A POTENTIAL FIELD APPROACH FOR DISTRIBUTED CONTROL OF DISCRETE ACTUATOR AND SENSOR ARRAYS

by

PRITPAL S. DANG

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2007

ACKNOWLEDGEMENTS

While working as a Research Assistant with Frank L. Lewis, I have attained a firsthand education in effective problem identification and solving, building coalitions, and navigating a rigorous work environment where expectations are high and the pace is fast. I thank him profusely for guiding me in my quest to attain academic excellence.

I would also like to acknowledge my other committee members Dr. Harry Stephanou, Dr. Dan Popa, Dr. Kamesh Subbarao, Dr. Mason Graff for their very valuable inputs throughout my research work.and also for taking out their time to be a part of my dissertation.

My parents, and my wife have willingly sacrificed much so that I could afford to spend time to complete my research. Without their infinite support I could not be where I am right now. Enough can never be said for their everlasting support and inspiration.

I would also like to thank all my colleagues in ACS group and DIAL group for their support and motivation.

November 5, 2007

ABSTRACT

A POTENTIAL FIELD APPROACH FOR DISTRIBUTED CONTROL OF DISCRETE ACTUATOR AND SENSOR ARRAYS

Publication No. ______

Pritpal S. Dang, PhD.

The University of Texas at Arlington, 2007

Supervising Professor: Frank L. Lewis

This dissertation presents new methods of distributed sensing, classification and actuation using potential field and neural network approaches. Humanoid robot serves as an excellent tool to implement the strategies developed for sensing, classification and actuation. Achieving dexterity in human-machine interaction or human look-a-like robot helps in better understanding of dynamic and complex environment. This work deals with the issues related to gain autonomy and adaptability through distributed sensing, classification and actuation capabilities. The work presented here starts by mentioning novel strategies employed for morphing a flexible structure to achieve the desired expressions on the artificial skin and then moves on to classification of facial expressions for humanoid robot and the final part mentions about the strategies for sensing and localization of mobile platforms.

A complete humanoid robot is one which not only is able to classify expressions but also achieve the required expression by morphing the flexible surface (artificial skin). In this work, a control law is designed using a certain potential field for morphing flexible structures with and without model parameter uncertainties and it also presents a novel method called as Extended Orthogonal Least Square (EOLS) method to select the actuators that needs to controlled. The EOLS method achieves the desired shape of the flexible surface by optimally activating the microactuators embedded in the flexible surface.

This work also presents a novel and robust method to classify expressions using a two stage neural network. Expression classification plays a major role to increase human-machine interfacing. It also serves as a tool for humanoid robots to gain more realistic behavior. A new approach termed as semi-decoupled extended Kalman filter is proposed for 3D object pose estimation. This approach helps in gaining robust classification when a human face is under motion.

Distributed sensing has received considerable attention in the field of localization of mobile platforms. The mobile platforms could be aerial vehicles, humanoid robots or ground vehicles. This work presents a novel approach based on dynamics localization to estimate the relative and absolute position of the mobile platforms.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ABBREVIATIONS

AOA	Angle of Arrival
AU	Action Units
DCT	Discrete Cosine Transform
EKF	Extended Kalman Filter
EOLS	Extended Orthogonal Least Square
FACS	Facial Action Coding System
GCS	Ground Control Station
GHA	Generalized Hebbian Algorithm
KLT	Karhunen-Loeve Transform
LDA	Linear Discriminant Analysis
LVQ	Learning Vector Quanitzation
NN	Neural Network
OLS	Orthogonal Least Square
PCA	Principal Component Analysis
PW	Principal Wrap
RBF	Radial Basis Function
ROC	Receiver Operating Characteristics

RSSI	Received Signal Strength Indication
SISL	Stable in the Sense of Lyapunov
SLAM	Simultaneous Localization and Mapping
TOA	Time of Arrival
TPS	Thin Plate Spline
UGS	Unattended Ground Sensor
UAV	Unmanned Aerial Vehicles
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

A complete system is defined as one which can know of its surroundings, classify from various activities and then take a particular action to perform the classified activity. Robustness, adaptability and autonomy are the standard paradigms for an engineer to achieve in a complete system. Engineering designs are always inspired from nature and prefer complete functionality. Therefore, this research lays out an approach to attain robustness, adaptability and autonomy to attain a complete system.

In the scope of the research work presented here, a system is defined with multiple actuators and sensors, which are termed as discrete actuator and sensor arrays. Therefore, in order to perform a task, the system should have control over the discrete actuator and sensor arrays. In this work, a distributed control scheme is developed based on a certain potential field concept to gain control over the multiple actuator and sensor arrays. While recent technological developments have enabled to put multiple actuators and sensors in one system, the full benefit will only be utilized when they can be used in an optimal manner to achieve adaptability and autonomy. That is, the measurements from multiple sensors could be used in an efficient way such that maximum information could be drawn about the system and with minimum use of the actuators in the system a certain task could be performed.

As mentioned earlier, the distributed control algorithms are determined using a certain potential field function. Potential field methods [1 - 5] have been widely applied in the field of mobile robot path planning and navigation. These methods gained popularity because of its mathematical simplicity and elegance, since they require only local gradient information which also makes them easy to be implemented in real time. The concept of potential field methods is to create a minimum at the goal configuration (attractive potential field) while taking into account the effect of all the obstacles (repulsive potential field) present in the environment. Some of the potential fields based on this idea are given in [3, 4, 6 - 10].

In this research work, the potential field methods are explored further and applied in the field flexible surface morphing having discrete actuators embedded/bonded on the surface. The application of the flexible surface morphing is in the field of artificial skin morphing for generating different facial expressions by a humanoid robot. The humanoid robot should be able classify the facial expressions in order to generate the correct expression, therefore an algorithm for expression classification based on a two-stage neural network is also presented in this research work.

In addition to the control of discrete actuators, the potential field methods are also used to localize the wireless sensor nodes randomly deployed in an unknown terrain. The localization of wireless sensor nodes using the potential field method is widely applicable to stationary and mobile platforms such as humanoid robots, unmanned aerial vehicles (UAVs), autonomous ground robots, etc.

One of the major drawbacks of the potential field method is that it gives rise to problem pertaining to local minimum. The local minimum arises due to the superposition of the attractive and repulsive potential field. Due to existence of local minima, the solution of the problem might get stuck up into the wrong solution or even no solution. In this research, based only on the range or position measurements, a certain potential field is created for distributed control of actuator and sensor arrays and the problem related to the local minima are addressed effectively.

The work presented over here starts with the problem description and then presents a review of the related work in the field of distributed control of discrete actuators and sensor arrays.

1.1 Problem Statement

In this research, we investigate the problem of developing distributed control algorithms for the control of discrete actuator and sensor arrays applicable to flexible surface shape morphing and localization of autonomous mobile/stationary platforms, respectively. The emphasis is on the

(i) Mathematical formulation for the system with multiple actuators and sensors.

(ii) Development of control strategies for efficient shape morphing.

