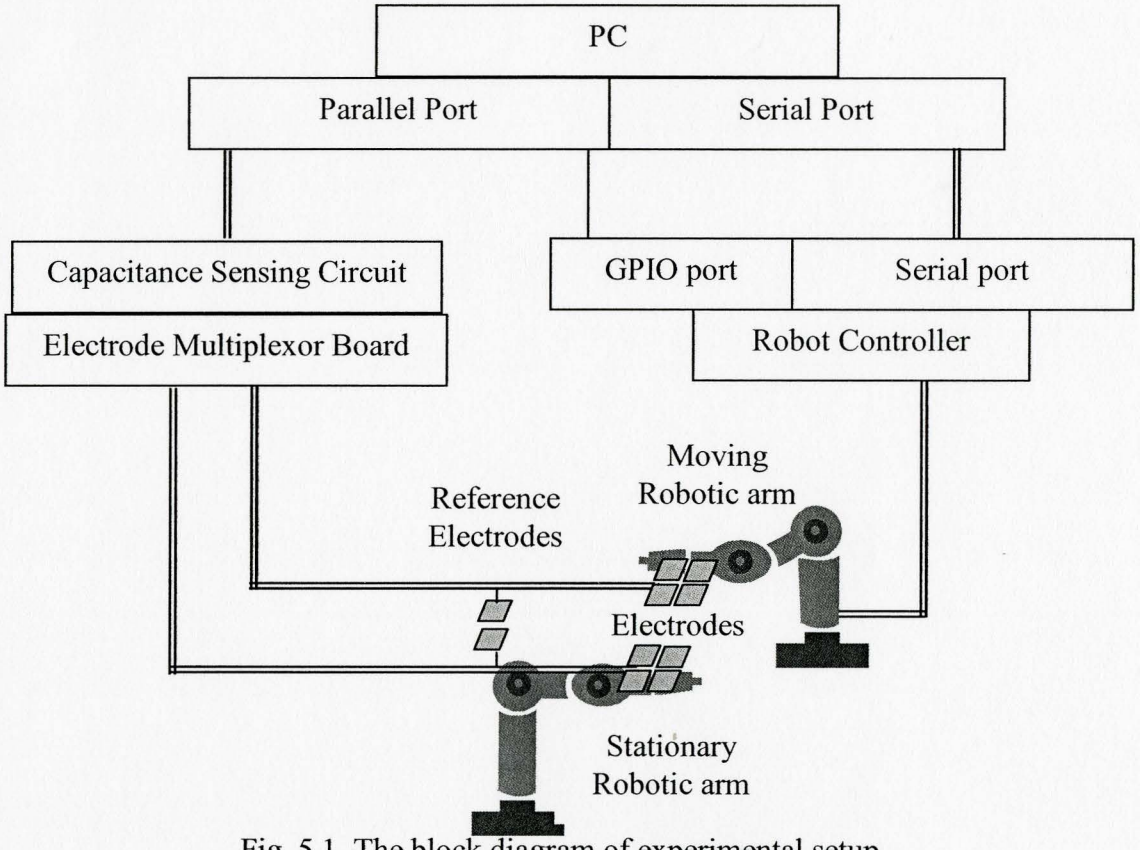

Fig. 5.1 The block diagram of experimental setup

5.2.2 Electrodes Setup

Fig.5.2 shows the electrodes setup. Four electrodes 1I to 4I were mounted on the CRS-F3 robotic arm and four electrodes 1E to 4E were placed on the stationary robotic arm. In our experiments, the stationary robotic arm was simulated using a wooden box. The sizes of the electrodes are listed on table 5.1 and table 5.2.

Table 5.1 Size of the electrodes on the moving robotic arm

Electrodes	1I	2I	3I	4I
Width	60 *mm*	61 *mm*	59 *mm*	61 *mm*
Length	122 *mm*	125 *mm*	120 *mm*	125 *mm*

Table 5.2 Size of electrodes on the stationary robotic arm

Electrodes	1E	2E	3E	4E
Width	62 *mm*	62 *mm*	62 *mm*	61 *mm*
Length	124 *mm*	121 *mm*	121 *mm*	120 *mm*

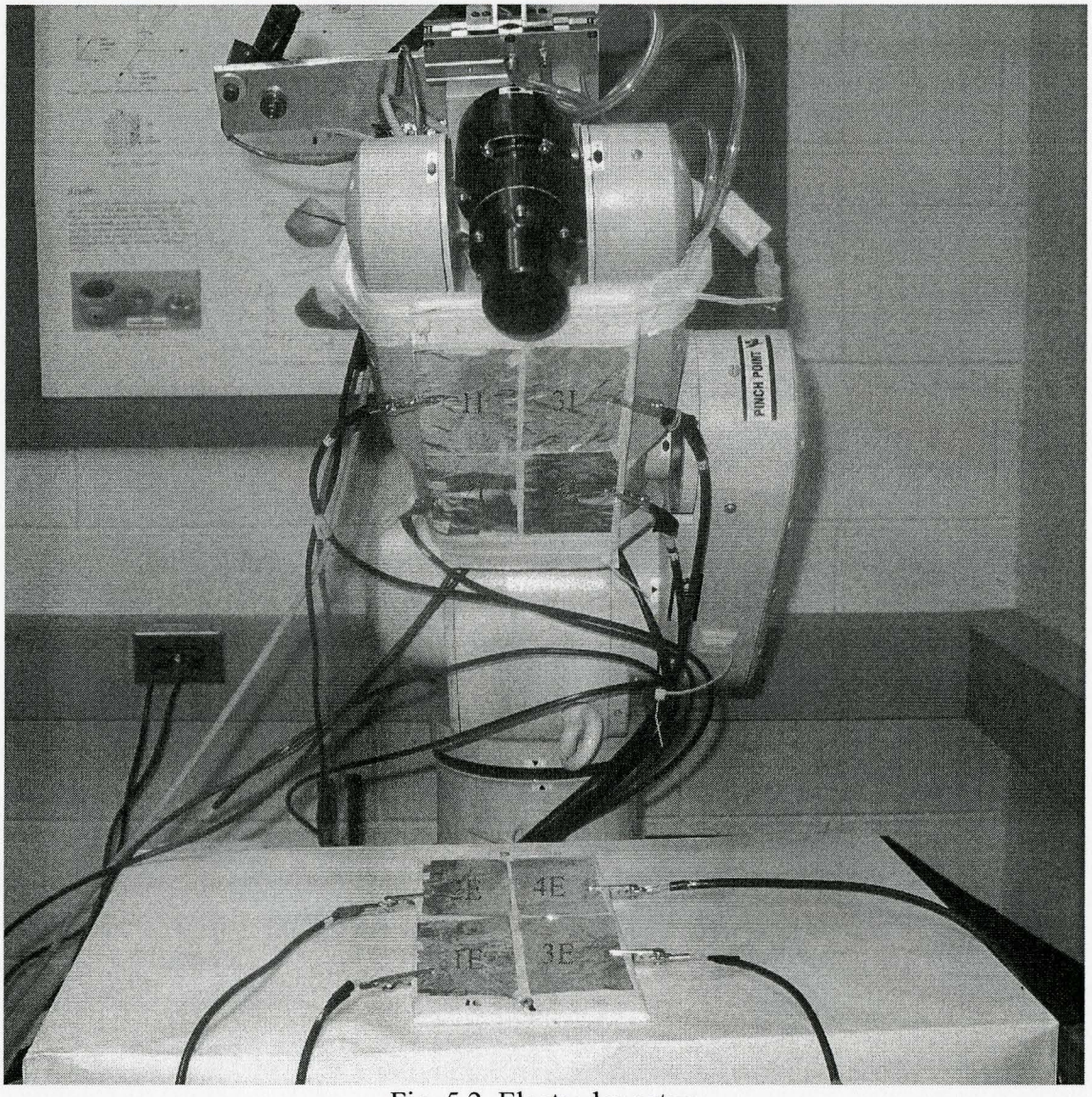

Fig. 5.2 Electrodes setup

The gap between adjacent electrodes is about 5 *mm*. If one of the electrodes from the moving arm is selected and another electrode is selected from the stationary arm, the total number of electrode combinations is 16.

5.3 Robot Motion and Capacitance Trends with the Motion

5.3.1 Motion Ranges and Pseudo Code

The moving electrodes were mounted on link 3 of the CRS-F3 robot. The motion of this link is determined by the angles of the first three robot joints, i.e. θ_1, θ_2, and θ_3. These angles were incremented to move the link horizontally and vertically while keeping the electrodes on the two robotic arms roughly parallel. The minimum perpendicular distance between the electrodes was 15.6 *mm* and the maximum distance was 67.4 *mm*. The joint motion ranges are listed in Table 5.3. The pseudo code for commanding the joint motions is listed in Table 5.4.

Table 5.3 The joint movement ranges used in the experiments

Joint	Joint 1	Joint 2	Joint 3
Minimum Angle	-11°	-110.5°	6.5°
Maximum Angle	9°	-99°	18°

Table 5.4 Pseudocode for commanded robot motion

direction = positive

$\theta_1 = -11^0$

$\theta_2 = -110.5^0$

$\theta_3 = 18^0$

$\theta_4 = 0^0$

while $\theta_2 < -99^0$

$\theta_2 = \theta_2 + 0.25^0$

$\theta_3 = \theta_3 - 0.25^0$

if direction = positive then

while $\theta_1 < 9^0$

$\theta_1 = \theta_1 + 0.25^0$

Send motion command to robot ()

Wait for robot to stop moving ()

Read capacitance values ()

end while

direction = negative

else

while $\theta_1 > -11^0$

$\theta_1 = \theta_1 - 0.25^0$

Send motion command to robot ()

Wait for robot to stop moving ()

Read capacitance values ()

end while

direction = positive

end if

end while

5.3.2 Motion Plotting

The joint angles generated from the pseudo code are sent as command angles to the CRS-F3 robot. After the robot is commanded to move and reached the desired location, the three actual joint angles, θ_1, θ_2, and θ_3, are recorded. Fig. 5.3 shows example plots of the actual angles. Note that to clearly show the trends, only a 400 point subset of the

recorded data is shown. Fig 5.3 shows θ_1 varying from -11° to 9°, θ_2 increasing from -110.5° to 99°, and θ_3 decreasing from 18° to 6.5°. The Fig also shows the definition of cycle. Substituting the actual robot angles into Eq. (4.43) and using the equations in Appendix B, the seven pose variables are obtained. Plots of the x, y, and z coordinates are given in Fig. 5.4. Fig. 5.5 shows the quaternion components, q_x, q_y, q_z, and q_0.

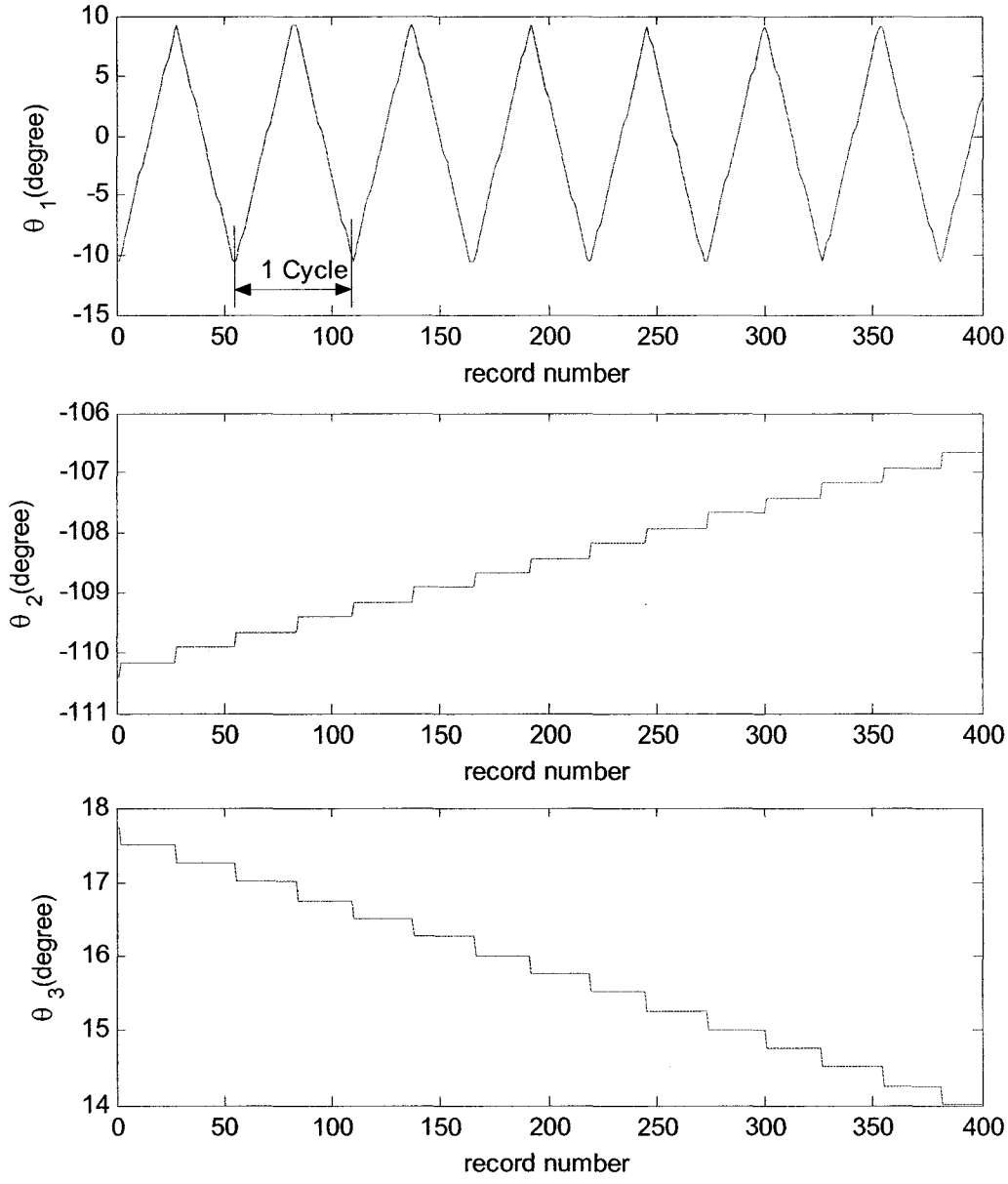

Fig. 5.3 Actual joint angles from part of an experiment

In Fig. 5.4, the x-coordinate values change periodically with the sweeping of joint 1, peaking when $\theta_1 = 0$. Because of the changing angles of joint 2 and joint 3, the x values

are different for each cycle. The *y*-coordinate values have different properties. They vary almost linearly with the value of joint 1. The peak values change from –81.7 *mm* to +70.7 *mm*. The plotting for z-coordinate clearly shows the increasing in *z* direction with the changing of the joint 2 and joint 3 angles. However, the values of *z*-coordinate do not change with joint 1. They remain constant during each cycle.

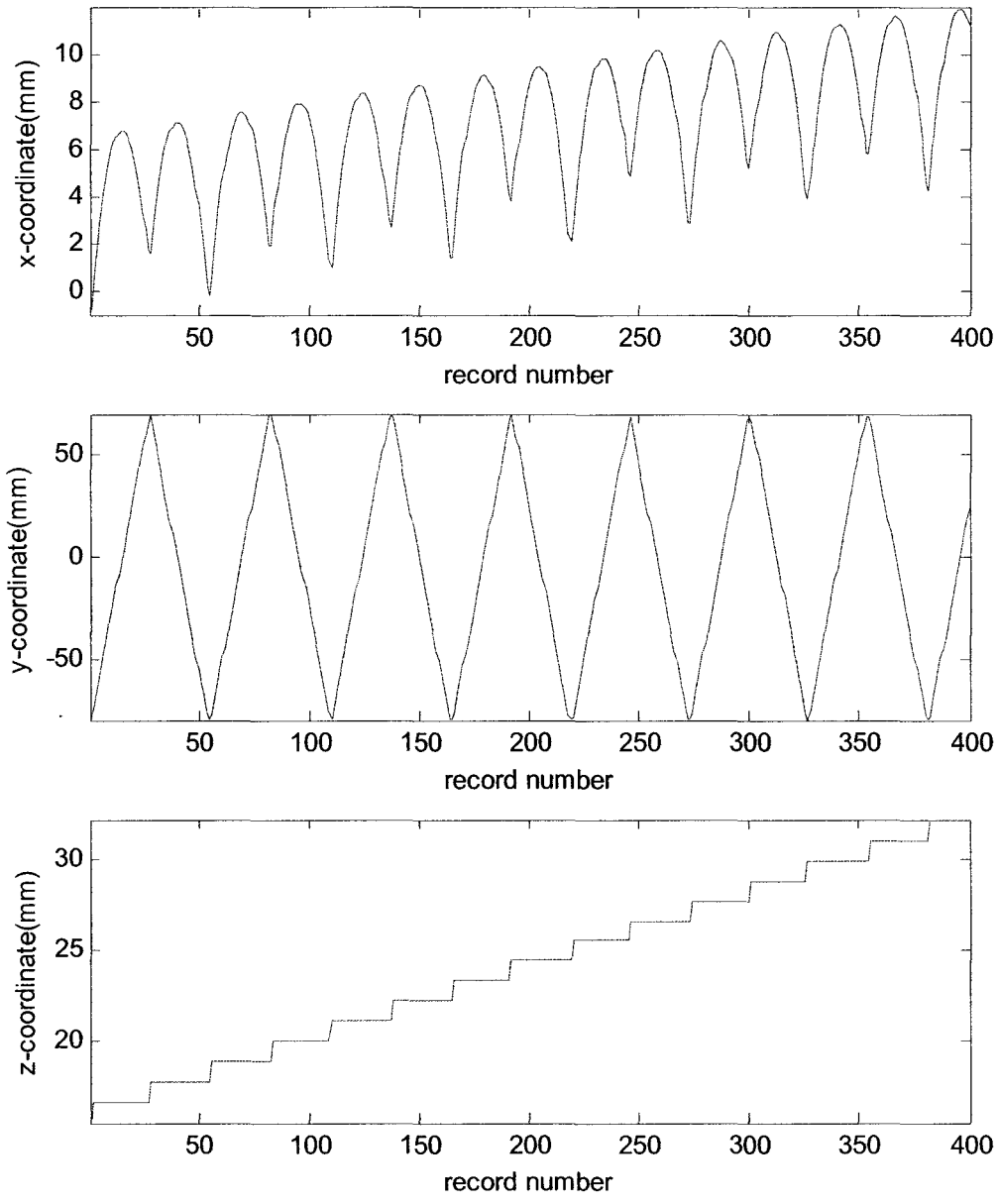

Fig. 5.4 *X*, *y* and *z* coordinates corresponding to the joint angles shown in Fig. 5.3

The quaternions in Fig. 5.5 show the orientation changing for the movement of the robotic arm. Variable q_y has the largest range from about -10^{-2} to 10^{-2}. Variables q_0, q_x, and q_z have the magnitude at around10^{-3} level.