(iii) Development of expression classification algorithm to get desired shape to be used for shape morphing with pose estimation.

(iv) Development and implementation of control architecture for localization sensor nodes.

1.2 Review of Related Work

1.2.1 A Potential Field Approach for Distributed Control of Discrete Actuators

Recent technological developments have lead to a growing interest in the field of shape morphing of flexible surface. With the availability and development of microactuators, it has become possible now to embed/bond these microactuators on a flexible surface and attain shape morphing phenomenon. The research in shape morphing is inspired by nature, where it is observed by the wings of birds as they travel at different maneuver and also by human beings to represent different expressions. The proposed research extends this biological inspiration to develop an efficient methodology for achieving the desired shape morphing autonomously.

To efficiently achieve shape morphing, the problem pertaining to distributed control of discrete actuator arrays needs to be addressed. The basic premise of shape control algorithm is the force transmission via elastic deformation with embedded/bonded actuators. The benefit of having a distributed control algorithm is that it will provide advantages in application requiring robustness and adaptability and will aid in achieving energy efficiency and size reduction of the structure.

Flexible structures are used in many applications such as artificial skin morphing, wing morphing for aircrafts, surgical instruments, space robotic systems, space antennas, etc. A lot of research has been done to deal with the effects of structural vibration and improve the performance [11 – 13], whereas the shape control of 2-D flexible structure is relatively new area and a lot of effort is being put on designing robust and optimal control methods [11, 13, 14]. The advantage of having a shape control system with microactuators is that a precise structural deformation could be obtained which also allows the weight of the flexible structure to be reduced by controlling only few actuators.

Active shape control of flexible structures through distributed networks of microactuators has been an active area of research. Recent technological developments have enabled to embed or bond microactuators to the flexible structures for its shape control. Flexible structures have widespread applications in the field of satellite communication antennae, space robotic systems and humanoid robotic systems, etc. The desire to control the shape of flexible structures is one of the major concerns to achieve accuracy and precision, during its operation. Various approaches for shape control algorithm are presented in [11 - 14].

Optimal actuator placement and the number of actuators used is also the key to gain shape control for the flexible structures, to be light in weight and operate under minimum power consumption. The main contribution of this research work is the determination of the optimal number and placement of microactuators, along with the control input law designed using the potential field concept.

Considerable research has been devoted to actuator placement methods for vibration control of a 2D structure. Various methods such as energy-based approach [15], genetic Algorithms (GA) [16, 17], have been proposed for vibration suppression of 2-Dimensional structures. These methods could also be applied for shape control, but they prove out to be computationally complex to implement. In [18], a method based on binary variables is used to determine the actuator layout and number of actuators to be used. Another method for actuator placement based on finite element (FE) method is given in [19]. In [20] and [21], methods based on Tabu Search and Simulated Annealing are discussed, respectively. Simulated annealing method given in [21] is combined with Monte Carlo approach to estimate the actuator positions. Another method based on genetic algorithm, to determine sensor and actuator locations and feedback gains simultaneously, is given in [22]. Genetic algorithm is combined with finite element analysis for actuator position in [23]. All these methods tends to have low convergence rate and high computation cost.

1.2.2 Expression Classification

The development of man-machine interfaces has been an active area of research. One of the key features for modern day human-like robots is to have the capability of imitating human-like facial expressions. In order to enhance the communication between man and human-like robots, it is very important to recognize and understand facial expressions. Facial expressions serve as an excellent tool for non-verbal communication and human interaction. Since facial expressions are responsible for 55% for communicating emotions as mentioned in [29], therefore it becomes imperative to recognize facial expressions in order to enhance the communication between man and humanoid robots. The design of intelligent algorithms for facial expression methodology achieves artificial sociability and enables the development of natural adaptive interfaces. Intelligent facial expression recognition systems have wide applications in the field of surveillance and security systems, man-machine interface and biomedical field.

The facial expression recognition problem has been addressed by many researchers using various approaches. Facial action coding system (FACS) has been a standard tool for expression recognition methods being developed. FACS was developed by Ekman [30] and is based on analyzing muscle motion for various expressions in terms of action units (AUs). Some of the approaches making use of FACS are listed in [31, 32]. In [31], an expression recognition system based on optical flow and a hidden Markov model is proposed. The approach in [31] relies on analyzing only the upper face to extract the key feature points for expression recognition.. [32] proposes a method to recognize the facial action dynamics based on FACS. Another method utilizing the FACS concept proposed in [33] uses a three step approach for expression classification. This method uses principal components for dimension reduction, feature selection and linear discriminant analysis (LDA) for expression classification. Various approaches based on kernel methods have proposed in [34, 35] for facial expression recognition. In [34], kernel based method on support vector machines (SVM) is proposed for expression classification and [35] proposes a method based on kernel canonical correlation analysis (KCCA). These methods have been effective, but there have been issues related to the computational cost-effectiveness and operating time.

In order to achieve high computational efficiency, statistical methods needs to be avoided and recursive methods such as neural networks need to be employed.

Various facial expression recognition schemes based on neural networks have also been addressed in [36 - 40]. Neural networks methods are very interesting and promising, since when trained properly, they easily map the extracted feature space to facial expression space and are able to form distinct decision boundaries. As compared to the expression classification based on statistical method in [31 - 35], neural network methods are more robust and have a high percentage of correct classification rate.

Designing an effective neural network has always been a challenging task. In [36], a back-propagation and radial basis function (RBF) neural network is proposed for expression classification and they reported a classification rate of 71.8% and 73.2% for backpropagation and RBF, respectively. In the neural network approach mentioned in [36] the whole face image was analyzed for classification which could lead to a time consuming process, if the image size is large. In [37], a backpropagation neural network is given for classification and it uses different attributes of the face image to classify various expressions, but the image size is initially reduced which destroy the accuracy of achieving high classification rate. A constructive feed-forward neural network is proposed in [39], which uses the discrete cosine transform (DCT) in image data pre-processing. This approach needs an image corresponding to neutral expression of a person to classify other expressions. Another approach based on radial basis function neural network is given in [40], which also uses FACS methodology and achieves a classification rate of 88%. The aim of this research is to propose a recursive recognition method based on neural network which is computationally effective and has higher classification rate.

A complete facial expression recognition system consists of image data preprocessing system and expression classification system. The research presented here concentrates on developing adaptive algorithms based on neural network technique for data preprocessing and expression classification, under the assumption that the image acquired is a well-framed face image for analysis. The first stage neural network consists of several parallel NN banks [41] to estimate the principal components of certain sub-images of the facial image. The inputs to the individual NN banks describe key attributes of the face image. The neural network banks allow the system to be more robust in the case of two different facial expressions that have similar characteristics; for example, a person might have the same movement of muscles near the eye region for two different expressions. In this scenario several neural networks banks are helpful. The advantage of using NN instead of statistical based method is that it gives the flexibility of online adaptive reformulation and using NN for PC estimation does not require having all the information at once i.e. batch processing is avoided. As the images are assimilated one by one, the PC are updated sequentially. The estimated principal components of the key attributes of the face and the second stage of neural network are used to classify the facial expressions such as happy, sad, fear, anger, surprise, disgust or neutral.