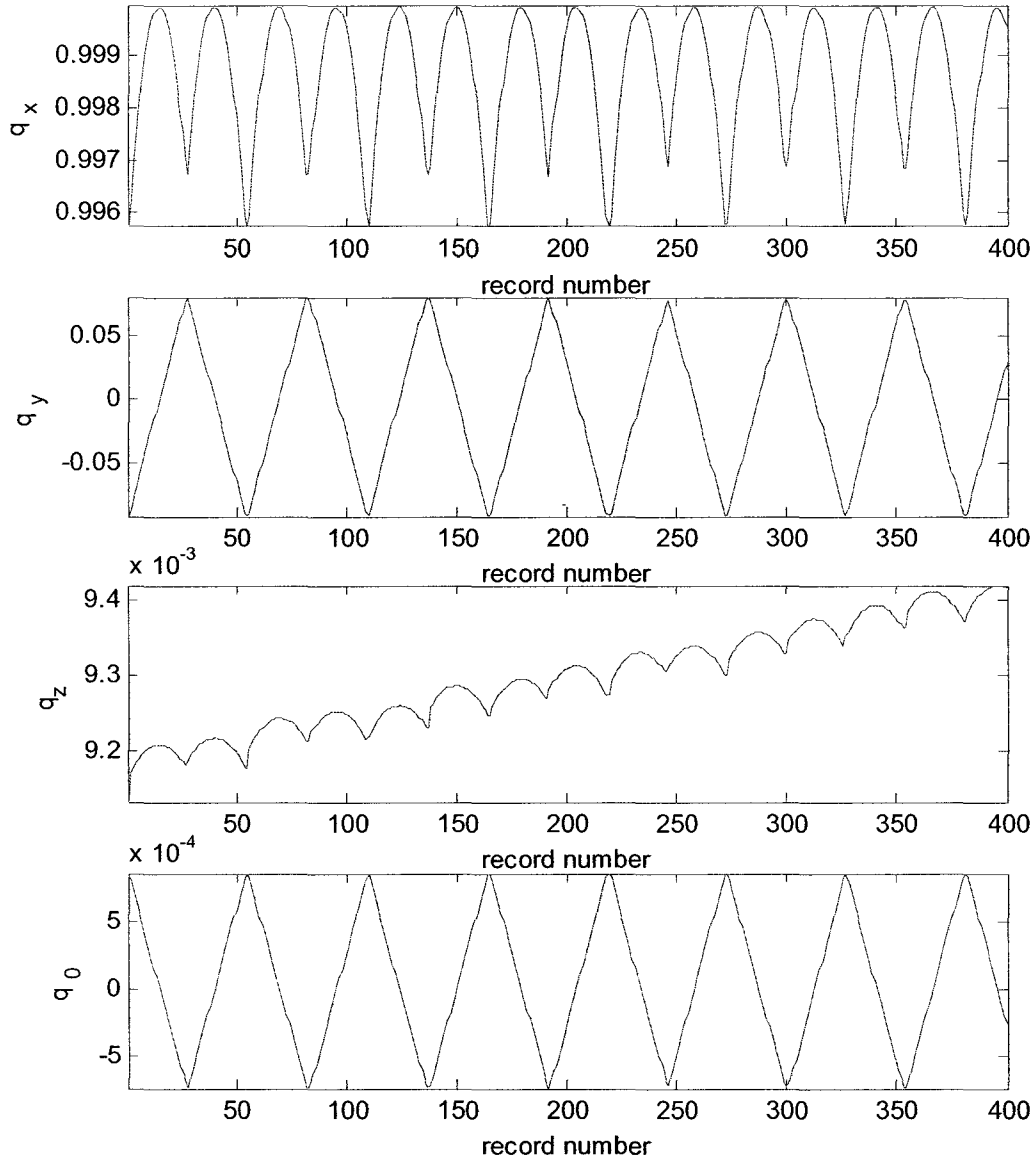

Fig. 5.5 Quaternions, q_x, q_y, q_z and q_0 corresponding to the joint angles shown in Fig. 5.3

5.3.3 Capacitance Trends

Fig.5.6 shows a typical capacitance measurement when joint one sweeps from -10.1 degrees to +9 degrees. The electrode geometry corresponding to points *A*, *B*, and *C* is shown in Fig. 5.7.

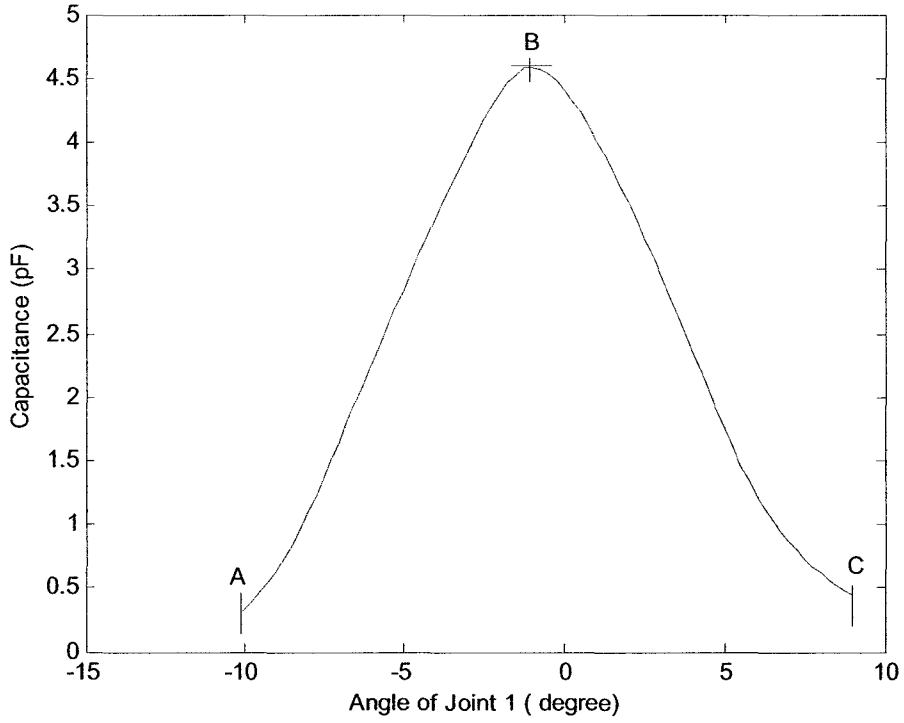

Fig. 5.6 Example capacitance measurement for joint 1 varying and joint 2 and 3 fixed

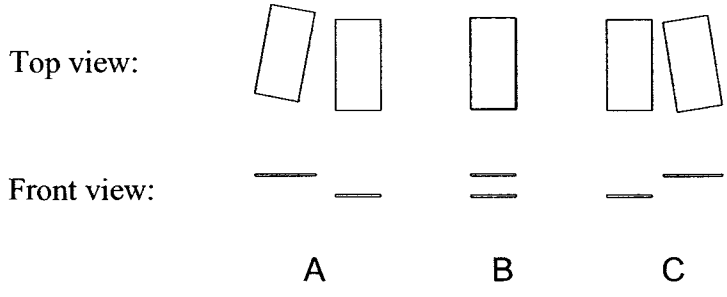

Fig. 5.7 Electrode geometry corresponding to Fig.5.6

The results show that the capacitance peaks when the electrodes fully overlap and drops as the overlap decreases. They also show that a single capacitance measurement cannot be used to differentiate between cases *A* and *C*.

Fig.5.8 shows all sixteen capacitance measurements when joint 1 sweeps from -10.1° to +9° while joint 2 and joint 3 are fixed. Each line represents one pair of electrodes. The actual angle of joint 2 is -109.2° and that of joint 3 is 17.0°[1].

In the Fig., the changing trends are different for different combinations of electrodes because the electric field is different for combinations of electrodes at the different angles of joint 1. With the movement of the robotic arm, the distances between some of the combinations increase and the capacitances are decreasing and vice-versa.

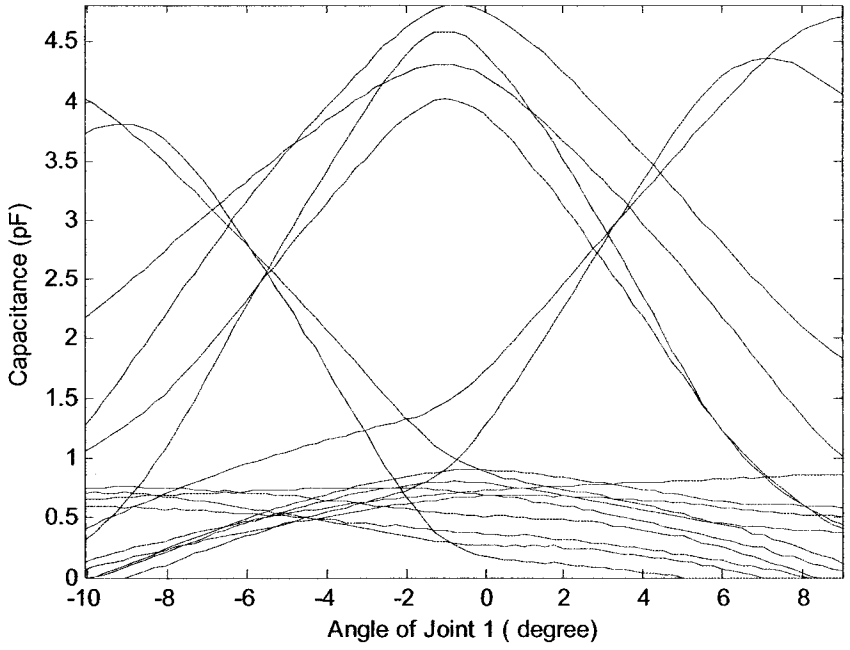

Fig. 5.8 All sixteen capacitance measurements for joint 1 varying and joints 2 and 3 fixed

[1] Based on the incremental encoders, note that the actual angle does not always match the commanded angle

5.4 Environmental Effects

Fig.5.9 shows 16 capacitance readings recorded on different dates as indicated by the dashed and solid lines. From the Fig., we can see the influence of the air temperature and humidity on the capacitances. Especially, there is a big difference in the peak values that will have a greater effect on the CMAC training results.

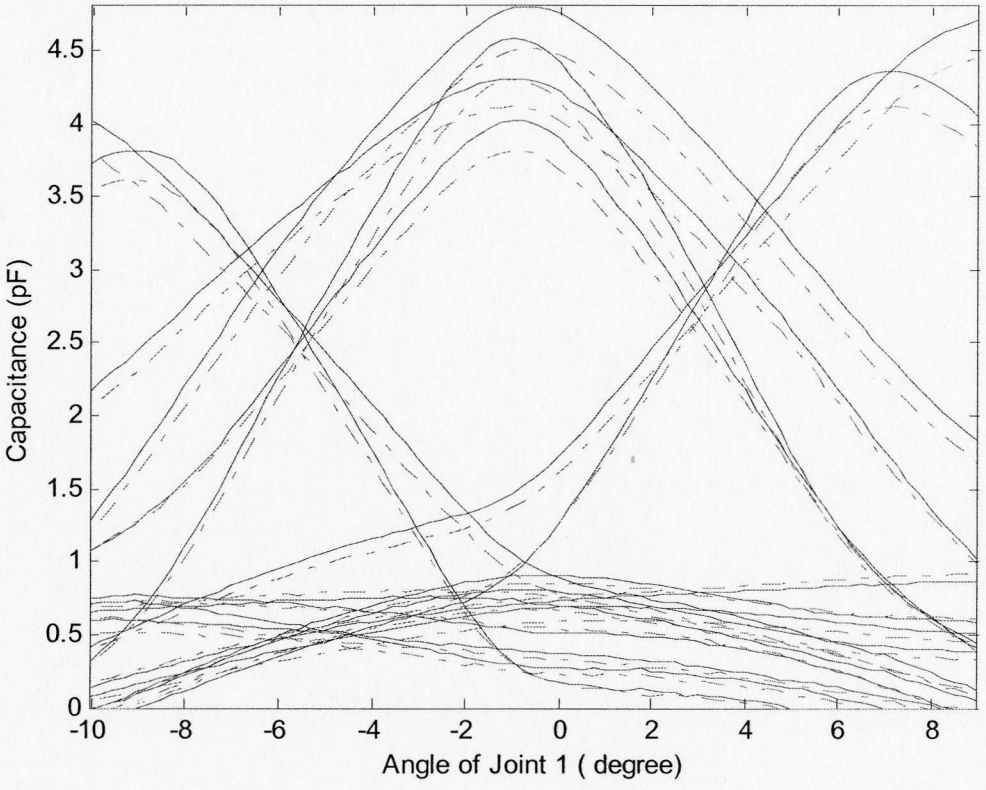

Fig. 5.9 Capacitances measured by all 16 sensors taken on two different days

5.4.1 Reference Pair of Electrodes

A pair of 78 *mm* by 78 *mm* square parallel electrodes with gap distance of 43 *mm* were mounted on the shoulder of the robot. These formed a "reference pair" that was used to compensate for changes in the environment.

From the approximate capacitance formula in chapter 3, because the geometry factor is fixed, the capacitance of the reference pair will be the function of environment, i.e. temperature and humidity of the air; and the ambient electric field.

Fig.5.10 is the plot of the capacitance of reference pair measured on different days. The change shown here is roughly 6%. Even larger changes would be observed when there is a change in season.

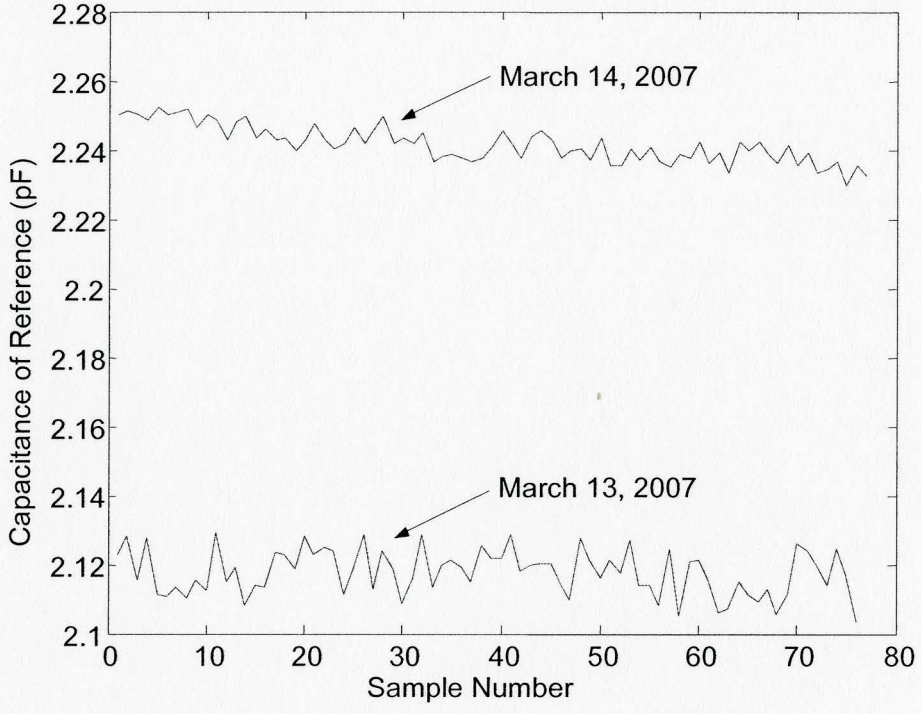

Fig. 5.10 Capacitance of reference pair measured on different days, sampling rate was 50 *Hz*

Fig.5.11 shows the reference electrodes mounted on the back of the robot link in the experiment.

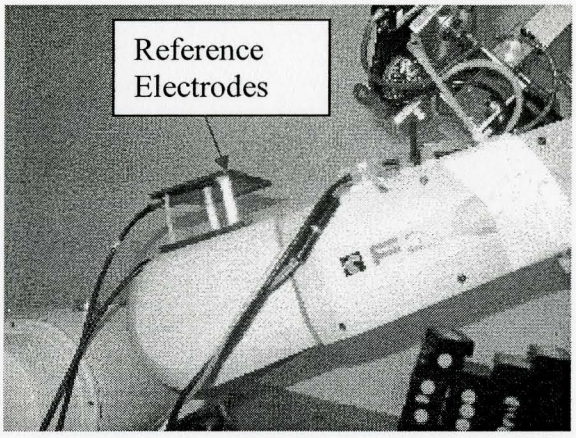

Fig. 5.11 Reference pair of electrodes

5.4.2 Normalization of Sensor Readings

The capacitance readings were normalized by dividing by the reference pair reading. Fig. 5.12 shows the result after normalization of the data in Fig. 5.9. The effectiveness of the normalization in reducing the environmental effects is very obvious. This normalization method was used for all subsequence capacitance measurements.

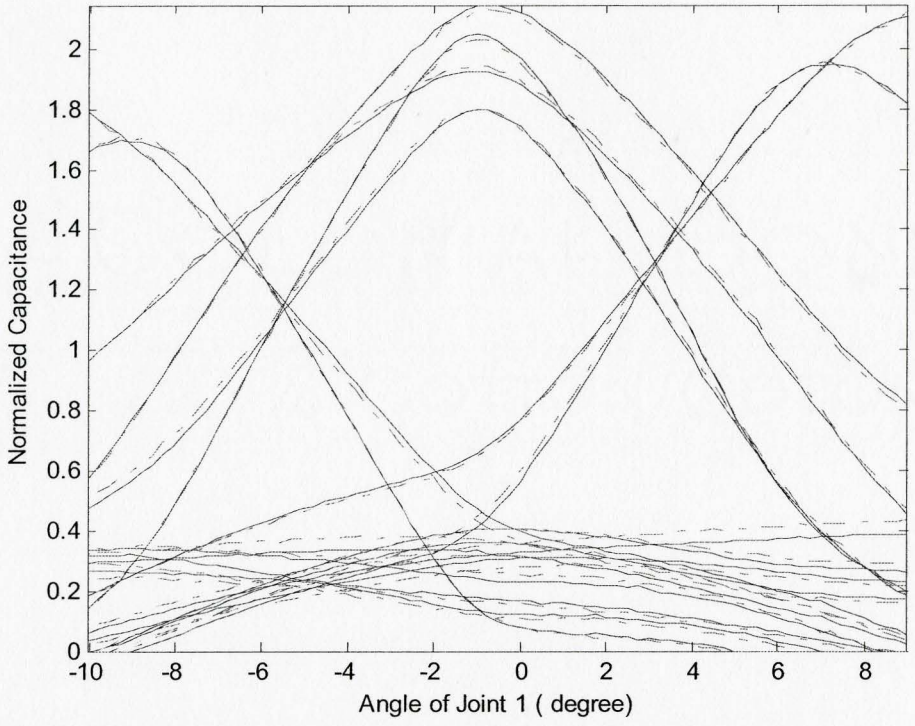

Fig. 5.12 Capacitance reading after normalization

5.5 CMAC Training Process

The whole sensory dataset was collected while moving the robot in the manner described in section 5.3. The set was then separated into two different parts, the training dataset and the testing dataset. The training dataset was used for the CMAC neural network training process, while the testing dataset was used to test the prediction ability of the neural network. The training errors are the indictors of the training results.

Two different modes may be used for training, the pattern mode and the batch mode. In the pattern mode the weight updating was performed after the presentation of each training example. In the batch mode the weight updating was done after the presentation of all the training examples that constitute an epoch [35]. The mode used in this research was the pattern mode.

It is important to note that a separate CMAC network was created for each of the seven pose variables, i.e. x, y, z, q_0, q_x, q_y and q_z. Each network has the 16 normalized capacitance readings as its inputs and one of the pose variables as its output.

5.5.1 CMAC Training Errors

Two types of the error metrics are used to evaluate the CMAC learning results. One is mean squared error and the other is relative error.

1. MSE

Mean Square Error (MSE) is defined as,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_{id} - y_i)^2 = \frac{1}{N}\sum_{i=1}^{N} e_i^{\,2} \tag{5. 1}$$

where, N is the total number of testing data, y_i is the network output, y_{id} is the desired output, and e_i is the error.

2. Relative Errors

The first step was to group the errors into the vector,

$$\vec{e} = [e_1 \quad e_2 \quad \cdots \quad e_i \quad \cdots \quad e_N] \quad i = 1 \cdots N \tag{5. 2}$$

In the following formulas, *abs()* is the absolute value function.

i. Maximum Absolute Errors (MAXAE)

Maximum absolute errors indicate the worst situation in the testing process, and are defined as

$$MAXAE = \max(abs(\vec{e})) \tag{5. 3}$$

ii. Minimum Absolute Errors (MINAE)

The best match situation is indicated by minimum absolute errors, defined as

$$MINAE = \min(abs(\vec{e})) \tag{5. 4}$$

iii. Mean Absolute Errors (MAE)

To show the average or expectation errors during the testing or training process, the mean absolute error is calculated using

$$MAE = mean(abs(\vec{e})) \tag{5. 5}$$

iv. Relative Absolute Error (RAE)

The relative absolute error is the percentage of maximum absolute error divided by maximum absolute value of the ideal output data. The maximum absolute value of the ideal output is

$$\max(abs(\vec{y}_d)) \tag{5. 6}$$

where $\vec{y}_d$ is the vector of ideal output values.

The relative maximum error is,

$$RAE = \frac{\max(abs(\vec{e}))}{\max(abs(\vec{y}_d))} \times 100\% \tag{5. 7}$$

v. Relative Mean Error (RME)

The relative mean error is calculated by the average of the errors divided by maximum absolute input data, which are training data in the following.

$$RME = \frac{Sum(\vec{e})}{N \max(abs(\vec{y}_d))} \tag{5. 8}$$

vi. Other metrics

Other metrics used is the maximum and minimum value of the ideal output data.

$$\min(\vec{y}_d) \tag{5. 9}$$

$$\max(\vec{y}_d) \tag{5. 10}$$

where, $\vec{y}_d$ is the ideal output data vector.

5.5.2 Training and Testing Datasets

In this research, three different datasets were used.

The first used all of the data, which had 3649 points[2]. 19 testing data points were then randomly collected and the remaining 3630 points were used for the training. This dataset was termed the "whole dataset".

To reduce the training time, a smaller dataset was used as the second one. The 3649 points were reduced to 1240 by taking one point in every three, and also keeping the peak points. The remaining 2409 points were used as the testing dataset. This may cause the problem of underestimating the error because no peak values were used in the testing. However, if we consider the very small difference between the peak values and their neighbors, this problem is unlikely. This dataset was termed the "sampled dataset"

The third dataset used the first 1019 points of the collected data. A 19 testing points set was then randomly collected from the 1019 examples. The remaining 1000 examples were used as the training dataset. Because this dataset was a subset in which the variables

[2] Based on the motion ranges given in section 5.3, there should be 80x46=3680 points. The number of collected points is less since the actual angles rather than the commanded angels were used to stop the movements.

and sensory data vary over a smaller range than the whole dataset, the errors were expected to be relatively small. This dataset was termed the "reduced dataset".

5.5.3 Training Parameters

In order to get a good training result, it is necessary to determine the parameters to use with the CMAC training program discussed in chapter 4.

1. The parameters for new CMAC allocation

1). Quantization parameters

The parameters were used for CMAC to form normalized integer input as described in Chapter 4. Table 5.5 shows the quantization parameters (Q. P.) for all 16 normalized capacitances to train the CMAC. The first row in the table is the input sequence number. The second row is the maximum normalized capacitance (M. N. C.) corresponding to each input. The third row is the quantization parameters for the CMAC.

Table 5.5 Quantization parameters for each input

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
M. N. C.	7	1.3	6.6	1.4	1.1	7.4	1.3	7.2	5.7	1.3	5.8	1.3	1.1	6.7	1.2	6.3
Q. P.	4	1	4	1	1	4	1	4	4	1	4	1	1	4	1	4

2). Generalization parameter G

This parameter has to be a power of two and defines the number of layers in CMAC. More layers can model more complex data. In this research $G = 1{,}024$.

2. The parameters for each pose variable

The parameters listed in table 5.6 are only for the electrode combinations 4 by 4 and 3 by 3. These electrode combinations are described further in section 5.6. With different

combinations, the parameters may be changed to get good training results. For example, x-coordinate has different sensor scale parameters.

Table 5.6 The training parameters

Link pose variable	Link pose variable scale	Sensor reading scale	CMAC receptive fields' geometry	β_1	β_2
x (4 by 4)	1.00E+04	200	Rectangular	1	50
x (3 by 3)	1.00E+04	100	Rectangular	1	50
y	1.00E+04	600	Rectangular	1	50
z	1.00E+04	500	Rectangular	1	50
q_0	1.00E+06	100	ALBUS	1	30
q_x	1.00E+05	2000	Rectangular	1	30
q_y	1.00E+06	800	ALBUS	1	30
q_z	1.00E+06	1000	ALBUS	1	30

3. Applying Fuzzy Logic to CMAC training and testing

In the process of CMAC training, the input was scaled using the Sensor Scale values tabulated above. However, the capacitive values had much difference when two electrodes were close and far. This can be observed in Fig. 5.13, which plots the maximum normalized sensor readings for each recording data. The maximum value is about 24 times the minimum value in the Fig.

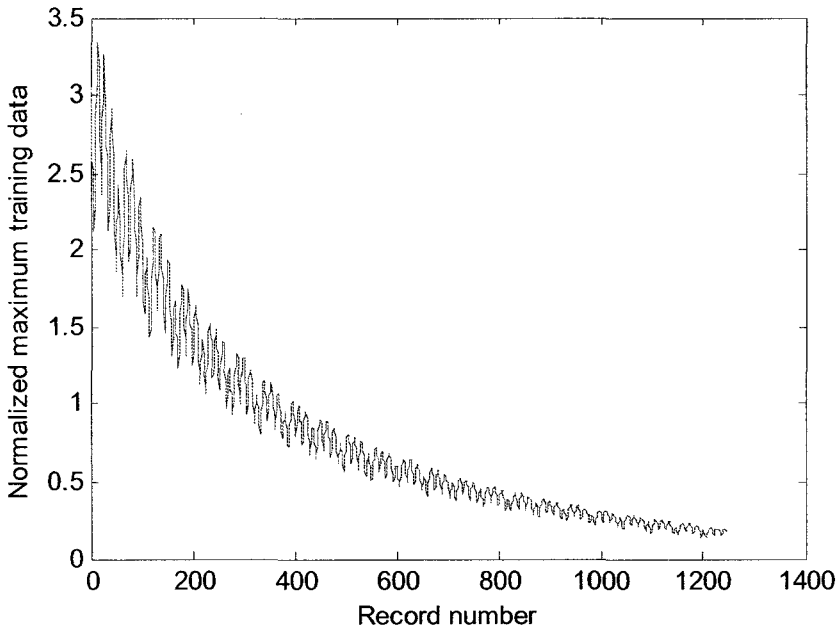

Fig. 5.13 Maximum normalized sensor readings for each epoch

A simple fuzzy logic procedure was created to improve the training and testing performance. The training process using fuzzy logic can be described as follows.

(1) The sensor scale values listed above were used to train the CMAC and got a set of weights termed "Weights A".

(2) Next, one quarter of the sensor scale values listed above were used to train the CMAC. This set of weights was termed "Weights B".

To improve the testing process, a simple Sugeno fuzzy model was used [36], as follows:

If maximum normalized sensor reading > 1.18 then

Use the scale/4 and Weights B

Else

Use the full scales and Weights A

End if

4. Dynamically changing the training parameters

A method was developed to dynamically change the weight normalization training gain, β_2, while the response error training gain, β_1, remained fixed. After the training of each epoch, the network response vector was obtained by inputting all the training data to the CMAC. The error vector was then calculated. Relative Error was used for the following procedure, which was trying to make the total errors zero through adjusting β_2:

If RME > 0.001

$\beta_2 = \beta_2 + 1$

Else if RME < -0.001

$\beta_2 = \beta_2 - 1$

End if

Fig.5.14 and Fig.5.15 are a comparison of fixed β_2 with the dynamically changed β_2. The figures show that the dynamically changed β_2 produces a much smaller MSE.

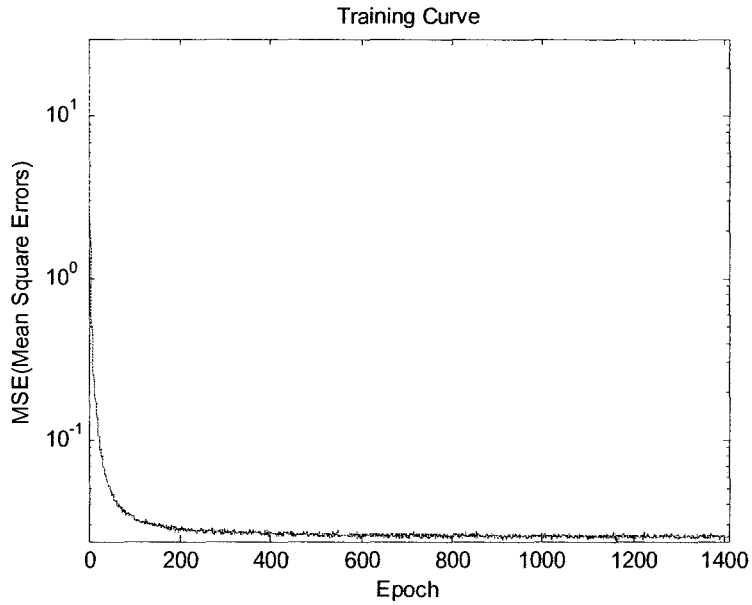

Fig. 5.14 Training curve using the fixed β_2

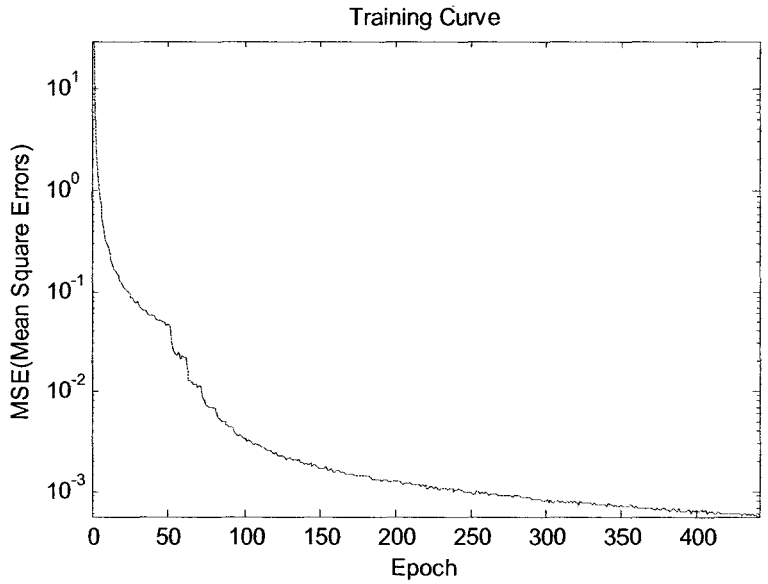

Fig. 5.15 Training curve using the dynamically changing β_2

5.6 Optimization of Number of Electrodes

In this application, fewer electrodes make the system simpler. A simple system has many advantages, such as less computation time, lower costs, and more reliability.

However, fewer electrodes will provide less information for the neural network, which then tends to produce larger errors.

For a well parameterized CMAC neural network, a convergent CMAC learning curve means the CMAC inputs contain enough information. On the other hand, a divergent curve reflects lack of information. In order to find out the minimum electrode combinations for the application, this research conducted two kinds of comparisons by observing the CMAC learning curve. First the thesis concluded the minimum electrode combinations using the reduced dataset after training CMAC neural network for eleven combinations.

Compared with the sampled dataset, the data varying range of the whole dataset is much larger. The research then used the whole training dataset and sampled dataset to train the CMAC neural network to further find out the suitable electrode combinations. These electrode combinations and the training/testing results are described in the sections that follow.

5.6.1 Electrode Combinations

The right side of Fig.5.16 shows how the electrodes were arranged on the moving robotic arm and stationary robotic arm (simulated by a wooden box as before) on the right side. On the left side is the concise representation with the name on the bottom, which will be used in the remaining drawings. All of the combinations are shown in Fig.5.17 and Fig.5.18.

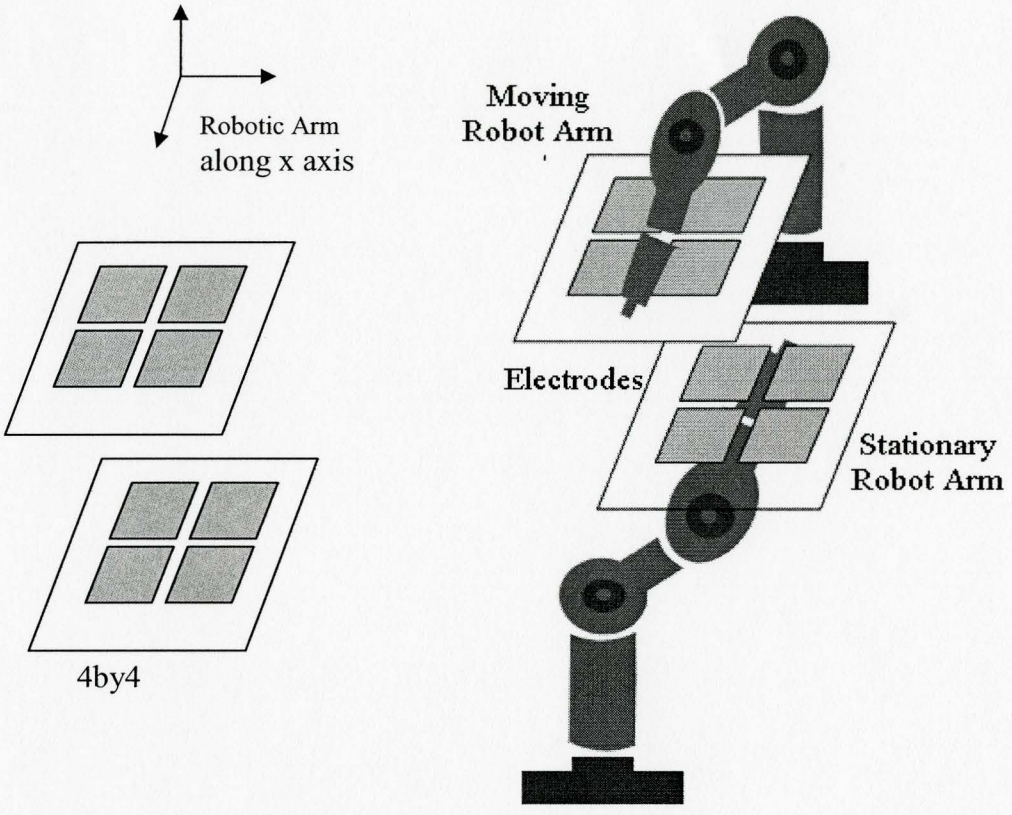

Fig. 5.16 The combination representation

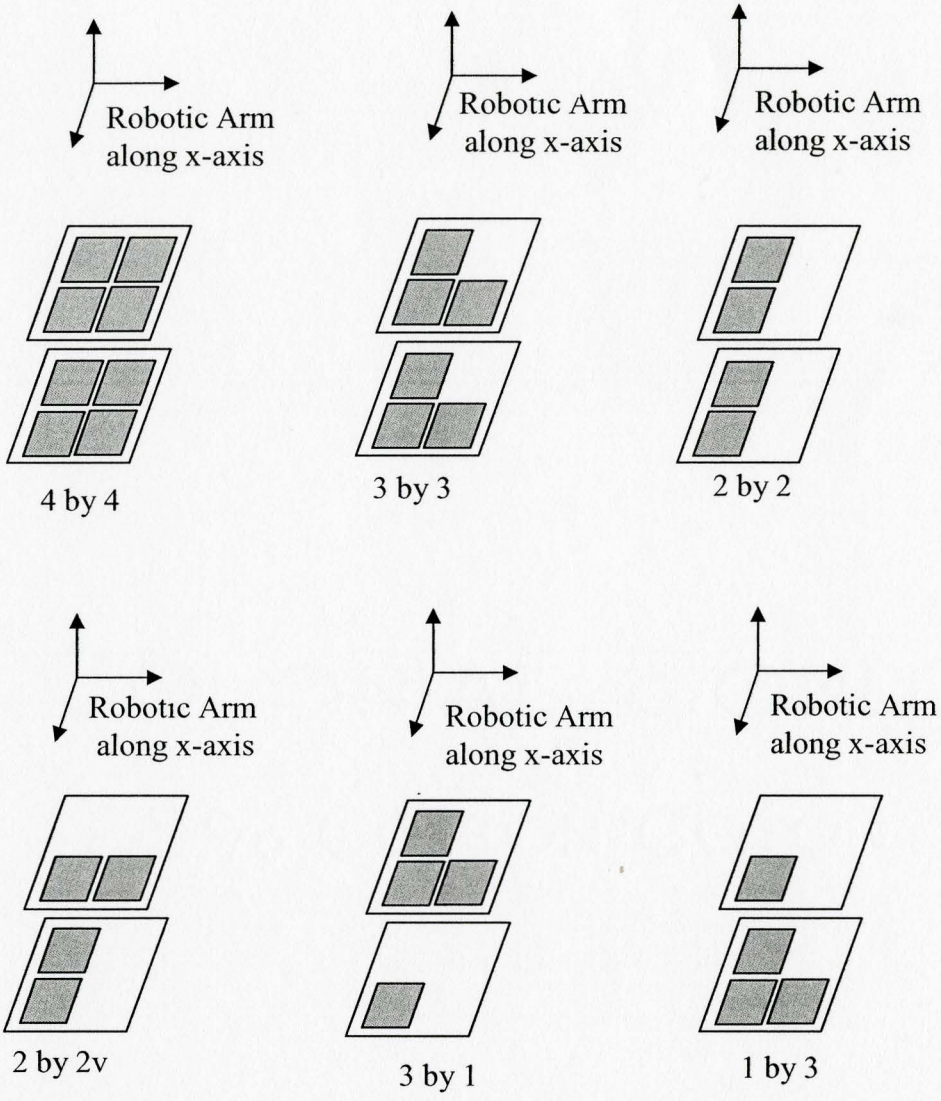

Fig. 5.17 Six of the eleven combinations used for optimizing the number of electrodes