1.2.3 3D Object Pose Estimation

In order to obtain expression classification from a moving human face, it is important to know the pose of the human head i.e. the position and orientation of the human head. Therefore, in this research, pose estimation of 3D objects (human face) is proposed using the concept of Extended Kalman filter approach. The role of vision guided control is important and in a wide range of applications such as humanoid robots, manufacturing environments, satellite and missile control and creating augmented reality motion. Recently, due to the technical advancements and increased availability of camera systems, robotic cameras are routinely used in surveillance [58], on movie sets [59], or in industrial robotics. One of the main issues related to the vision guided control system is the evaluation of the position and orientation vector of an object from the camera field of view, the so-called pose estimation problem. The pose estimation is also combined with the visual servoing method for object tracking and increased field of view. Visual servoing is typically achieved using an image based Jacobian, or a combination of position and image [60, 61] based methods. Certain applications, such as automated space-station docking, or the interaction between humans and humanoid robots require the estimation of an object pose in 6D, and visual tracking of the object based on that pose. Even more challenging, applications can sometimes provide a single "eye-in-hand" camera view, as opposed to stereo vision, leading to the so-called 2 1/2D vision servoing problem [62]. The corresponding pose determination problem is to calculate the three-dimensional (3D+3D) position and orientation of an object from a set of feature points captured by two-dimensional (2D) images.

Considerable research has already been conducted [63-66] and various approaches have been proposed to determine the pose of an object using robot cameras. In [63], an approach based on active appearance model (AAM) is proposed, which is based on model matching. A neural network method to determine the pose of an object is proposed in [64], which requires offline training of the neural network and it fails if the object is cubical in shape. In [65], a geometric based approach is given, which uses 4 features points to obtain the pose parameters and an averaging method is used to deal with system noise. Another method based on hypothesis-testing logic is proposed in [66], which also uses more than 6 feature points.

Since visual measurements are highly affected by the system noise due to lens distortion, lighting and background inconsistency, inconsistency in image processing algorithms, etc, the pose estimation algorithm may produce large errors. This issue of system noise could be significantly avoided by the use of a Kalman filter, which improves the accuracy of the estimation process. Various approaches using Kalman filter for pose determination have been effectively adopted in [67 - 72]. In [67, 68], an approach based on Kalman filter is proposed, which uses 5 non-coplanar feature points with a single camera. Use of multiple cameras with Kalman filter has been proposed in [69, 72]. These methods use more than 4 non-coplanar feature points for accurately determining the pose parameters.

Since the 2D camera measurements are related through a nonlinear projection relationship with the pose parameters, an Extended Kalman filter (EKF) is needed. The EKF is an extension of linear Kalman Filter for nonlinear dynamical systems. The EKF plays an important role in dealing with noise and predicting the output in case the measurements are not available. It is well-known that such filters are sensitive to state vector and covariance matrix initial conditions. Inappropriately chosen parameters could produce large errors or even to the divergence of the estimates. For more on the divergence of the extended Kalman Filter see [73].

In this research we propose and experimentally validate an efficient, semi-decoupled EKF-based 6D pose estimation algorithm and combine it with object tracking with a robotic camera. The semi-decoupled pose estimation algorithm uses two separate filters, one for object 3D position and one for object 3D orientation vectors. We call it "semi-decoupled" because only the position vector is passed to the orientation filter, and not vice-versa. This scheme makes the pose estimates less sensitive to image depth, and as a result, smaller tracking errors and improved robustness to initial conditions are obtained compared to a fully coupled 6D filter. In addition, this implementation is more computationally efficient because of reduced matrix dimensions. Furthermore, pose estimation using the proposed approach is accomplished with only 3 non-colinear feature points. The semi-decoupled EKF method was found to be robust to large initial condition errors and noise through the set of experiments conducted for the current experimental setup.

1.2.4 A Potential Field Approach for Distributed Control of Discrete Sensors

Distributed sensing capabilities are used to localize the mobile platforms in an unknown terrain. The mobile platforms could be aerial vehicles, humanoid robots or ground vehicles. In this work, autonomous aerial vehicles are taken as a platform to highlight the effectiveness of the proposed approach.

The role of autonomous surveillance has proven to be important and applicable to a wide range of applications such as target location, map building, border security, pollution monitoring and control, and battle damage assessment. UAV (unmanned aerial vehicles) fit into the scenario of autonomous surveillance perfectly as they involve a low risk factor and facilitate technological advancements, making their use feasible in real world scenarios. UAV are generally classified by their flight altitude, launch and recovery methods as detailed in [106].

UAV, together with randomly deployed stationary unattended ground sensors (UGS) can further enhance the performance of the autonomous surveillance tasks mentioned above. Since the information collected from UGS is of limited use if no information about the sensor position is available, the task of localizing the sensor nodes is of prime importance for sensor network applications. In this paper we discuss an air-ground localization scheme in which UGS nodes, with the aid of UAV having on board GPS, generate their optimal position estimates.

Localization is classified in two categories: relative and absolute localization. In relative localization the sensor nodes are localized using the distances measured among the nodes with respect to an arbitrary internal coordinate system. Absolute localization, on the other hand localizes the network with respect to a known specified coordinate system.

An important development is an air-ground localization scheme, which performs relative and absolute localization of stationary UGS network with the aid of UAV. The UGS nodes are simple and support local sensing, communication and computation. It becomes impractical to have GPS capability on UGS nodes due to energy and cost constraints. UAV having GPS can be used to localize the UGS network absolutely.

The UGS node localization problem has been addressed by many researchers using various approaches. In [103] a distributed algorithm is proposed, where a fraction of nodes are already localized. Other schemes for localization such as SHARP, virtual force algorithm (VFA) and self-localization have been proposed in [85, 99, 107] respectively. Other methods using fixed beacons with known positions are proposed in [88, 89, 102, 106]. Various approaches involving RSSI, TOA, AOA, signal pattern matching are explained in [90]. A detailed introduction to localization in sensor networks is given in [86]. Cooperative localization methods have been developed for relative localization in [100].

Air-ground localization schemes based on terrain-aided navigation have been addressed in [94 – 96, 101]. These algorithms are known as Simultaneous Localization and Mapping (SLAM). Other airborne localization algorithms involving regular broadcast of UAV location have been proposed in [92]. These localization schemes are dependent on the path of UAV with respect to the deployed nodes and are restricted by power consumption and network congestion problems due to the regular broadcasts of UAV positions. In [98], problem of localizing vehicles where GPS signals are often unavailable is described.

With no priori terrain information available the problem of localization is extremely challenging and various approaches based on the Extended Kalman filter (EKF) have been proposed in [96, 97, 101]. The EKF is a recursive estimation technique based on first-order linearization of the nonlinear system, and can yield large estimation errors and even divergence of the filter. Thus, these techniques are sometime not very effective. For more on the divergence of the Kalman filter, see [73].