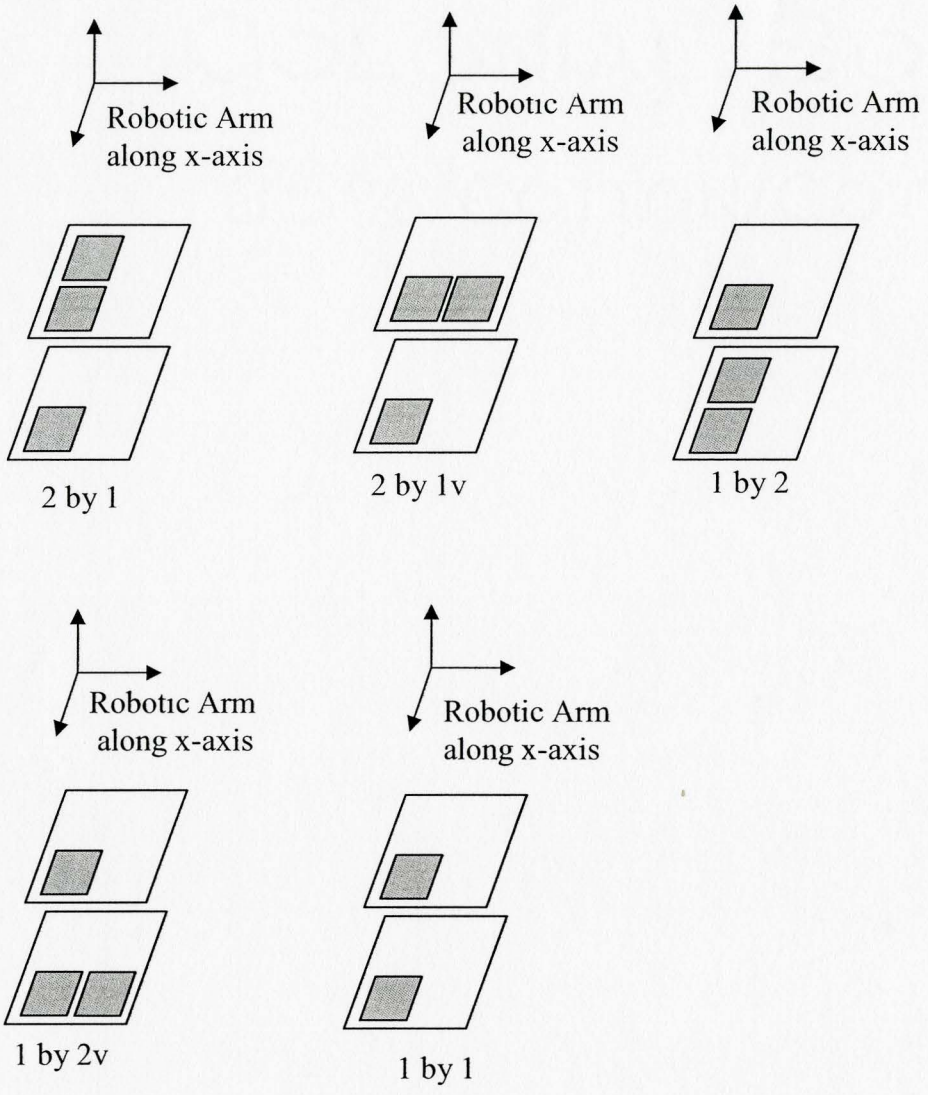

Fig. 5.18 The other five of the eleven combinations

5.6.2 Comparison of Electrode Combinations Results for Reduced Dataset

The learning curves were different for different robotic link pose variables. However, comparing the learning curves for these variables, there were two groups: curves that converged for all the electrode combinations, and curves that converged for total numbers

of electrodes larger than 4. Moreover, the arrangement of the electrode combination was also found to affect the convergence.

Pose variables x, z, and q_0 converged for all combinations. They all had a good learning curve. A typical one is shown in Fig.5.19. It seems that 1 by 1 combination can be used to get these pose variables, although the error will be relatively large. The training curve for the remaining variables, like the typical ones in Fig. 5.20, did not converge if the electrode numbers on each arm were fewer than two, even with 2 by 2 combination, when the capacitance readings did not have enough information. The arrangement of the electrodes affected the convergence as well. Fig. 5.20 also shows that although the 2 by 2 and 2 by 2v combinations had the same amount of electrodes, the learning curve of 2 by 2v converged while 2 by 2 was oscillating because of the different electrodes arrangement. The entire training results showed that the 2 by 2v combination had less MSE than 2 by 2. From this point of view, it is concluded that at least 2 by 2v is needed to get the information for seven link pose variables.

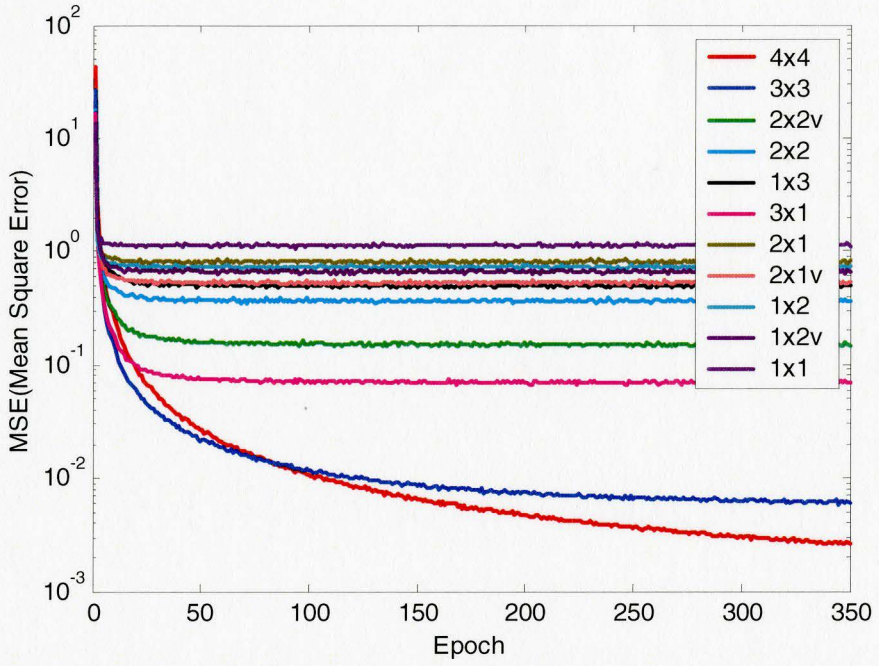

Fig. 5.19 Typical training curves for the group of pose variables that converged for all the combinations

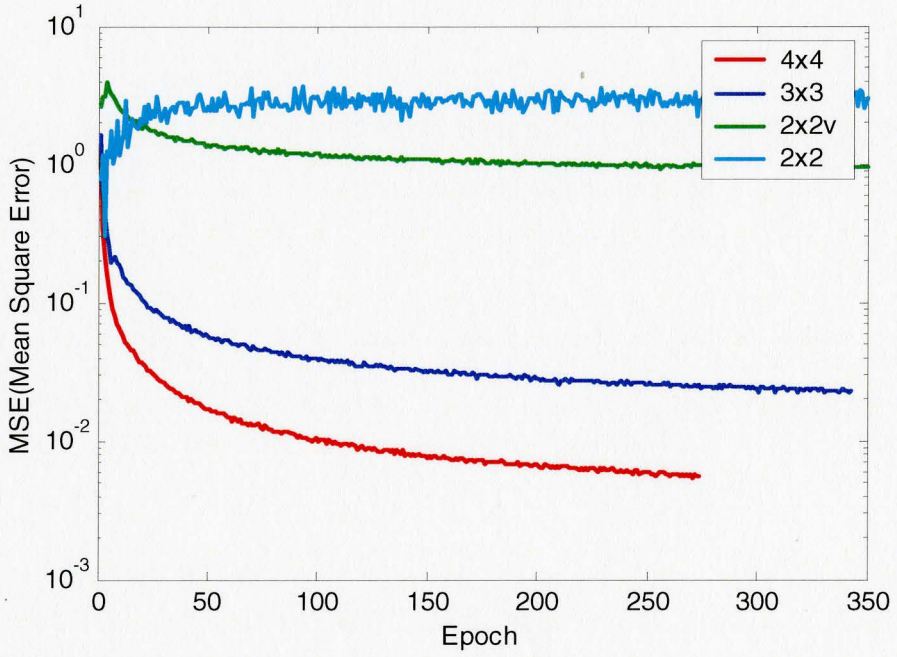

Fig. 5.20 Typical training curves for the group of pose variables that converged only for the electrode numbers on each arm larger than two

5.6.3 Comparison of Electrode Combinations Results for Whole Dataset and Sampled Dataset

The whole dataset and sampled dataset were used for training the three combinations, 4 by 4, 3 by 3, and 2 by 2v. These combinations were chosen because they produced good results with the reduced dataset. Fig.5.21 is a typical training curve for one of seven variables. Combinations 4 by 4 and 3 by 3 converged for both training datasets. Combination 2 by 2v did not converge. In the Fig., the green line is the training curve for 2 by 2v combination trained using the whole dataset. The pink line is the training curve using the sampled dataset. Both green and pink oscillated and did not converge. The data for the 2 by 2v combination will not be reported further.

The sampled dataset with less data can be used to save about two thirds of the training time. The training results were also helpful to further determine if the sampled dataset could represent the whole dataset. In Fig. 5.21, 4 by 4 and 3 by 3 are the learning curves using whole dataset while 4 by 4s and 3 by 3s are the learning curves using the sampled dataset. The Fig. shows that the training error differences between the sampled dataset and the whole dataset were small. Therefore, the sampled dataset can be used to represent the whole dataset. The Fig. also shows that the training errors for 4 by 4 are smaller than the training errors for the combination of 3 by 3.

Although 3 by 2 and 2 by 3 could be other pairs, to generalize the research to more than two robotic arms, symmetric pairs were chosen for the project. Therefore, combinations 4 by 4 and 3 by 3 are suitable for this application. The 4 by 4 combination had less MSE error than 3 by 3. Based on the above observations and conclusions, the

remainder of the chapter focuses on the 3 by 3 and 4 by 4 combinations trained using the sampled dataset.

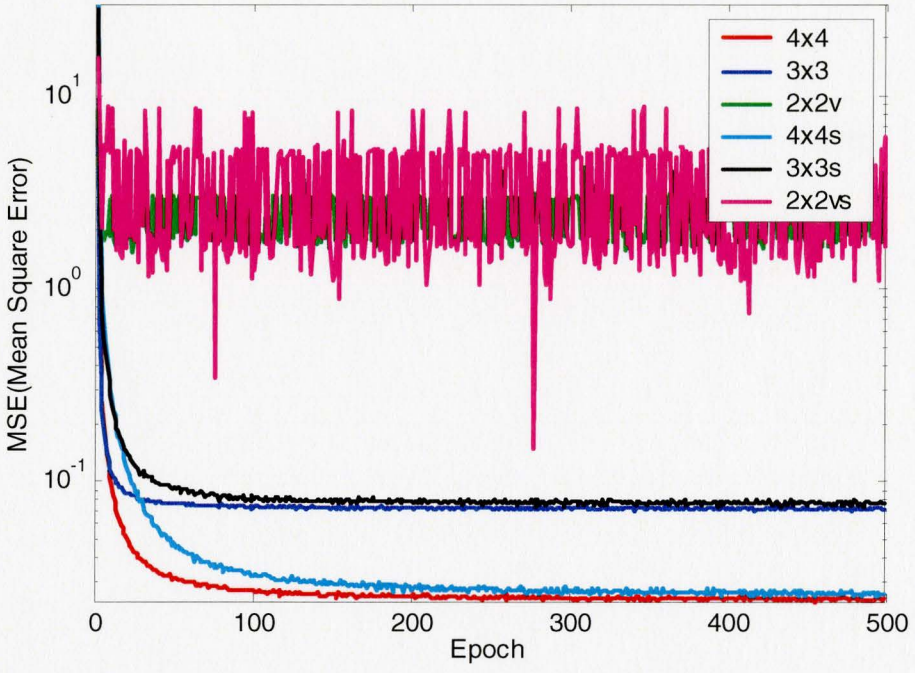

Fig. 5.21 Typical training curves for the whole dataset and sampled dataset

5.7 Detailed Comparison of Experimental Results for 3 by 3 and 4 by 4 Electrode Combinations

5.7.1 Introduction

The testing results for the 3 by 3 and 4 by 4 combinations trained using the sampled dataset will be compared using comparison plots, testing error plots, and error metrics.

The comparison plots presented in Appendix C directly show the robot pose variable superimposed onto the CMAC neural network response. The testing errors plots show the

error distribution for the testing dataset, and error metrics show the maximum, minimum and relative absolute values of the error.

5.7.2 Comparison Plots

Comparison plots for 4 by 4 combination are presented in Appendix C for each variable. In each Fig., the solid line represents for the CMAC response while the dash - dot line represents the desired output. The figures show that the output of the neural network matches the desired response very well. These plots were not included in the main body since they do not showing the errors very clearly.

5.7.3 Testing Error Plots

To better observe the errors generated in the testing process, all testing errors for the seven pose variables for the 4 by 4 combination are plotted in Fig. 5.22 and Fig. 5.23.

In Fig. 5.22, the errors for x coordinate are mostly within the range of -0.2 *mm* to +0.2 *mm* with several big errors around ± 0.4 *mm*. The errors for y coordinate are mostly distributed within ± 1.2 *mm*. The largest errors occurred with the y-coordinate. They are in the range of around ± 2.1 *mm*. The errors are relatively evenly distributed. In the beginning and the end of the test, *z*-coordinate has several big errors while in the middle the errors are relatively small.

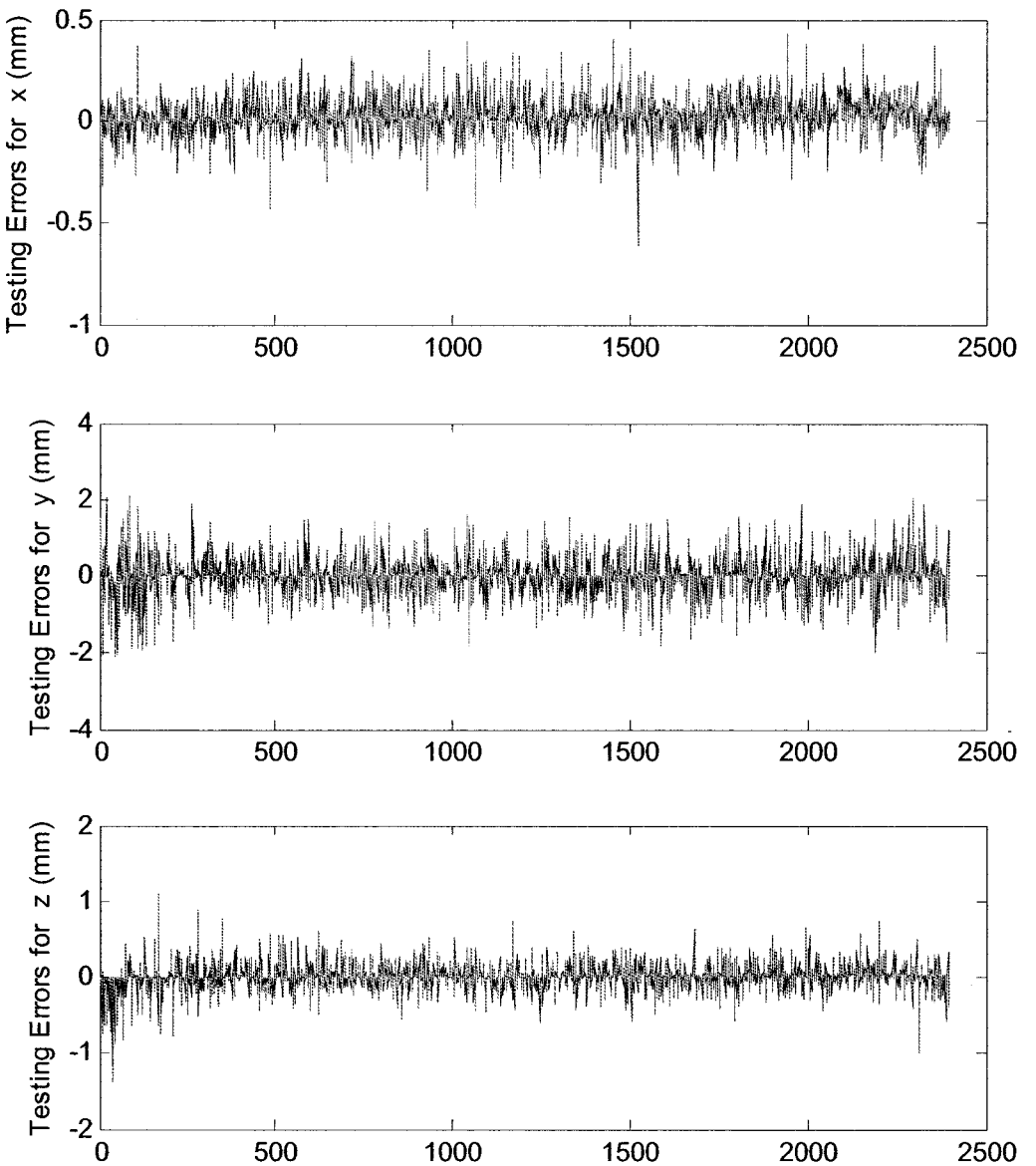

Fig. 5.22 Plots of x, y and z coordinates testing errors

Quaternions have smaller errors because their magnitude is much smaller than the coordinate variables. This is shown in Fig. 5.23. The errors for variable q_z are the smallest of the quaternions, which are in the 10^{-5} level, while the errors for variables q_0

and q_x are in the 10^{-4} level. Even in the worst case, the largest errors are the errors for variable q_y. They are generally in the 10^{-3} level with the largest errors in 10^{-2} level occurring at the beginning and the end of the test. The range of the errors for the quaternions is reasonable.

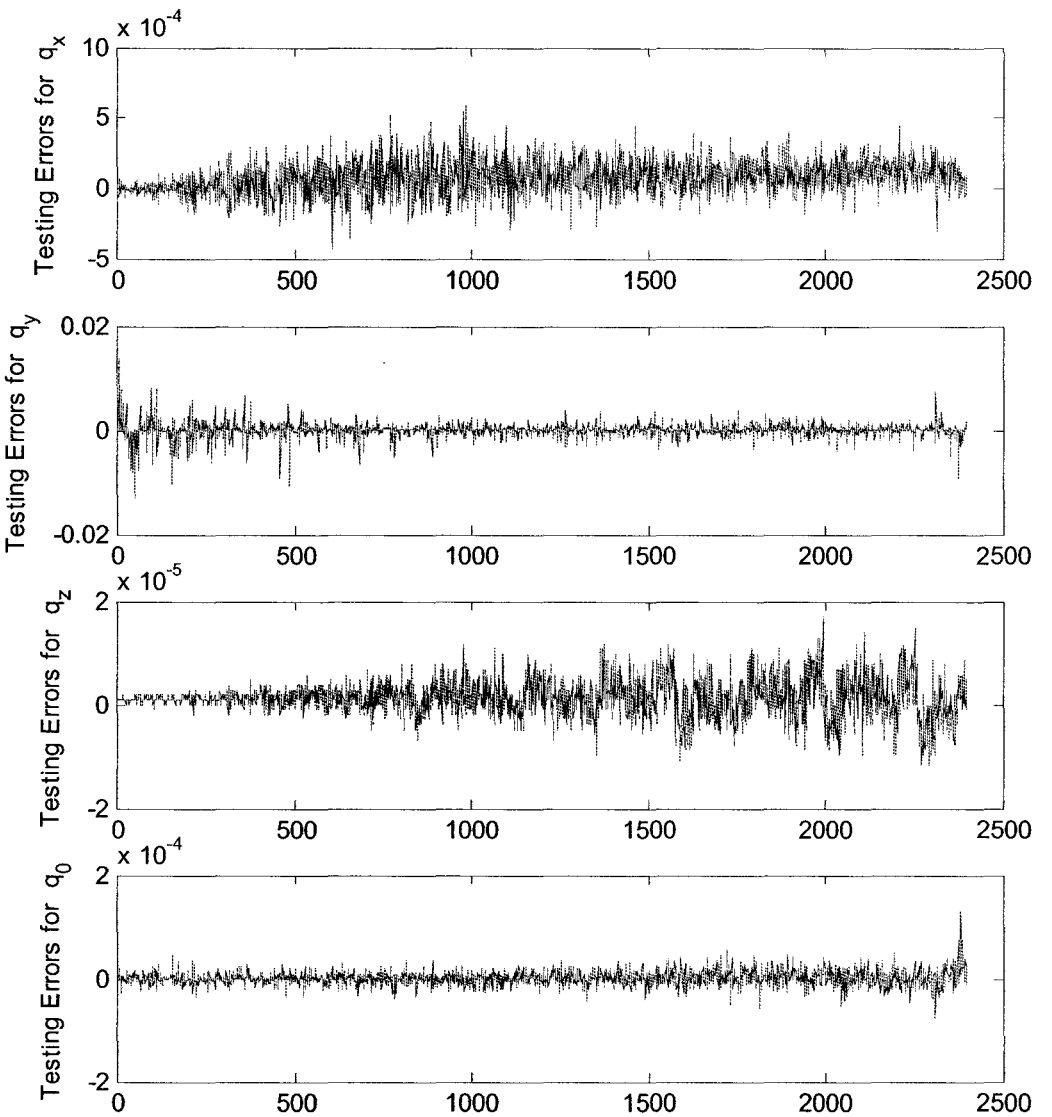

Fig. 5.23 Plots of quaternions, q_x, q_y, q_z, q_0 testing errors

The error is mainly related to the training parameters and varying range of the link pose variables. The training parameters have been refined. Because y-coordinate has the largest varying range, the errors are the biggest in entire variables.

5.7.4 Numerical Values of the Testing Error

The training errors metrics discussed in section 5.5 are used to calculate the numerical values of the testing errors. The varying range, MSE and relative errors are calculated after the training and new data testing using the sampled dataset.

i) Varying Range of Seven Link Pose Variables

In order to understand the experimental results, the maximum, minimum and varying range of seven robot link pose variables are shown in table 5.7. By examining the changing range, two variables are very obvious. *Y*-coordinate has the biggest changing range, which is about 152 *mm*. Variable q_z has the smallest changing range of about10^{-4} and q_x and q_0 have the changing range of about 10^{-3}.

Table 5.7 Ideal output data range of test value

	Maximum	Minimum	Changing Range
x	1.99E+01	-9.56E-02	2.00E+01
y	7.07E+01	-8.17E+01	1.52E+02
z	6.74E+01	1.67E+01	5.07E+01
q_x	1.00E+00	9.96E-01	4.39E-03
q_y	8.06E-02	-9.35E-02	1.74E-01
q_z	9.89E-03	9.17E-03	7.16E-04
q_0	9.12E-04	-7.87E-04	1.70E-03

* The unit for x, y, z coordinates is *mm*

ii) MSE

Table 5.8 shows the MSE of the testing data. Generally, the 4 by 4 combination is better than the 3 by 3 for each variable. The table shows the biggest MSE occurred with the *y*-coordinate while quaternions, q_0 and q_z, have the smallest MSE in both combinations. This occurred because variable *y* has the widest changing range; and variables, q_0 and q_z, have the narrowest changing range.

Table 5.8 MSE of the testing data

	4 by 4	3 by 3
x	7.82E-03	4.03E-01
y	2.21E-01	4.37E+00
z	2.91E-02	4.28E-01
q_x	1.88E-08	2.20E-04
q_y	1.82E-06	3.54E-06
q_z	1.43E-11	2.40E-09
q_0	1.88E-10	2.07E-09

iii) Relative Errors

Table 5.9 shows the maximum, minimum, mean and relative absolute testing errors for the testing data. The MAXAE is about 2.1 *mm* from y-coordinate for 4 by 4 while it is about 7.2 *mm* for 3 by 3 combination. Variables q_0 and q_z have the smallest errors around 10^{-4} and 10^{-5} level for both combinations. The MINAE is all zero for 4 by 4 combination, while only part of MINAE is 0 for 3 by 3. For MAE, coordinate y has the biggest error in both case, around 0.27 *mm* for 4 by 4 combination and 1.48 *mm* for 3 by 3 combinations. Variable q_0 and q_z still have smallest errors with 10^{-6} level for 4 by 4 combination and 10^{-5} for 3 by 3 combination. Generally, for the first three metrics, the coordinates have more errors than the quaternions. The RAE values of six variables are very close except

q_x that has the smallest values. Variable q_0 and q_y have the biggest errors of 14.1% and 14.7%, respectively, for 4 by 4 combination and 19.6% and 14%, respectively, for 3 by 3 combinations. Especially, compared to the other metrics, variable y has the lower RAE errors. Its RAE are about 2.56% and 8.81% for 4 by 4 and 3 by 3 combinations respectively.

Table 5.9 Maximum, minimum, mean and relative absolute testing error

	MAXAE*		MINAE*		MAE*		RAE (%)	
	4 by 4	3 by3	4 by 4	3 by 3	4 by 4	3 by 3	4by4	3by3
x	6.24E-01	2.49E+00	0	1.00E-03	5.75E-02	4.92E-01	3.13E+00	1.25E+01
y	2.09E+00	7.20E+00	0	2.99E-04	2.67E-01	1.48E+00	2.56E+00	8.81E+00
z	1.39E+00	4.41E+00	0	6.00E-04	9.63E-02	4.56E-01	2.06E+00	6.55E+00
q_x	5.80E-04	1.92E-01	0	0	1.08E-04	4.37E-03	5.80E-02	1.92E-01
q_y	1.38E-02	1.31E-02	0	0	6.27E-04	1.09E-03	1.47E+01	1.40E+01
q_z	1.70E-05	5.02E-04	0	0	2.83E-06	1.86E-05	1.72E-01	5.08E+00
q_0	1.29E-04	1.79E-04	0	0	9.66E-06	3.48E-05	1.41E+01	1.96E+01

* The unit is *mm* for *x, y, z* coordinates.

5.8 Conclusions

In the experiment, four aluminum foil electrodes were mounted on a CRS-F3 robot used as the moving robotic arm, another four were mounted on a wooden box used as the stationary robotic arm, and one pair of reference electrodes was mounted on the back of the CRS-F3. The capacitance sensing circuit and a multiplexor board were used to choose different electrode combinations.

The normalized capacitance, calculated using the sensed capacitive data divided by the reference pair's capacitive value, was found to be very effective at eliminating the environmental effects. It was subsequently used for the whole training and testing process.

Three datasets, named whole dataset, sampled dataset and reduced dataset were used for CMAC neural network training and testing. Because the sensed values changed broadly, applying fuzzy logic to CMAC training was a very good approach to get better results. Dynamically adjusting the weight normalization training gain was observed to further reduce the training errors.

To study the influence of electrode combinations on arm pose determination, the CMAC network training results for eleven different combinations were compared. It was concluded that 4 by 4 and 3 by 3 were the suitable combinations, where 3 by 3 had bigger errors than 4 by 4. The experimental results also showed that the sampled dataset can be used to train the neural network as accurately as, and faster than, the whole dataset. The sampled dataset was used for the rest of the training and testing.

The new data testing results were analyzed using comparison plots, error plots and error metrics. The errors for coordinates are larger than the quaternions variables because the varying range of the coordinates is much bigger than the quaternions. It is observed from the comparison plots and error plots that the network response matches very well with the desired output. The error analysis using metrics quantified this observation. The largest absolute testing error is 2.1 *mm* for 4 by 4 combination and 7.2 *mm* for 3 by 3 combination from y-coordinate, because y has the largest varying range. Variables q_0 and q_y, exhibit largest RAE with 14.1% and 14.7%, respectively, for 4 by 4 combination and 19.6% and 14%, respectively, for 3 by 3 combination. For x, y, and z coordinates, the RAE were less than 3.13% each for 4 by 4 combination.

CHAPTER 6

SENSOR - BASED ROBOT COLLISION AVOIDANCE

6.1 Introduction

The purpose of this chapter is to demonstrate the use of capacitance sensing in a simple collision avoidance system. The motion of the robot was restricted to vertical movement in the world *xz* plane. The hardware was the same as in Chapter 5 except that only two electrodes are used[3]. One electrode was mounted on link 4 of the CRS-F3 robot and the second was mounted on a box simulating a second robot. The CRS-F3 robot avoids colliding with the second robot by using the capacitance measured between the two electrodes. Note that for this simple setup, there is a nonlinear but one to one mapping between the distance and capacitance. The collision avoidance software and experimental results are described in the remainders of this chapter.

6.2 Collision Avoidance Software

Collision avoidance required software to be written for the PC and the CRS-F3 robot controller. The PC was programmed in C and the robot controller in RAPL-3 as before. These programs will be described in the following sections.

6.2.1 Collision avoidance Program Running on the PC

The program written for the PC compares the normalized capacitance value with a preset threshold to avoid a potential collision. When the detected capacitance is larger

[3] These two electrodes are in addition to the reference pair. The size of both electrodes was 153 mm x 205 mm.

than the threshold the PC program sends a stop signal to the GPIO port on the robot controller. The flowchart for this program is given in Fig. 6.1. In our implementation a threshold of 0.54 was used.

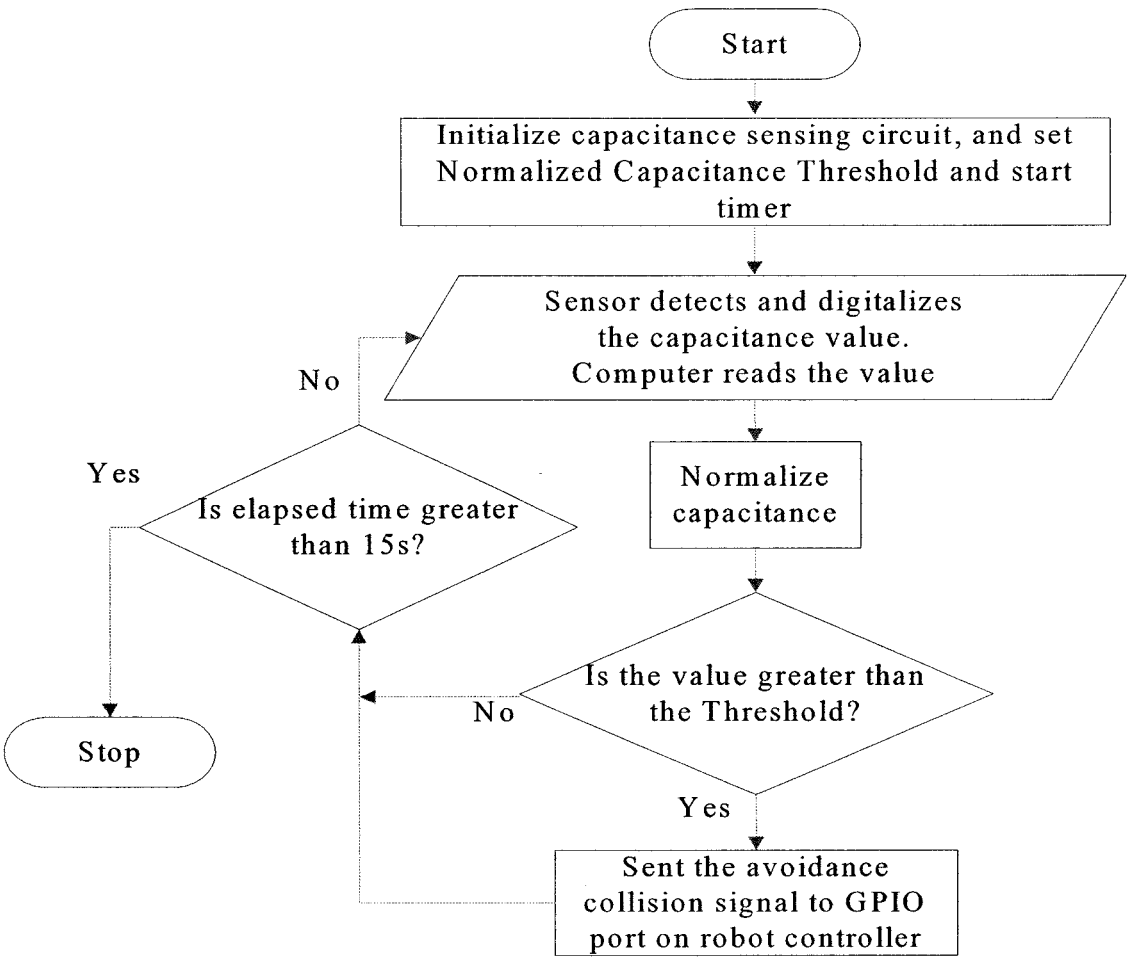

Fig. 6.1 Flowchart for collision avoidance program running on the PC

6.2.2 Collision avoidance Program Running on the Robot Controller

This program uses two preset locations in the xz plane named P_1 and P_2. Point P_2 was chosen to be inside the second robot to create a potential collision. The robot is first commanded to move from its current location to P_1. It is then commanded to move to P_2

unless halted by a collision avoidance command from the PC. A flowchart of the robot program is presented in Fig. 6.2. The 100 *ms* delay provides additional safety. Fig. 6.3 illustrates the experimental setup in the *xz* plane.

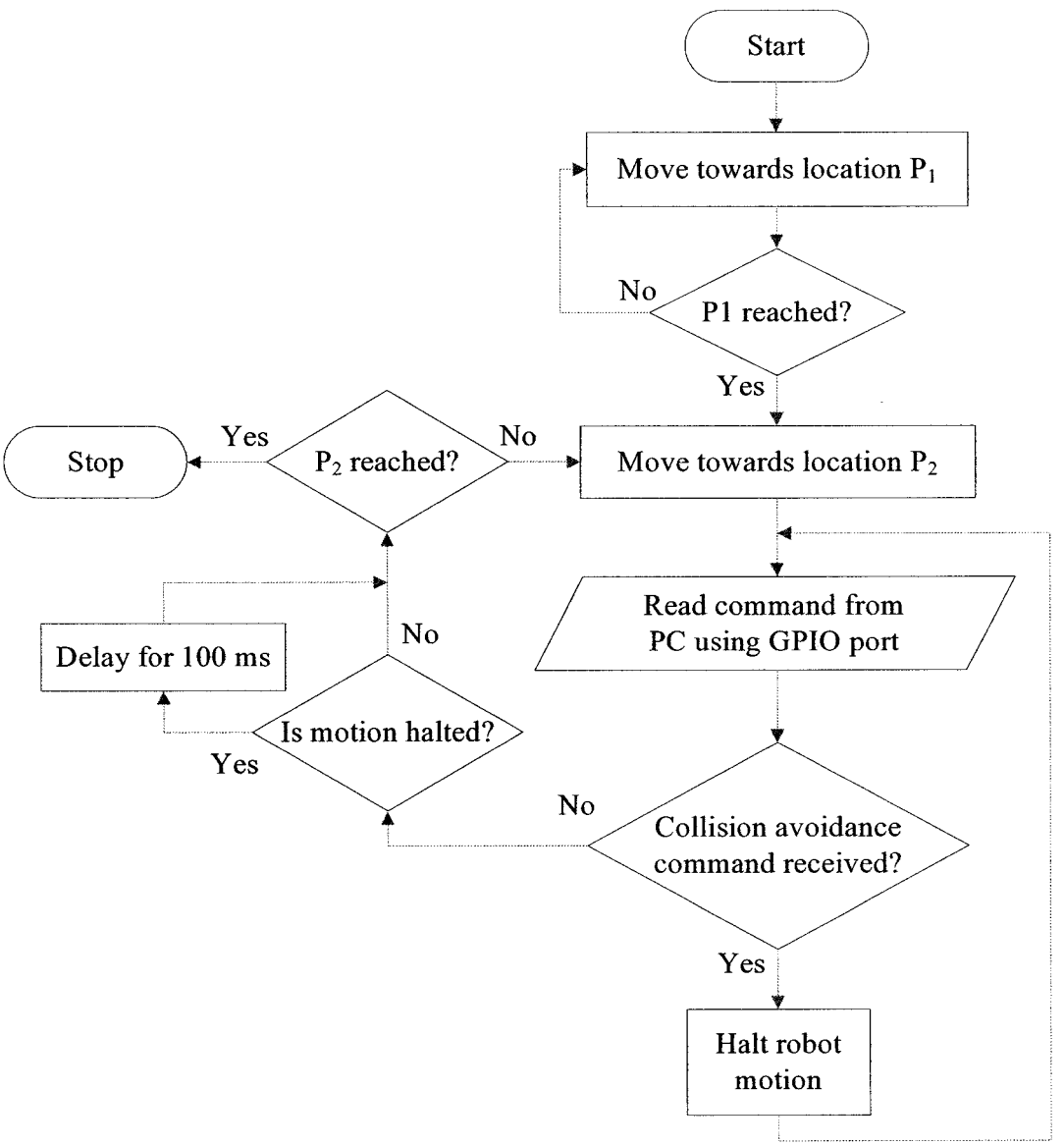

Fig. 6.2 Flowchart for collision avoidance program running on the robot controller

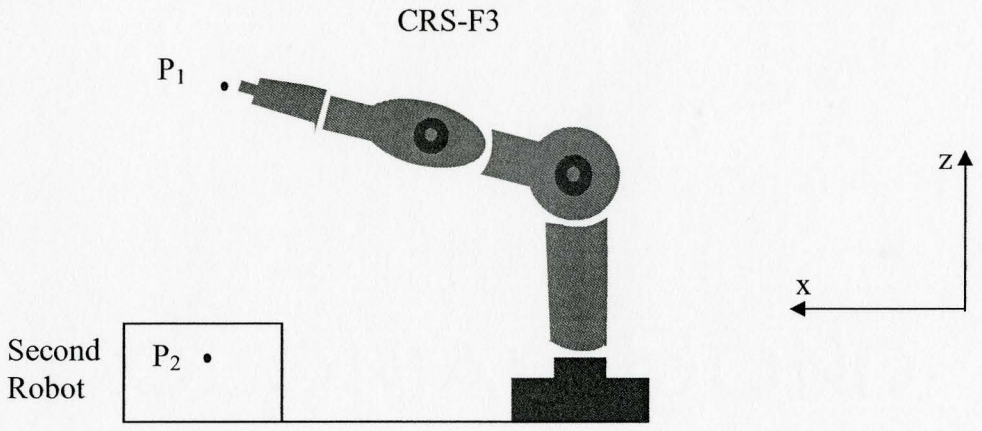

Fig. 6.3 Geometry of robots and preset locations

6.3 Experimental Results

Fig 6.4 shows still images taken from a video of a collision avoidance experiment. At t = 0 *s*: The CRS robot was returning to point P_1. At this time, the normalized capacitance was less than the threshold. The robot kept moving towards point P_1. At $t = 3$ *s*, the robot reached the original point P_1, and the robot was ready to move to point P_2. The normalized capacitance was much less than the threshold. The robot continued to work. At $t = 7.98$ *s*, the normalized capacitance was larger than the threshold, the stop signal was sent to the robot controller. After a short delay, the robot stopped. At $t = 8.3$ *s*, the movement of the second robot was simulated by removing the wooden box. The normalized capacitance reduced rapidly. After a 100 *ms* delay and verifying that the capacitance was less than the threshold, the robot resumed its motion. At $t = 14$ *s*: The robot continued to move and has reached point P_2. The potential collision was avoided.