The research presents an alternative method of relative and absolute localization based on a potential field method [93]. Two algorithms are given: relative localization algorithm and absolute localization algorithm. A dynamical model for each sensor node estimates the relative positions by employing a correction term based on a certain fictitious virtual force. In the relative localization algorithm the stationary UGS nodes are localized with respect to an internal coordinate frame. The relative localization algorithm proposed in the work assumes that distance (i.e. range) measurements between sensor nodes are available. For absolute localization, it is assumed that some nodes have GPS absolute position information. Specifically, herein, the UGS nodes are localized with respect to a global frame provided using the absolute positions of several UAV with GPS as shown in Figure 1.1.

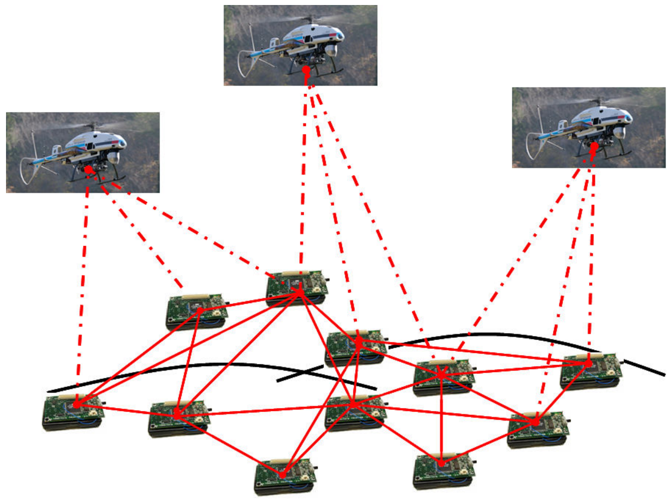


Figure 1.1 Air Ground Sensor Network Configuration

1.3 Dissertation Layout

The dissertation is organized in the following way: Chapter 2 presents the methods developed for distributed control of discrete actuator arrays along with any model uncertainties of the flexible surface. In Chapter 3, a novel method is developed for selction/placement of microactuators to morph a physical flexible structure. Chapter 4 proposes a two stage neural network for expression classification to acquire a desired shape. Chapter 5 outlines the approach of acquiring the expression under the influence when the body is mobile. The distributed control of sensor arrays for localization is mentioned in Chapter 6.

1.4 Mathematical Preliminaries and Notations

This section describes the basic definitions and background material related to the control of dynamical systems. More details are described in [83, 84].

1.4.1. Norm of a Vector

The *p*-norm for any vector $x \in \Re^n$ is given as,

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{1.1}$$

and for $p = 2$, the norm is defined as a Euclidian norm.

1.4.2 Quadratic Form

If $Q \in \Re^{n \times n}$ is a positive semi-definite matrix then for any vector $x \in \Re^n$ then the following inequality holds and is defined as quadratic form

$$\sigma_{\min}(Q)\|x\|^2 \le x^T Q x \le \sigma_{\max}(Q)\|x\|^2 \tag{1.2}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the minimum and maximum eigenvalues of the matrix Q, respectively.

1.4.3 Stability

Stability is a performance requirement for closed loop systems. For the dynamical system given as

$$\dot{x} = f(x,t) \tag{1.3}$$

where $x \in \Re^n$, a state x_e is an equilibrium point of the system if $f(x_e,t) = 0$, $t \geq t_0$. Therefore, an equilibrium point x_e is stable in the sense of Lyapunov (SISL) at t_0 if for every $\varepsilon > 0$ there exists a $\delta(\varepsilon,t_0) > 0$ such that $\|x_0 - x_e\| < \delta(\varepsilon,t_0)$ implies that $\|x_0 - x_e\| < \varepsilon$ for $t \geq t_0$.

An equilibrium point x_e is asymptotically stable at t_0 if there exists a compact set $S \subset \Re^n$ such that, for every initial condition $x_0 \varepsilon S$, one has $\|x_0 - x_e\| \to 0$ as $t \to \infty$.

CHAPTER 2

A POTENTIAL FIELD APPROACH FOR ACTIVE SHAPE CONTROL OF FLEXIBLE STRUCTURES

This chapter presents a methodology for modeling, analyzing and controlling the dynamics of shape morphing for a flexible structure. This paper is organized in the following way: Section 2.1 outline the procedure to derive a generic dynamic model for flexible structure embedded/bonded with microactuators and then gives the approximate mathematical model to be considered for control algorithm design. In Section 2.2, the foundation of deriving the control law for relative and absolute pixel points (micriactuator points) is established. Section 2.3 derives the control law with model uncertainties in the system. The effectivenss of the proosed control law is shown by the simulation results given in Section 2.4. Finally, the conclusion is presented in Section 2.5.

2.1 System Overview

For deriving any control law, a mathematical model of the system is required. The approach towards obtaining the mathematical model of a flexible structure with embedded/bonded structure and deriving the control input for shape control is mentioned. The approach adopted in here is to consider each microactuator point as one pixel as shown in Figure 2.1 and then control individual pixel to create local deformation, which in turn causes the global deformation of the flexible surface.

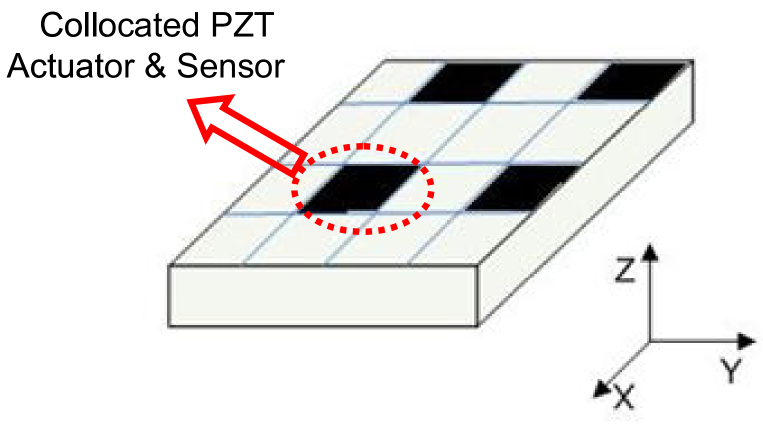


Figure 2.1 Flexible surface with embedded/bonded actuators

Each pixel (microactuator point) has either the information about the desired relative distance it needs to maintain with the other pixels or it has the desired co-ordinate information of where it needs to be. The pixels with relative distance information are termed as relative pixel points and the pixels with desired co-ordinate point informaiton are termed as absolute pixel points.

Since the research results presented here emphasize the design of the control input, the mathematical model of the flexible structure is approximated with a series of inter-connected mass-spring damper system. This mass-spring-damper system imitates the behavior of the actual flexible surface, which serves as an excellent tool to test the effectiveness of the designed control algorithm. The section begins with deriving the mathematical model of the system using Lagrange's equation of motion and then develops the approximated model of the flexible structure based on inter-connected mass-spring-damper.