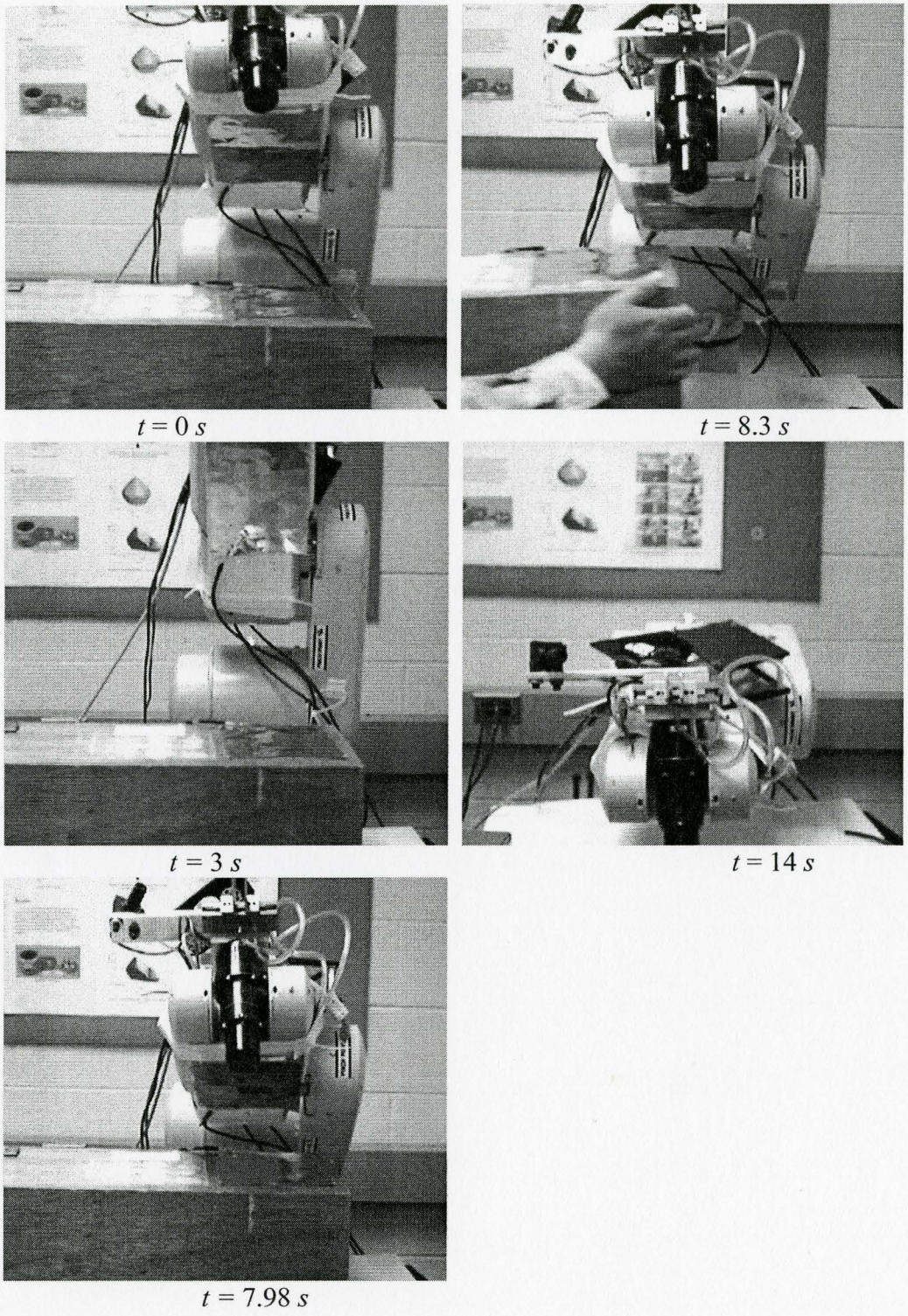

Fig. 6.4 Still images of a collision avoidance experiment

Corresponding to Fig. 6.4, Fig. 6.5 is the plotting of normalized capacitance versus time. The threshold value used was 0.54. The Fig. clearly shows the normalized capacitance was changing with the movement of the robotic arm. When time was at 3 *s*, the normalized capacitance reached its minimum value, which was the moment when robot reached point P_1. At 7.91 *s*, the detected capacitance was larger than the threshold value, and the stop signal was sent to the robot. And the robot stopped at 7.98 *s*. After about 0.3 *s*, the potential collision was manually removed. Because the second arm was far, the normalized capacitance dropped to 0.28. At this time, the robot continued its motion. At 14 *s*, the robot reached P_2.

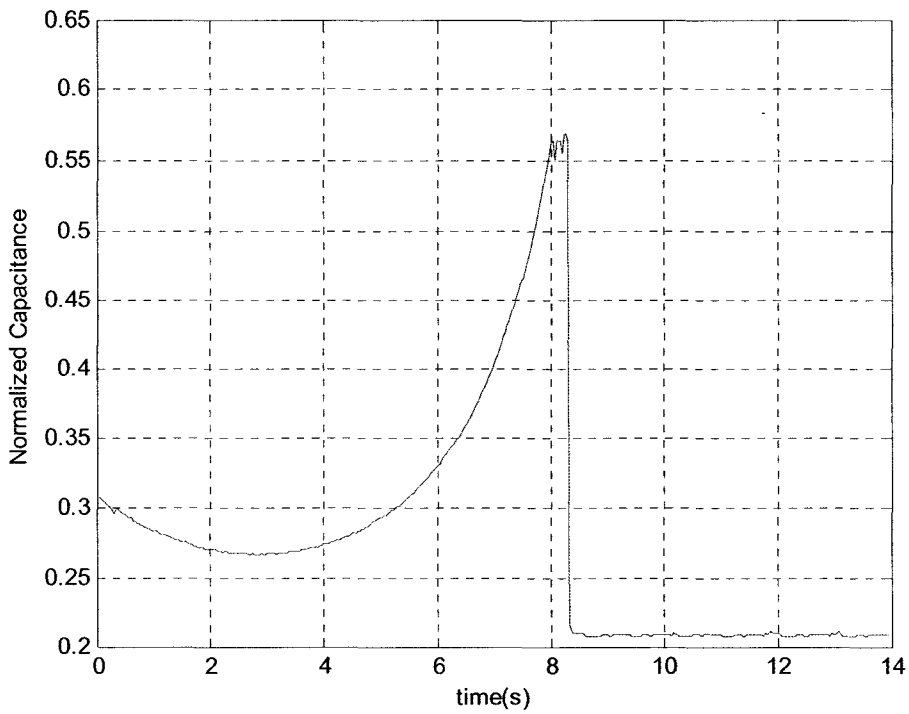

Fig. 6.5 Illustration of capacitance changing with the robotic arm movement

The detailed plot at the moment of normalized capacitance passed the threshold value is plotted in Fig. 6.6. The stopping signal was sent at 7.91 *s* with the normalized capacitance value of 0.54. After 70 *ms*, the robot stopped and the normalized capacitance was about 0.555. The sensor continued to detect. Because of the dynamic variations of the robot, the air flow and the electrical noise, the normalized capacitance was not in fixed value.

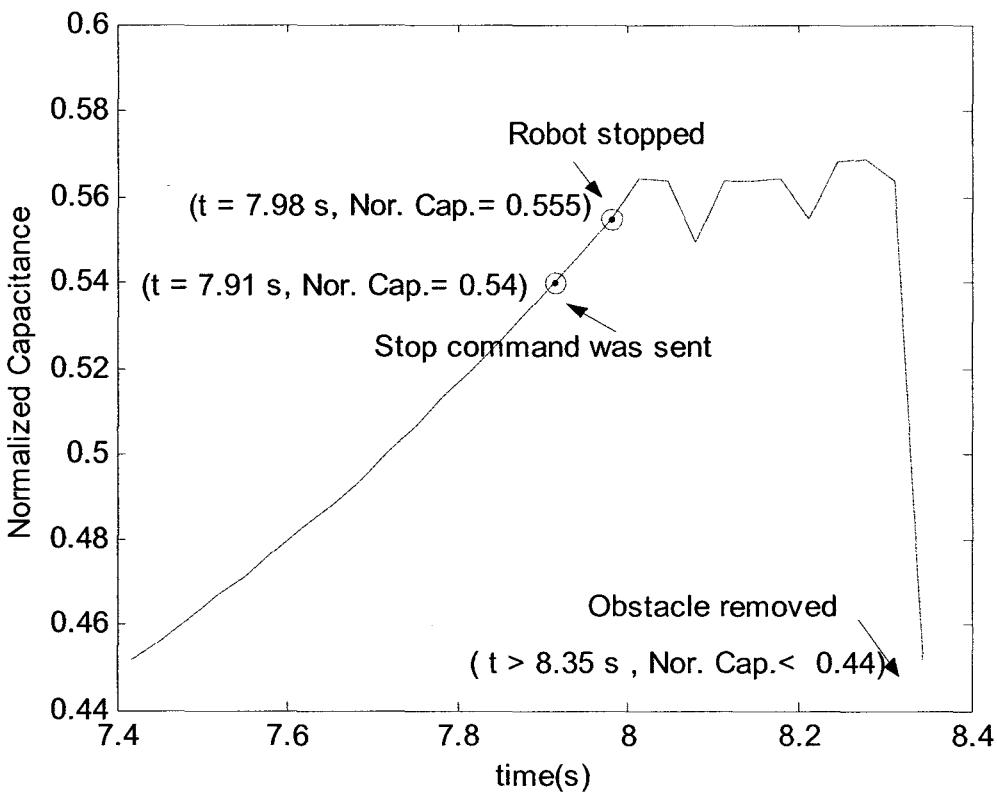

Fig. 6.6 Detail of capacitance changing with the robotic arm movement

6.4 Conclusions

A simple collision avoidance system was implemented using capacitance sensing. The capacitance was sensed using one pair of sensing electrodes, one on the moving

robotic arm, which was a CRS-F3, and another on a stationary robotic arm which simulated using a wooden box. By comparing the normalized capacitance from the sensor with the preset threshold, the PC program decided if there would be a potential collision with an obstacle in the robot's path. During the collision avoidance experiment, when the system detected a potential collision, the PC sent a stop command to the robot through the parallel port. The robot stopped in 70 *ms* and the sensor system continued to detect. When the sensor system determined that the obstacle was removed, i.e. the normalized capacitance was smaller than the threshold; the robot was commanded to continue its motion, after a 100 *ms* delay. The system successfully detected and avoided the potential collision.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Summary

In this research, capacitance sensing for robotic arm collision avoidance was investigated. After studying forward capacitance models, an inverse capacitance model was implemented using CMAC neural networks. The inputs of the inverse model are the capacitive sensor readings and the response is the robotic link pose represented by seven variables, three coordinates, (x, y, z) and four quaternions, (q_x, q_y, q_z, q_0). The inverse modeling was validated by comparing the responses of the trained CMAC neural networks with the ideal outputs calculated from the actual robot joint angles. Over a 15 *cm* range, pose variable y had a maximum absolute errors, 2.1 *mm*, for the 4 by 4 electrode combination and, 7.2 *mm*, for the 3 by 3 electrode combination. For the 4 by 4 combination the maximum relative errors were less than 3% for the x, y, and z variables, and less than 15% for the quaternion variables. For the 3 by 3 combination these values increased to 13% and 20%, respectively. The larger relative errors for the quaternion variables were due to their smaller ranges of variation.

7.2 Achievements

The main achievements of this thesis are summarized below:

1) Studied and compared the conventional method, MoM and FEM approaches for forward capacitance modeling. The MoM demonstrated that the fringing electric

field ignored by the conventional model is significant for the robotic arm application due to the relatively large ratio of electrode gap to electrode area. Even the state of the art FEM approach was shown to be unable to predict experimental capacitance measurements.

2) Developed an inverse model for predicting the pose of a robotic link using capacitance sensing and seven CMAC neural networks.
3) Developed the electrical circuitry and computer software for capacitance sensing with multiple combinations of electrodes.
4) Demonstrated the effectiveness of using a reference electrode pair to compensate for the effects of the environment on the capacitance measurements.
5) Demonstrated that dynamically changing the CMAC learning factors is an effective approach for reducing the training errors.
6) Created a new fuzzy logic approach that allowed the range of the CMAC input data to be increased without significantly increasing the training error.
7) Successfully implemented a simple collision avoidance system based on capacitance sensing.

7.3 Recommendations and Future Work

Many interesting research problems related to this thesis should be investigated, for example:

1) Test the performance of inverse capacitance modeling approach for the two rotational DOF of the pose that were kept constant in this thesis, and for larger ranges of joint motion than were used.

2) Test the inverse model under dynamic conditions (*i.e.* measure the capacitances while the robot is moving).
3) Develop a robotic collision avoidance system that utilizes the link pose estimated using the inverse capacitance modeling approach.
4) Investigate the best number, size and location of electrodes for measuring the pose of all the links of a robotic arm.

APPENDIX A ANSYS MACRO PROGRAMS FOR FORWARD CAPACITANCE MODELING

ANSYS macro programs for computing the capacitance between two cylindrical robot links at a given distance and angle.

A.1 Program for Section 3.4.3 (Non-Parallel Case)

```
/clear
/title, Arms to infinity capacitance using a Trefftz Domain
!
N=5                     !subdivision parameter
R1=108.18/2             !radius of the lower Arm (millimeters)
R11=123/2               !radius of the upper Arm
d1=R1+15
d11=R11+15
a=10                    !a is the angle between two arms
H11=330                 !Height of the upper Arm
H1=400                  !Height of the lower Arm
Zt=1.25*H1+d11+R11      !Choose the largest value
R2=3*Zt                 !half side of the cube
!
/PREP7                  !Enter Preprocessor
!
et,1,123                !10 node tetrahedral
!
emunit,epzro,8.854e-3   !free space permittivity (μMKSV units)
mp,perx,1,1             ! relative permittivity
!
CYL4,-d11,0,R11,0,R11,360,H11
!Creates a circular area or cylindrical volume anywhere on the working plane.
```

```
WPROTA,0,0,a
CYL4,d1,0,R1,0,R1,360,H1
WPROTA,0,0,-a
BLOCK,-R2,R2,-R2,R2,-R2,R2
vsbv,3,all
!
nummrg,all
!
mshape,1                !mesh with tets
mshkey,0                !free meshing
esize,,n                !mesh control
!
VMESH,ALL               !mesh volumes
cm,vol,volu             !group FE volumes into a component
BLOCK, -Zt, Zt, -Zt, Zt,-Zt, Zt
!BLOCK, X1, X2, Y1, Y2, Z1, Z2
cmsel,u,vol
cm,tvol,volu            !Trefftz volume component
!
nsel,s,loc,x,R2         !select outer nodes of FE domain
nsel,a,loc,x,-R2
nsel,a,loc,y,R2
nsel,a,loc,y,-R2
nsel,a,loc,z,R2
nsel,a,loc,z,-R2
sf,all,inf              !infinite surface flag on FE exterior
ASEL,S,AREA,,5,8        !select nodes at the conductor surface

nsla,s,1
cm,cond2,node
!Second Electrode surface node component for CMATRIX
```

```
ASEL,S,AREA,,1,4
nsla,s,1
cm,cond1,node          !Electrode surface node component for CMATRIX
csys,0
allsel
tzamesh,'tvol',,2      !Create Trefftz nodes
tzegen                 !Create Trefftz Domain
!
finish
!
/solu
antyp,static
eqslv,jcg              !select JCG solver
!
!Compute capacitance
!
cmatrix,1,'cond',2,1   !1,1 symmetry=1, no cond=1, ground at infinity
finish
```

A.2 Program for Section 3.4.2 (Parallel Cylinders Simulation)

```
/clear
/title, Parallel Cylinders to infinity capacitance using a Trefftz Domain
N=5                    !subdivision parameter
R1=100                 !radius of the cylinder (millimeters)
d=2*R1+350
R2=3*R1+d/2            !half side of the cube
R3=125
H=400
Rz=4*H+3*d
X1=2*R1+d/2
```

```
X2=1.25*X1
Y1=1.5*R1
Z1=0.25*H
Z2=1.5*H
/PREP7                  !Enter Preprocessor
et,1,123                !10 node tetrahedral
emunit,epzro,8.854e-3   !free space permittivity (mmMKSV units)
mp,perx,1,1             !relative permittivity
CYL4,d/2,0,R1,0,R1,360,H
CYL4,-d/2,0,R1,0,R1,360,H    !Creates a cylindrical volume on the working plane.
/view,1,3,3,3
/replot
BLOCK, -Rz, Rz, -Rz, Rz, -Rz, Rz
vsbv,3,all
nummrg,all
mshape,1                !mesh
mshkey,0                !free meshing
esize,,n                !mesh control
VMESH,ALL               !mesh volumes
cm,vol,volu             !group FE volumes into a component
BLOCK, -X2, X2, -Y1, Y1, -Z1, Z2
cmsel,u,vol
cm,tvol,volu            !Trefftz volume component
nsel,s,loc,x, Rz        !select outer nodes of FE domain
nsel,a,loc,x,-Rz
nsel,a,loc,y,Rz
nsel,a,loc,y,-Rz
nsel,a,loc,z,Rz
nsel,a,loc,z,-Rz
sf,all,inf              !infinite surface flag on FE exterior
ASEL,S,AREA,,1,4
```

```
nsla,s,1
cm,cond1,node          !Electrode surface node component for CMATRIX
ASEL,S,AREA,,5,8
nsla,s,1
cm,cond2,node          !Second Electrode surface node component for CMATRIX
Asel,u,AREA,,5,8
csys,0
allsel
tzamesh,'tvol',,2      !Create Trefftz nodes
tzegen                 !Create Trefftz Domain
finish
/solu
antyp,static
eqslv,jcg              !select JCG solver
cmatrix,1,'cond',2,1   !Compute capacitance
finish
```

APPENDIX B SOLUTION OF QUATERNION ROTATION VARIABLES

From given rotation matrix,

$$R=\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}=\begin{bmatrix} q_0^2+q_x^2-q_y^2-q_z^2 & 2q_xq_y-2q_0q_z & 2q_xq_z+2q_0q_y \\ 2q_xq_y+2q_0q_z & q_0^2-q_x^2+q_y^2-q_z^2 & 2q_yq_z-2q_0q_x \\ 2q_xq_z-2q_0q_y & 2q_yq_z+2q_0q_x & q_0^2-q_x^2-q_y^2+q_z^2 \end{bmatrix}$$

on the right hand side we have

$$tr(R)=6q_0-3+2(q_x^2+q_y^2+q_z^2)$$

$$|q|=1$$

$$q_0^2+q_x^2+q_y^2+q_z^2=|q|^2$$

$$tr(R)=6q_0-3+2(1-q_0^2)$$

$$tr(R)=4q_0^2-1 \qquad \text{(B.44)}$$

On the left side,

$$tr(R)=m_{11}+m_{22}+m_{33}$$

substituting into Eq.(4.28),

$$4q_0^2=tr(R)=m_{11}+m_{22}+m_{33}$$

$$4q_0q_x=m_{32}-m_{23}$$

$$4q_0q_y=m_{13}-m_{31}$$

$$4q_0q_z=m_{21}-m_{12}$$

The solution involves four cases as follows:

1. Case one: tr(R)+1>0

If $tr(R)+1 = m_{11}+m_{22}+m_{33}+1 > 0$, the solution is

$$q_0 = (1/2)\sqrt{m_{11}+m_{22}+m_{33}+1} \tag{B.45}$$

$$q_x = (m_{32}-m_{23})/(4q_0) \tag{B.46}$$

$$q_y = (m_{13}-m_{31})/(4q_0) \tag{B. 47}$$

$$q_z = (m_{21}-m_{12})/(4q_0) \tag{B.48}$$

2. Case two: $tr(R)+1 \le 0$, m_{11} is the greatest value

If $tr(R)+1 \le 0$ and $m_{11} > m_{22}$ and $m_{11} > m_{33}$,

The solution is,

$$1+m_{11}-m_{22}-m_{33}$$

$$=1+2q_0^2+2q_x^2-1-(2q_0^2+2q_y^2-1)-(2q_0^2+2q_z^2-1)$$

$$=4q_x^2$$

$$q_x = \frac{1}{2}\sqrt{1+m_{11}-m_{22}-m_{33}}$$

$$m_{32}-m_{23} = 4q_0q_x$$

Therefore,

$$q_0 = \frac{1}{4q_x}(m_{32}-m_{23})$$

The same way we have

$$m_{12}+m_{21} = 4q_xq_y$$

Or, $q_y = \frac{1}{q_x}(m_{12}+m_{21})$

$$m_{13} + m_{31} = 4q_x q_z$$

Or, $q_z = \frac{1}{4q_x}(m_{13} + m_{31})$

3. Case three: tr(R)+1 ≤ 0 , m_{22} is the greatest value

$1 + m_{22} - m_{11} - m_{33} = 1 + q_0^2 - q_x^2 + q_y^2 - q_z^2 - (q_0^2 + q_x^2 - q_y^2 - q_z^2) - (q_0^2 - q_x^2 - q_y^2 + q_z^2) = 4q_y^2$

or,

$$q_y = \frac{1}{2}\sqrt{1 + m_{22} - m_{11} - m_{33}} \tag{B.49}$$

Using the same procedure, the equation for this situation is

$$q_0 = \frac{1}{4q_y}(m_{13} - m_{31}) \tag{B.50}$$

$$q_x = \frac{1}{4q_y}(m_{12} + m_{21}) \tag{B.51}$$

$$q_z = \frac{1}{4q_y}(m_{23} + m_{32}) \tag{B.52}$$

4. Case four: tr(R)+1 ≤ 0 , m_{33} is the greatest value

$1 + m_{33} - m_{11} - m_{22} = 1 + (q_0^2 - q_x^2 - q_y^2 + q_z^2) - (q_0^2 + q_x^2 - q_y^2 - q_z^2) - q_0^2 - q_x^2 + q_y^2 - q_z^2 = 4q_z^2$