2.1.1. Exact System Model

Extensive literature exists for the modeling of the flexible structure with embedded/bonded actuators and sensors. Some of the work regarding the modeling of the system with respect to shape control and vibration suppression could be found in [11 - 15]. The approximation is done to highlight the effectiveness of a novel control algorithm presented later in the research work.

The flexible structure is a general collection of deformable bodies, rigid bodies and particles. Their mathematical model is derived in the form of differential equations which forms the basis of their equations of motion. These equations of motion are constructed using Hamilton's principle and Lagrange's equations. In general, the Hamilton's principle is given by

$$\int_{t_1}^{t_1} \delta(T-V)dt + \int_{t_1}^{t_2} \delta W dt = 0 \tag{2.1}$$

where T is the kinetic energy, V is the potential energy (including strain energy and potential energy of external forces), $\delta()$ is the virtual displacement operator and W is the virtual work done by the damping forces and the external forces (not taken into account in V). For the flexible structure with embedded/bonded actuators, the kinetic energy is given as a function of mass density and volumes of the flexible structure. The potential energy is given as a function of stress and strain vectors of the structure, material stiffness properties, electric and field displacement. Therefore, using the energy functions and the Hamilton's principle defined in (2.1), equations of motion are constructed by

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau \tag{2.2}$$

where the Lagrangian L is defined as

$$L = T - V \tag{2.3}$$

and q_i are the generalized coordinates and τ are the generalized forces.

It has been shown in [11 - 13] that using (2.1) and (2.2), the system model could be reduced in the form given as

$$M\ddot{q}_i + C(q_i, \dot{q}_i)\dot{q}_i + K(q_i)q_i = \tau \tag{2.4}$$

where M is the structural inertia matrix, $C(q_i, \dot{q}_i)$ accounts for the velocity effects like viscous friction, centrifugal and coriolis effects, $K(q_i)$ represents the elastic effects due to spring, bending stiffness and gravity, and τ is the force acting on the system by the embedded/bonded actuators. Since the task over here is to show the effectiveness of the novel control algorithm approach, therefore an approximated version of the flexible surface is derived in the next subsection.

2.1.2. Approximate System Model

As given in Section 2.1.1, the elastic nature of the flexible structure can also be represented by a series of inter-connected mass-spring-damper system as shown in Figure 2.2. The equations of motion derived for this system are also in the same structure form as given in (2.4). This approximation of the flexible structure is done to highlight the effectiveness of the control algorithm proposed further in the research work. Each spring-damper system labeled as N_j has the following parameters, Spring constant as K_j, Damping co-efficient as C_j and spring length as R_j for $j = 1,2.....,12$.

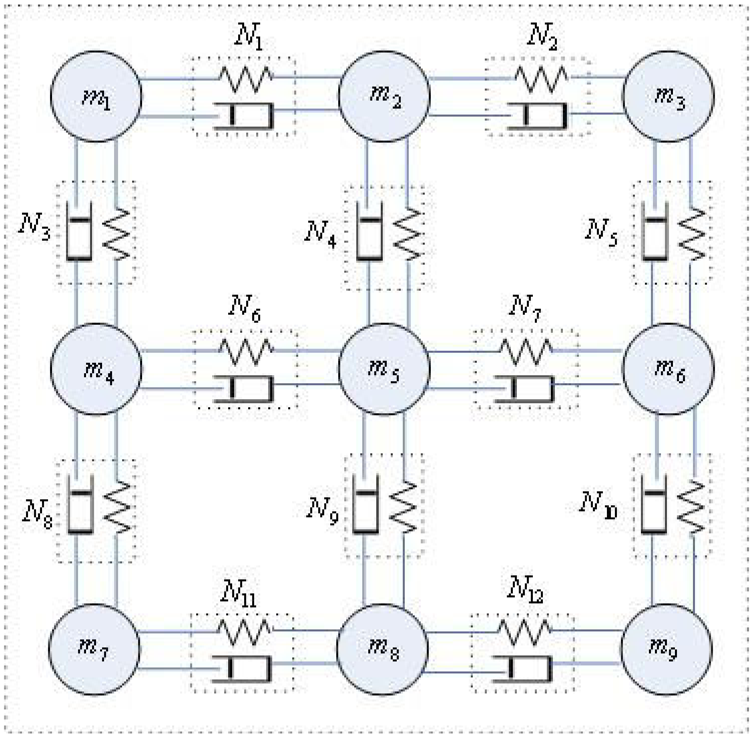


Figure 2.2 Approximated model of flexible structure with inter-connected mass-spring-damper system

Using the Newton's law of motion to derive the equation of motion for the approximated flexible structure in two dimensional (2D), the dynamics of every point mass mode is given as,

$$m_1\ddot{x}_1 = K_1(x_2 - x_1 - R_1) + C_1(\dot{x}_2 - \dot{x}_1) + F_{1x} \tag{2.5}$$

$$m_1\ddot{y}_1 = K_3(y_4 - y_1 - R_3) + C_3(\dot{y}_4 - \dot{y}_1) + F_{1y} \tag{2.6}$$

where x_1 and y_1 are the *x*-*y* coordinates for the point mass position estimate, and F_{1x}, F_{1y} are the force acting in *x* and *y* direction. Similarly the dynamical equation representing every point mass model could be formulated in the standard form given as,

$$M_i\ddot{Z}_i + C_i\dot{Z}_i + K_iZ_i + K_i^D = \vec{F}_i \tag{2.7}$$

for $i = 1,2.....,N$ where N is the total number of point mass to be considered (for ex. in this case $N = 9$), Z_i is the position vector for an actuator node given as $Z_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$, M_i is the mass matrix, C_i is the coriolis matrix, K_i is the stiffness matrix, K_i^D is the vector containing spring lengths and $\vec{F}_i$ is the force vector given as $\vec{F}_i = \begin{bmatrix} F_{ix} & F_{iy} \end{bmatrix}^T$.

2.2 Controller Design for Dynamic Shape Morphing

In the previous section, the mathematical model of a flexible structure is derived. Now, in this section, a control methodology is developed for the shape control of flexible structures with embedded/bonded actuators.

There are two types of actuators points in the system, which are classified as

1) Absolutely actuated mass points
2) Relatively actuated mass points.

Absolutely actuated mass points are the ones which have information about their desired co-ordinates points, whereas relatively actuated mass points only have information about the distance they need to maintain between the corresponding mass actuator points. The dynamic controller designed in this section takes into account the effects of both the absolute actuator mass points and relative actuator mass points on shape morphing. The following subsections describe the controller design procedure for the two different mass actuator points.

2.2.1. Dynamic Controller Design for Relatively Actuated Mass Points

The dynamical model of the structure developed in (2.7) is rewritten in the form given as,

$$\ddot{Z}_i = M_i^{-1}(\vec{F}_i - C_i\dot{Z}_i - K_i Z_i - K_i^D) \tag{2.8}$$

This is done to represent the system model in (2.7) in the state variable representation. In order to control the point mass position, a two step design procedure is used where the system dynamics in (2.8) are further reduced using the preliminary feedback linearization technique [83], given as

$$\ddot{Z}_i = u_i \tag{2.9}$$

where $u_i = M_i^{-1}[\vec{F}_i - C_i\ddot{Z}_i - K_i\dot{Z}_i - K_i^D]$

In the two step design, the first step involves selecting a feedback control $u(t)$ that stabilizes the point mass positions and the second step computes the required force using the inverse relation given as

$$\vec{F}_i = M_i u(t) + C_i\dot{Z}_i + K_i Z_i + K_i^D \tag{2.10}$$

The control law in (2.10) is a nonlinear feedback control that guarantees the convergence of the mass point positions. The control law is derived using a certain potential field to be introduced later in the section, so that the mass point positions reaches steady-state values. The control structure for the feedback control design is shown if Figure 2.3.

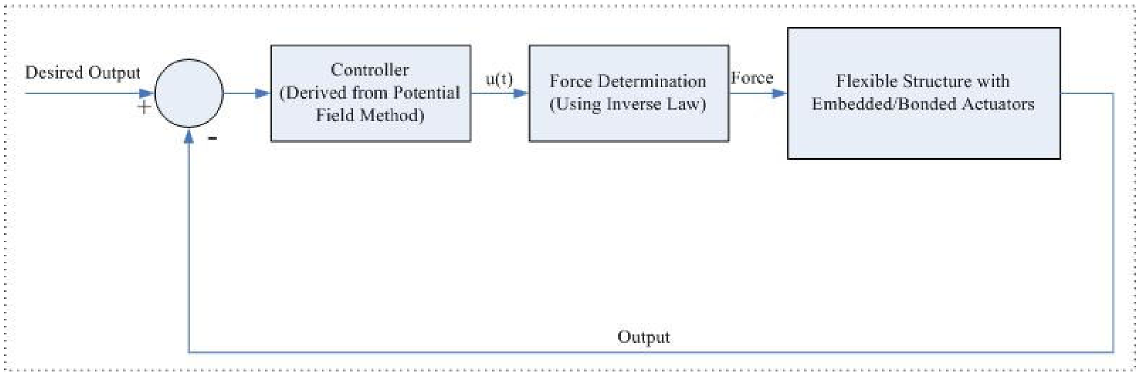

Desired Output
+
-
Controller (Derived from Potential Field Method)
u(t)
Force Determination (Using Inverse Law)
Force
Flexible Structure with Embedded/Bonded Actuators
Output

Figure 2.3 Two step Controller Design for flexible surface shape morphing

It is not always necessary to actuate all the actuators of the flexible surface, therefore a novel approach to select which actuators needs to be controlled is given in Chapter 3. The method described in Chapter 3 selects the mass point that needs to be controlled in order to achieve the desired shape of the flexible surface with minimum error.

For the first step of the control design procedure, the control input $u_i(t)$ is designed using the concept of potential field, which is given as

$$V = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})^2 \tag{2.11}$$

where $r_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ is the calculated distance and $\bar{r}_{ij}$ is the actual measured distance between i^{th} and j^{th} mass points. By defining the potential function for a single mass point i, (2.11) can be written as

$$V_i = \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{2} K_{ij}(r_{ij} - \bar{r}_{ij})^2 \tag{2.12}$$

Now by calculating the gradient of the potential function with respect to the mass point position vector Z_i, the direction and the magnitude at which the control input needs to be applied is obtained, which is given as

$$\frac{\partial V_i}{\partial Z_i} = \vec{\nabla} V_i = \vec{\nabla} \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{2} K_{ij} (r_{ij} - \bar{r}_{ij})^2 = \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \vec{\nabla} (r_{ij} - \bar{r}_{ij}) \tag{2.13}$$

which on further simplification is written as

$$\frac{\partial V_i}{\partial Z_i} = \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \left[\frac{x_i - x_j}{\|Z_i - Z_j\|} \hat{x} \quad \frac{y_i - y_j}{\|Z_i - Z_j\|} \hat{y} \right] \tag{2.14}$$

where $\|Z_i - Z_j\| = r_{ij}$

Theorem 2.1: Consider the position dynamics (2.9) for each relatively actuated mass point m_i in the mass-spring-damper system. Let the control input $u_i(t)$ for point mass m_i be given as,

$$u_i(t) = -\sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \frac{(Z_i - Z_j)}{\|Z_i - Z_j\|} - K_v \dot{Z}_i \tag{2.15}$$

Then the position vector Z_i reaches steady-state of the actual point mass positions in the sense that the potential function V is minimized.

Proof: Define the Lyapunov function

$$L = V + \sum_{i=1}^{N} \frac{1}{2} \dot{Z}_i^T \dot{Z}_i \tag{2.16}$$

Differentiate to obtain

$$\dot{L} = \sum_{i=1}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\dot{r}_{ij} + \sum_{i=1}^{N} \dot{Z}_i^{\,T}\ddot{Z}_i \tag{2.17}$$

One can compute

$$\dot{r}_{ij} = \frac{\dot{Z}_i^{\,T}(Z_i - Z_j)}{\left\| Z_i - Z_j \right\|} \tag{2.18}$$

and on substituting (2.9) and (2.18), we obtain

$$\dot{L} = \sum_{i=1}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\frac{\dot{Z}_i^{\,T}(Z_i - Z_j)}{\left\| Z_i - Z_j \right\|} + \sum_{i=1}^{N} \dot{Z}_i^{\,T} u\ i(t) \tag{2.19}$$

Further, substituting the control input from (2.15) yields

$$\dot{L} = -\sum_{i=1}^{N} \dot{Z}_i^{\,T} K_v \dot{Z}_i \tag{2.20}$$

which on observation clearly states that, $\forall\ K_v > 0$, $\dot{L} \le 0$ and the vector $\begin{bmatrix} Z_i & \dot{Z}_i \end{bmatrix}^T$ is bounded which shows that the position estimate dynamics is SISL. Evaluating $\ddot{L}$ yields

$$\ddot{L} = -2\sum_{i=1}^{N} \dot{Z}_i^{\,T} K_v u_i \tag{2.21}$$

and on substituting (2.15), we obtain

$$\ddot{L} = -2\sum_{i=1}^{N} \dot{Z}_i^{\,T} K_v \left\{ -\sum_{j=1}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\frac{(Z_i - Z_j)}{\left\| Z_i - Z_j \right\|} - K_v \dot{Z}_i \} \right\} \tag{2.22}$$

Using the result obtained from Lyapunov analysis, it can be seen that the vector $\begin{bmatrix} Z_i & \dot{Z}_i \end{bmatrix}^T$ is bounded and yields $\ddot{L}$ in (2.15) to be bounded. By Barbalat's Lemma [105] we deduce that $\dot{L} \to 0$ as $t \to \infty$, which yields $\dot{Z}_i \to 0$ as $t \to \infty$. Therefore (2.9) shows that $u_i(t)$ goes to zero $\forall i$. Finally (2.13) and (2.14) show that $\frac{\partial V}{\partial Z_i} \rightarrow 0, \forall i$, so V reaches a minimum. ■

After obtaining the design of the control input using the potential field concept, the required force vector is obtained using the inverse control law given as,

$$\vec{F}_i = M_i u_i(t) + C_i \dot{Z}_i + K_i Z_i + K_i^D \tag{2.23}$$

2.2.2. Controller Design for Dynamic Absolutely Actuated Mass Points

In the previous subsection, a control law for relative mass points was proposed, whereas in this section, a control law for absolute mass points is proposed. In the absolute mass points, the information about their desired co-ordinates is available together with the distance information between the corresponding mass points.