Or,

$$q_z = \frac{1}{2}\sqrt{1 + m_{33} - m_{11} - m_{22}} \tag{B.53}$$

Using the same procedure, the equation for this situation is

$$q_0 = \frac{1}{4q_y}(m_{21} - m_{12}) \tag{B.54}$$

$$q_x = \frac{1}{4q_z}(m_{13} + m_{31}) \tag{B.55}$$

$$q_y = \frac{1}{4q_z}(m_{23} + m_{32}) \tag{B.56}$$

APPENDIX C COMPARISON PLOTS

C.1 Testing results for *x*-coordinate

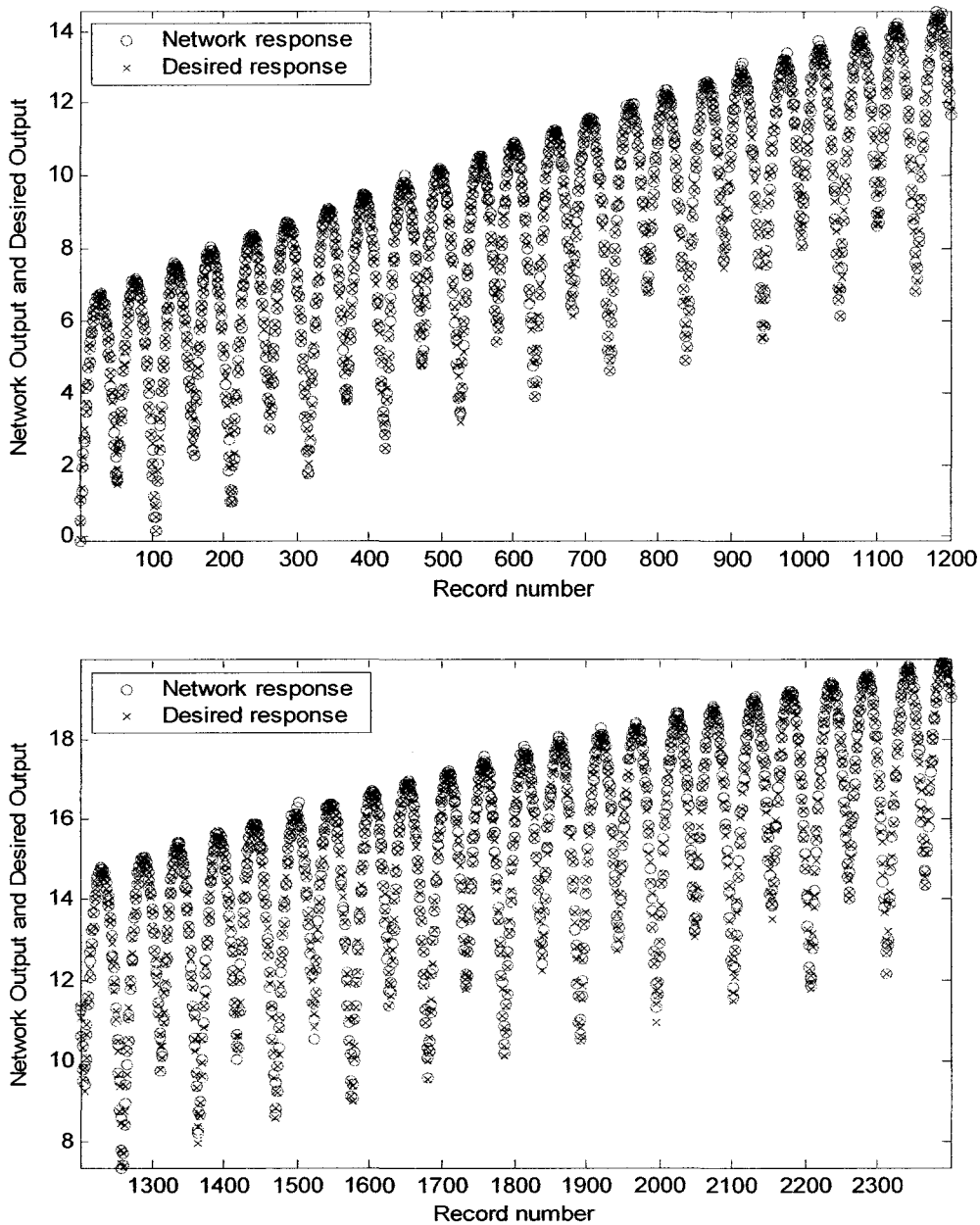

Fig. C.1 Testing results for *x*-coordinate

C.2 Testing results for *y*-coordinate

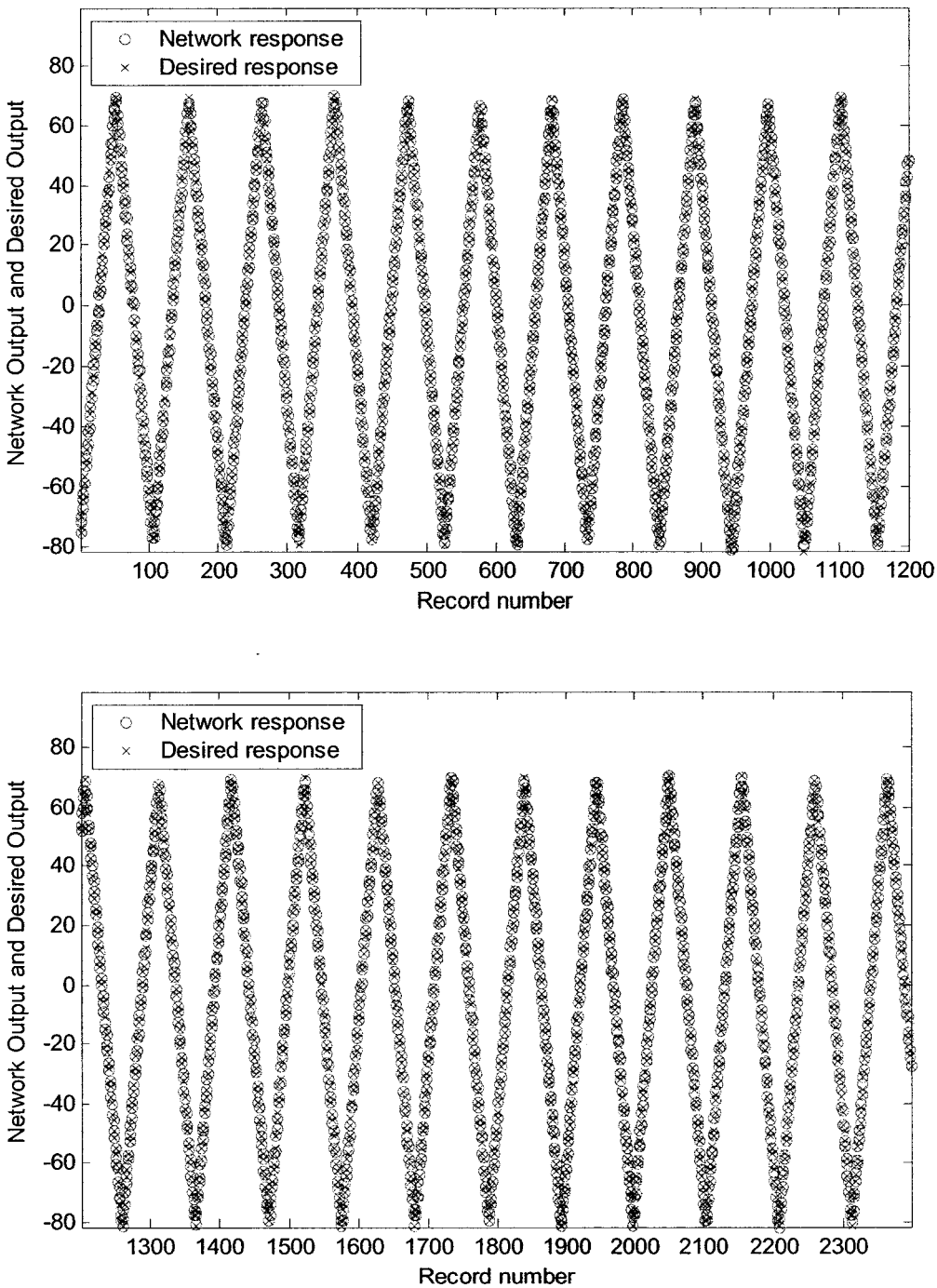

Fig. C.2 Testing results for *y*-coordinate

C.3 Testing results for z-coordinate

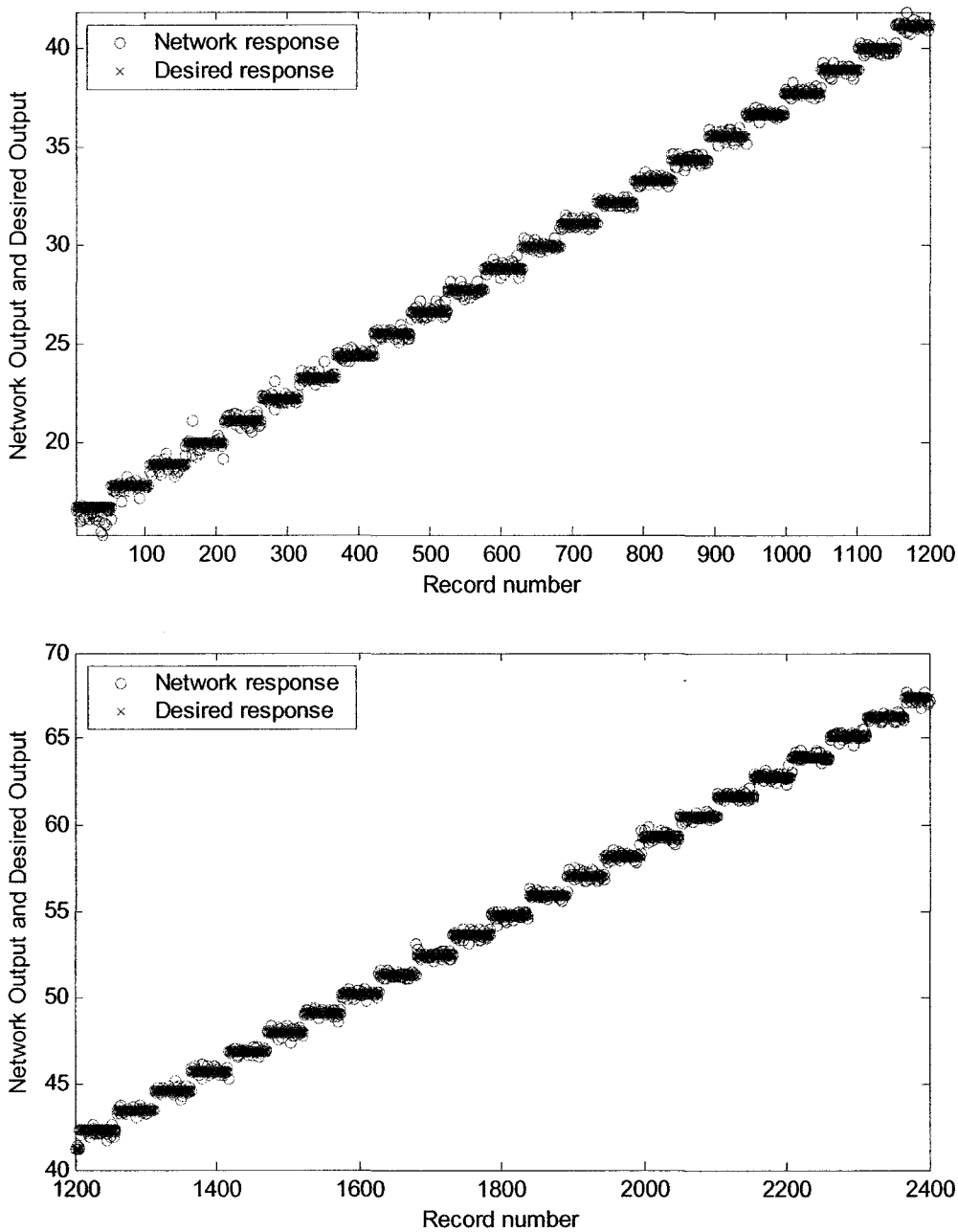

Fig. C.3 Testing results for z-coordinate

C.4 Testing results for quaternion, q_x

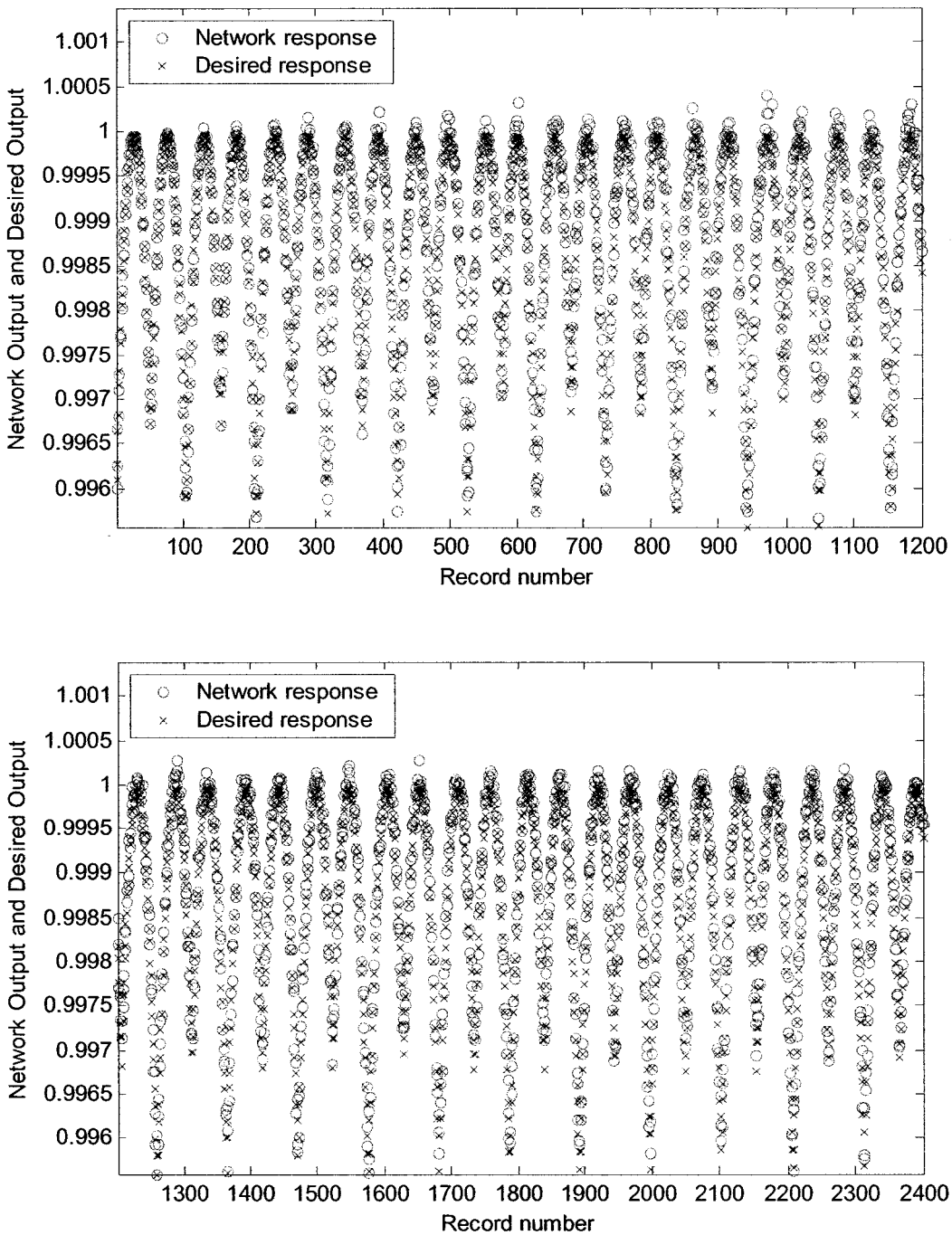

Fig. C.4 Testing results for q_x

C.5 Testing results for quaternion, q_y

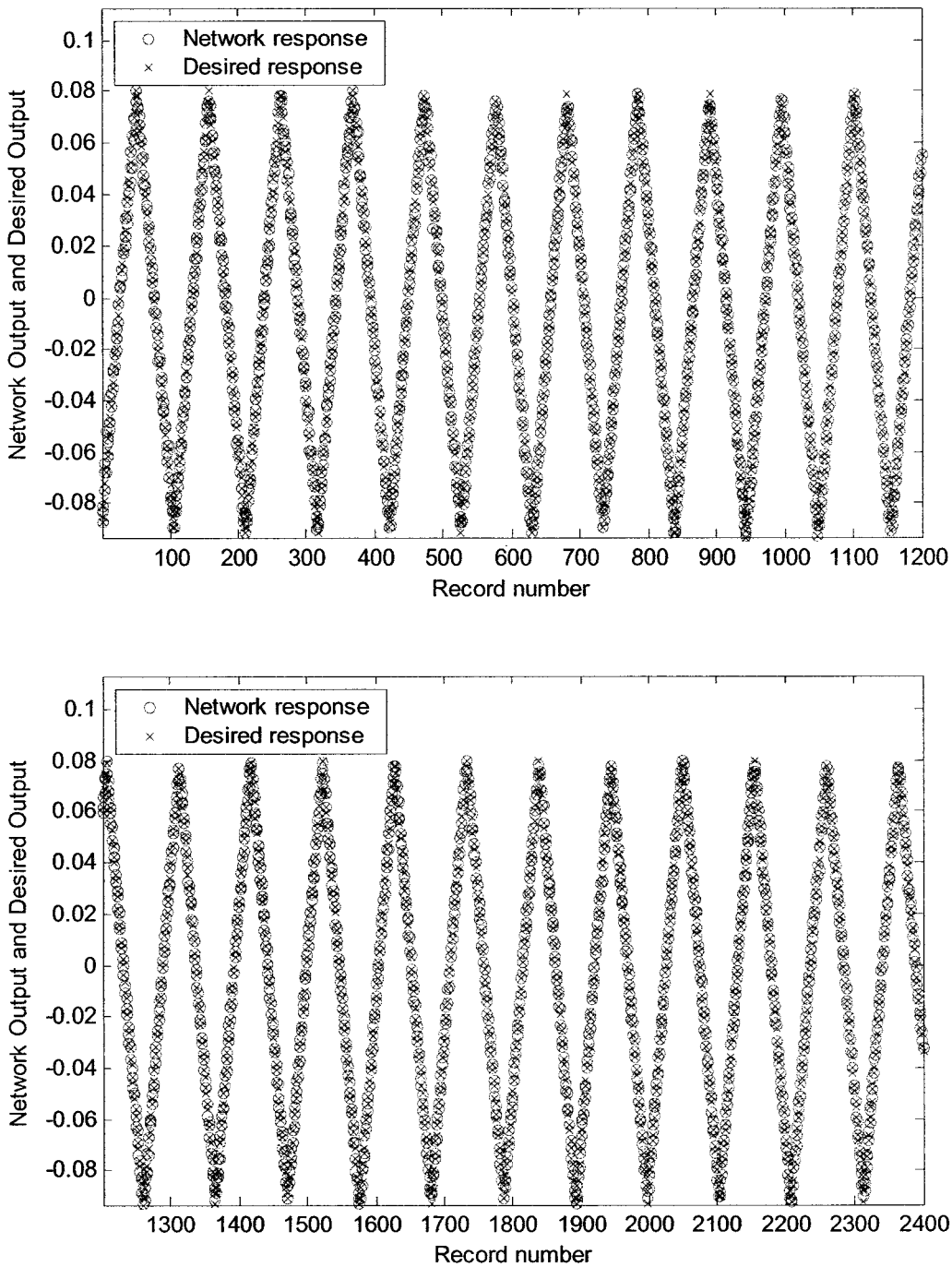

Fig. C.5 Testing results for q_y

C.6 Testing results for quaternion, q_z

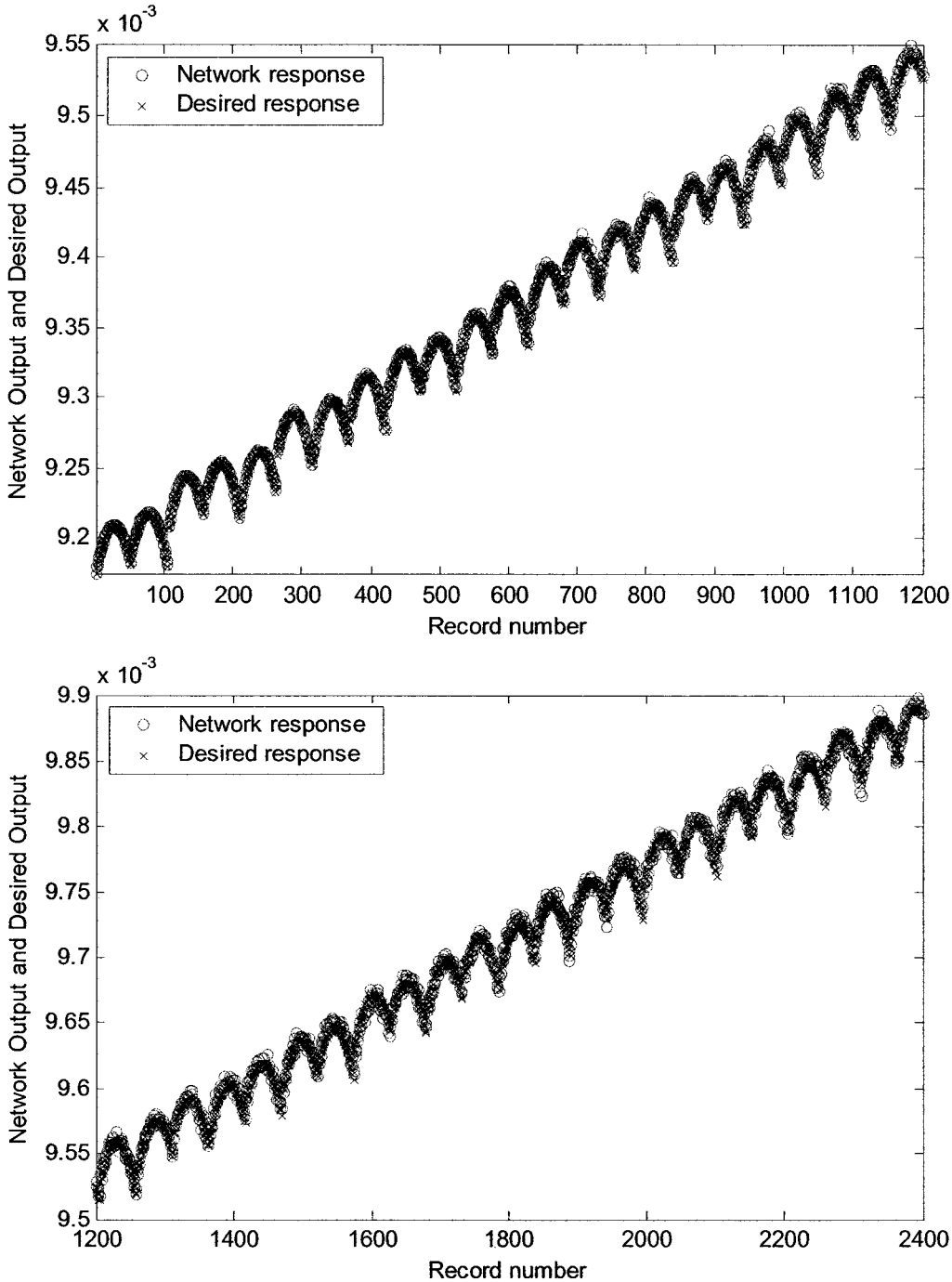

Fig. C.6 Testing results for q_z

C.7 Testing results for quaternion, q_0

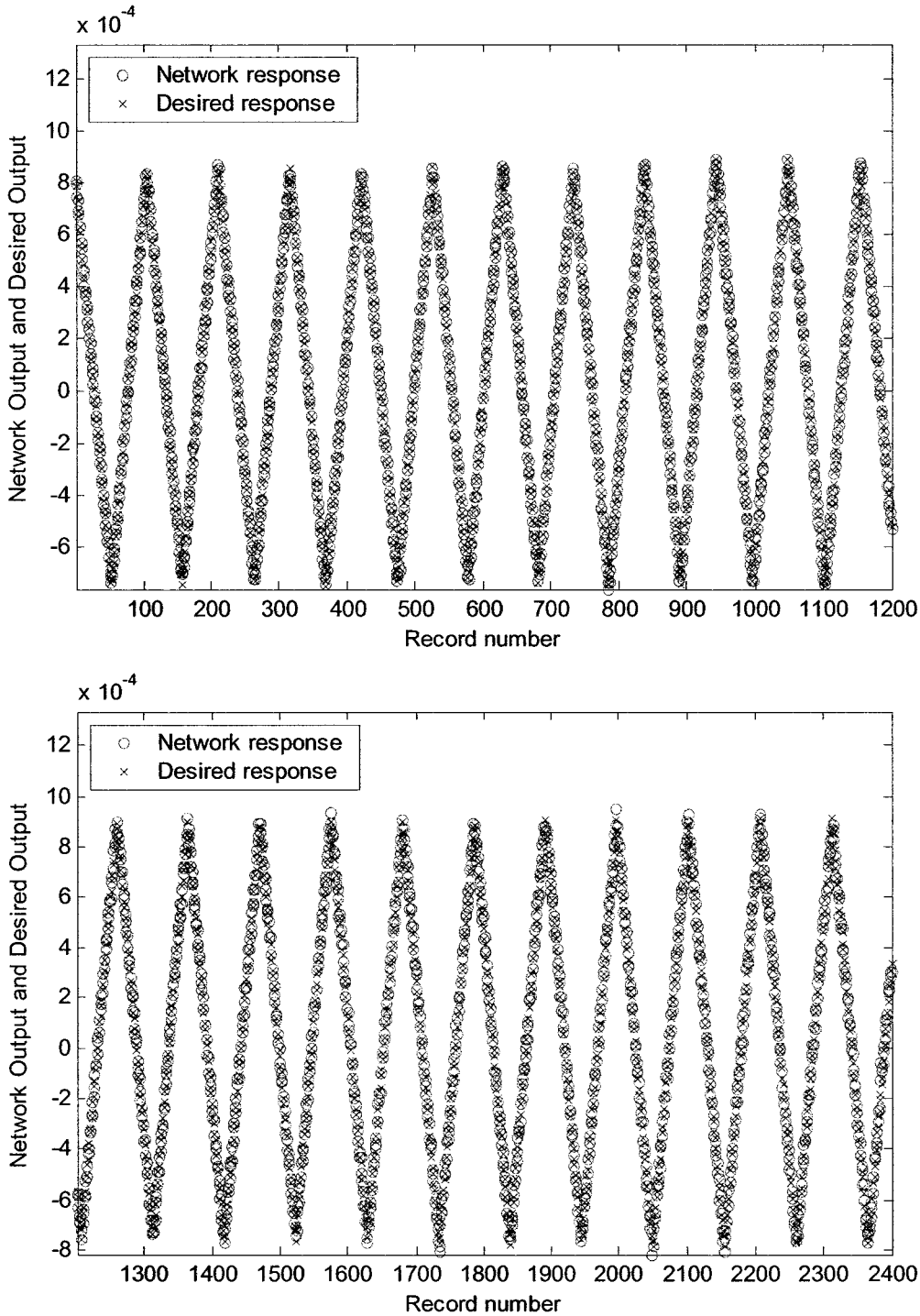

Fig. C.7 Testing Results for q_0

REFERENCES

[1] C. L. Boddy and J. D. Taylor, "Whole-arm reactive collision avoidance control of kinematically redundant manipulators," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1993, pp 382-387.

[2] E. Cheung and V. Lumelsky, "Motion planning for robot arm manipulators with proximity sensoring," in *IEEE International Conference on Robotics and Automation*, 1988, vol. 2, pp 740–745.

[3] E. Cheung and V. Lumelsky, "Motion planning for a whole-sensitive robot arm manipulator," in *IEEE International Conference on Robotics and Automation*, vol.1, May 1990, pp. 344 - 349.

[4] H. Seraji, R. Steele and R. Ivlev, "Sensor-Based Collision Avoidance: Theory and Experiments," *Journal of Robotic Systems*, vol. 13, n 9, pp 571-586. Sep, 1996.

[5] D. Gandhi and E. Cervera, "Sensor Covering of a Robot Arm for Collision Avoidance," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, 2003, pp 4951-4955.

[6] N. Blanc, T. Oggier, G. Gruener, J. Weingarten, A. Codourey, and P. Seitz, "Miniaturized smart cameras for 3D-imaging in real-time," in *Proceedings of the IEEE Sensors,* 2004, p 471-474.

[7] J. W. Weingarten, G. Gruener, and R. Siegwart, "A state-of-the-art 3D sensor for robot navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , 2004, p 2155-60.

[8] L. Marques, U. Nunes, and A.T. de Almeida, "A new 3D optical triangulation sensor for robotics," in *5th International Workshop on Advanced Motion Control Proceedings, AMC'98 - Coimbra.,* 1998, pp 512-517.

[9] D. Ebert, T. Komuro, A. Namiki, and M. Ishikawa, "Safe human-robot-coexistence: emergency-stop using a high-speed vision-chip," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 2923-2928.

[10] S. Morikawa, T. Senoo, A. Namiki, and M. Ishikawa, "Realtime collision avoidance using a robot manipulator with light-weight small high-speed vision systems," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 794-799.

[11] A. Tsalatsanis, K. Valavanis, and A. Yalcin, "Vision Based Target Tracking and Collision Avoidance for Mobile Robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 48, n 2, pp 285-304. Feb. 2007.

[12] M. Jagiella and S. Fericean, "Miniaturized Inductive Sensors for Industrial Applications," in *Proceedings of IEEE Sensors*, vol. 1, n 2, 2002, pp 771-778.

[13] M. Jagiella, S. Fericean, A. Dorneich, and M. Eggimann, "Theoretical progress and recent realizations of miniaturized inductive sensors for

automation," in *Proceedings of the Fourth IEEE Conference on Sensors*, 2005, pp 488-491.

[14] Pepperl+Fuchs,http://www.am.pepperl-fuchs.com/products/productfamily.jsp?division =FA&productfamily_id=3, (Accessed 25 June 2007)

[15] J. M. Vranish, R. L. McConnell, and S. Mahalingam, "Capaciflector collision avoidance sensors for robots," *Computers & Electrical Engineering*, vol. 17, pp 173-179, 1991.

[16] J. L. Novak. and J. T. Feddema, "A capacitance-based proximity sensor for whole arm obstacle avoidance," in *IEEE International Conference on Robotics and Automation Proceedings*, vol.2, 1992, pp. 1307–1314.

[17] J. T. Feddema and J. L. Novak, "Whole arm obstacle avoidance for teleoperated robots," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 3303-3309.

[18] R. Volpe and R. Ivlev, "A Survey and Experimental Evaluation of Proximity Sensors for Space Robotics," in *Proceedings-IEEE International Conference on Robotics and Automation*, 1994, pp 3466-3473.

[19] S. R. H Hoole, "Artificial neural networks in the solution of inverse electromagnetic field problems," *IEEE Transactions on Magnetics*, Vol. 29, n2, pp 1931-1934, Mar. 1993.

[20] Q. Marashdeh, W. Warsito, L.S. Fan, and F.L Teixeira, "Solution of non-linear forward problems in electrical capacitance tomography using neural

networks," in *IEEE Antennas and Propagation Society, AP-S International Symposium (Digest)*, vol. 1A, 2005.

[21] R. C. Booton, Jr., *Computational Methods for Electromagnetics and Microwaves*. John Wiley & Sons, 1992.

[22] R. F. Harrington, *Field Computation by Moment Methods.* New York: Macmillan, 1968.

[23] F. T. Ulaby, *Fundamentals of applied electromagnetics*. Media Edition, Prentice Hall, New Jersey, 2001.

[24] Release 10.0 Documentation for ANSYS.

[25] W. T. Miller, F.H. Glanz, and L. G. Kraft, "CMAC: an associative neural network alternative to backpropagation", in *Proceedings of the IEEE, Special Issue on Neural Networks*, vol. 78, pp.1561-1567, Oct. 1990.

[26] J. S. Albus, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME*, September 1975.

[27] R.Mukndan, "Quaternions: From classical mechanics to computer graphics, and beyond," in *Proceedings of the 7th Asia Technology conference in mathematics*, 2002.

[28] Jack B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality.* Princeton, N.J., Princeton University Press, c1999.

[29] C. Hu, Q.-H. Meng, M. Mandal, and X. P. Liu,, "Robot rotation decomposition using quaternions," in *IEEE International Conference on Mechatronics and Automation*, ICMNA, 2006.

[30] J.S. Albus, "A New Approach to Manipulator Control: The CereBellar Model Articulation Controller (CMAC)," *Transactions of the ASME*, September 1975.

[31] G. Horvath, "Kernel CMAC: an Efficient Neural Network for Classification and Regression, *Acta Polytechnica Hungarica*," Vol. 1, 2006.

[32] W. Thomas Miller, http://www.ece.unh.edu/robots/cmac.htm, (access-ed 18th May 2007).

[33] Martin John Baker, euclideanspace, http://www.euclideanspace.com/ maths/geometry/rotations/conversions/matrixToQuaternion/index.htm (accessed 18th May 2007).

[34] S. B. Niku, *An introduction to robotics analysis, systems, applications*, Upper Saddle River, N.J., Prentice Hall, c2001

[35] S. Haykin, *Neural networks: a comprehensive foundation.* Pretice-Hall, Toronto, 1999.

[36] J. R. Jang, E. Mizutani, and C. Sun, *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence.* Upper Saddle River, NJ: Prentice Hall, c1997.