Here, the mass points from the flexible surface modeling are divided into two sets, one containing the relative mass points and the other containing the absolute mass points. Let the total number of mass points to be considered for flexible surface morphing N indexed by a set Z_i, where $i = \{1,2,3,.......N\}$. From this set, let the number of absolute mass points be m, indexed by a set $Z_{i_p}^a$, where $\{p = 1,2,....., m\}$ such that $Z_{i_p}^a \subset Z_i$ and the relative mass points are indexed by Z_{i_p}, where $\{p = m+1,....., N\}$ such that $Z_{i_p} \subset Z_i$.

From (2.9), the position vector Z_{i_p} dynamics are given as

$$\ddot{Z}_{i_p} = u_{i_p} \tag{2.24}$$

where $Z_{i_p} = \lfloor x_{i_p} \quad y_{i_p} \rfloor^T$, x_{i_p} and y_{i_p} are the *x*-*y* coordinates of the relative mass point and u_{i_p} is a control input to be specified.

Extending the same concept of forming the dynamics, the absolte mass point dynamics are given as

$$\ddot{Z}^a_{i_p} = u^a_{i_p} \tag{2.25}$$

where $Z^a_{i_p} = \left[x^a_{i_p} \quad y^a_{i_p}\right]^T$, $x^a_{i_p}$ and $y^a_{i_p}$ are the *x*-*y* coordinates of the absolute mass points and $u^a_{i_p}$ is a control input to be specified.

The potential function given in (2.11) is modified to include the information about the desired co-ordinates of absolute mass points and is given as

$$V^a = \frac{1}{2}\sum_{p=1}^{m}\sum_{j=1}^{N} K^a_{i_p j}(r^a_{i_p j} - \bar{r}^a_{i_p j})^2 + \frac{1}{2}\sum_{p=1}^{m} K^a_{i_p}\left\|e^a_{i_p}\right\|^2 \tag{2.26}$$

where $e^a_i = \left[(x^a_{i_p} - \bar{x}^a_{i_p})^2 + (x^a_{i_p} - \bar{y}^a_{i_p})^2\right]^{1/2}$ and $\bar{Z}^a_{i_p} = \left[\bar{x}^a_{i_p} \quad \bar{y}^a_{i_p}\right]^T$ is the desired co-ordinates of the mass point Z^a_i. Now, the total potential function of the mass points containing relatively and absolutely actuated mass points is given as

$$V_p = V^a + V \tag{2.27}$$

which on substitution of (2.11) and (2.26) is written as

$$V_p = \frac{1}{2}\sum_{p=1}^{m} K_{i_p}^a \left\| e_{i_p}^a \right\|^2 + \frac{1}{2}\sum_{p=1}^{m}\sum_{j=1}^{N} K_{i_p j}^a (r_{i_p j}^a - \bar{r}_{i_p j}^a)^2 + \frac{1}{2}\sum_{p=m+1}^{N}\sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \bar{r}_{i_p j})^2 \tag{2.28}$$

where $r_{i_p j}^a = [(x_{i_p}^a - x_j)^2 + (y_{i_p}^a - y_j)^2]^{1/2} = \left\| Z_{i_p}^a - Z_j \right\|$.

By defining the potential field for a single absolute mass point i_p^a as

$$V_{i_p}^a = \frac{1}{2} K_{i_p}^a \left\| e_{i_p}^a \right\|^2 + \frac{1}{2}\sum_{\substack{j=1 \\ j \neq i_p}}^{N} K_{i_p j}^a (r_{i_p j}^a - \bar{r}_{i_p j}^a)^2 \tag{2.29}$$

and calculating the gradient of the potential with respect to the absolute mass point position vector $Z_{i_p}^a$ given as,

$$\frac{\partial V_{i_p}^a}{\partial Z_{i_p}^a} = K_{i_p}^a (Z_{i_p}^a - \bar{Z}_{i_p}^a) - \sum_{j=1}^{N} K Z_{i_p j}^a (r_{i_p}^a - \bar{r}_{i_p j}^a) \frac{(Z_{i_p}^a - Z_j)}{\left\| Z_{i_p}^a - Z_j \right\|} \tag{2.30}$$

one can compute the control input to be applied to the absolute mass points.

Theorem 2.2: Consider the position dynamics (2.24) for each relative point mass m_i in the mass-spring-damper system and (2.25) for each absolute point mass m_i^a Let the control input $u_{i_p}(t)$ for relative point mass m_i and the control input $u_{i_p}^a$ for absolute mass point m_i^a be given as

$$u_{i_p} = \sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \bar{r}_{i_p j}) \frac{(Z_{i_p} - Z_j)}{\left\| Z_{i_p} - Z_j \right\|} - K_v \dot{Z}_{i_p} \tag{2.31}$$

$$u^a_{i_p} = -K^a_{i_p}(Z^a_{i_p} - \overline{Z}^a_{i_p}) - \sum_{j=1}^{N} K^a_{i_p j}(r^a_{i_p} - \overline{r}^a_{i_p j})\frac{(Z^a_{i_p} - Z_j)}{\left\| Z^a_{i_p} - Z_j \right\|} - K^a_v \dot{Z}^a_{i_p} \tag{2.32}$$

Then the mass point position reaches steady-state values in the sense that V_p is minimized.

Proof: Define the Lyapunov function as

$$L_p = V_p + \frac{1}{2}\sum_{p=1}^{m} \dot{Z}^{a\,T}_{i_p} \dot{Z}^a_{i_p} + \frac{1}{2}\sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\ T} \dot{Z}_{i_p} \tag{2.33}$$

Differentiate to obtain

$$\dot{L}_p = \dot{V}_p + \frac{1}{2}\sum_{p=1}^{m} \dot{Z}^{a\,T}_{i_p} \ddot{Z}^a_{i_p} + \frac{1}{2}\sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\ T} \ddot{Z}_{i_p} \tag{2.34}$$

One can compute

$$\begin{aligned}\dot{V}_p &= \sum_{p=1}^{m} K^a_{i_p} \dot{Z}^{a\,T}_{i_p}(Z^a_{i_p} - \overline{Z}^a_{i_p}) + \sum_{p=1}^{m}\sum_{j=1}^{N} K^a_{i_p j}(r^a_{i_p j} - \overline{r}^a_{i_p j})\dot{Z}^{a\,T}_{i_p} \frac{(Z^a_{i_p} - Z_j)}{\left\| Z^a_{i_p} - Z_j \right\|} \\ &+ \sum_{p=m+1}^{N}\sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \overline{r}_{i_p j})\dot{Z}_{i_p}^{\ T} \frac{(Z_{i_p} - Z_j)}{\left\| Z_{i_p} - Z_j \right\|}\end{aligned} \tag{2.35}$$

and on substitution of (2.30), (2.32) and (2.46) in (2.45) we get

$$\begin{aligned}\dot{L}_p &= \sum_{p=1}^{m} K^a_{i_p} \dot{Z}^{a\,T}_{i_p}(Z^a_{i_p} - \overline{Z}^a_{i_p}) + \sum_{p=1}^{m}\sum_{j=1}^{N} K^a_{i_p j}(r^a_{i_p j} - \overline{r}^a_{i_p j})\dot{Z}^{a\,T}_{i_p} \frac{(Z^a_{i_p} - Z_j)}{\left\| Z^a_{i_p} - Z_j \right\|} \\ &+ \sum_{p=1}^{m} \dot{Z}^{a\,T}_{i_p} u^a_{i_p} + \sum_{p=m+1}^{N}\sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \overline{r}_{i_p j})\dot{Z}_{i_p}^{\ T} \frac{(Z_{i_p} - Z_j)}{\left\| Z_{i_p} - Z_j \right\|} + \sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\ T} u_{i_p}\end{aligned} \tag{2.36}$$

Further substituting the control input mentioned in (2.31) and (2.32) yields

$$\dot{L}_p = -\sum_{p=1}^{m} \dot{Z}_{i_p}^{a\,T} K_v^a \dot{Z}_{i_p}^a - \sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\,T} K_v \dot{Z}_{i_p} \tag{2.37}$$

clearly, $\forall \quad (K_v, K_v^{\,a}) > 0$, $\dot{L}_p \le 0$ and the vector $\begin{bmatrix} Z_{i_p} & \dot{Z}_{i_p} \end{bmatrix}^T$ and $\begin{bmatrix} Z_{i_p}^a & \dot{Z}_{i_p}^a \end{bmatrix}^T$ is bounded which shows that the system is SISL. Evaluating $\ddot{L}_p$ yields

$$\ddot{L}_p = -2\sum_{p=1}^{m} \dot{Z}_{i_p}^{a\,T} K_v^{\,a} \ddot{Z}_{i_p}^a - 2\sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\,T} K_v \ddot{Z}_{i_p} \tag{2.38}$$

which on substitution of (2.28) and (2.30) gives

$$\ddot{L}_p = -2\sum_{p=1}^{m} \dot{Z}_{i_p}^{a\,T} K_v^a u_{i_p}^a - 2\sum_{p=m+1}^{N} \dot{Z}_{i_p}^{\,T} K_v u_{i_p} \tag{2.39}$$

Using the result from the Lyapunov analysis that the vector $\begin{bmatrix} Z_{i_p} & \dot{Z}_{i_p} \end{bmatrix}^T$ and $\begin{bmatrix} Z_{i_p}^a & \dot{Z}_{i_p}^a \end{bmatrix}^T$ is bounded also yields that $\ddot{L}_p$ is also bounded. By Barbalat's Lemma [105] we deduce that $\dot{L}_p \to 0$ as $t \to \infty$, which yields $\dot{Z}_{i_p}^a \to 0$ and $\dot{Z}_{i_p} \to 0$ as $t \to \infty$. Therefore (2.28) shows that u_{i_p} goes to zero $\forall i_p$ with no absolute position information and (2.30) shows that $u_{i_p}^a$ goes to zero $\forall i_p^a$ with absolute position information. Finally (2.37) and (2.38) shows that $\frac{\partial V_{i_p}}{\partial Z_{i_p}} \to 0$ and $\frac{\partial V_{i_p}^a}{\partial Z_{i_p}^a} \to 0$ respectively, so V_p reaches a minimum. ■

From the control input derived in equation (2.31) and (2.32), the force acting on relatively and absolutely actuated mass points is given in (2.40) and (2.41), respectively.

$$\vec{F}_{i_p} = M_i u_{i_p}(t) + C_i \dot{Z}_i + K_i Z_i + K_i^D \tag{2.40}$$

$$\vec{F}_{i_p}^{a} = M_i u_{i_p}^{a}(t) + C_i \dot{Z}_i + K_i Z_i + K_i^D \tag{2.41}$$

2.3 Controller Design for Dynamic Shape Morphing with Parameter Uncertainty

In the previous section, a dynamic shape control algorithm is derived for flexible structure without consideration of model uncertainties. In reality, model uncertainties always exist in the mathematical model derived, therefore consideration of model uncertainties leads into designing an effective control algorithm. In this section, the control algorithm developed in section 2.2 is modified to include the uncertainties in the model parameters. Generally, the common uncertainties include the unknown mass points parameters such as mass, spring constant and damping co-efficient. The control structure with model uncertainties is shown in Figure 2.4.

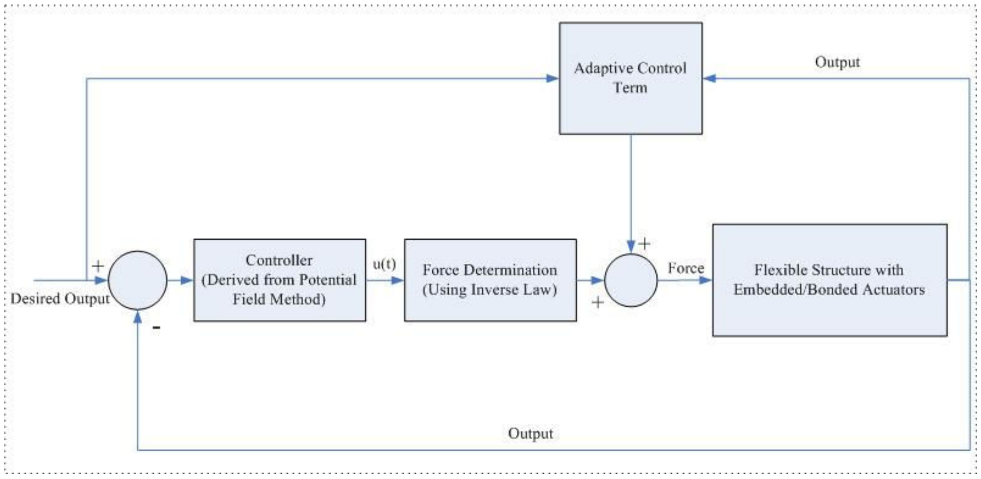

Adaptive Control Term
Output
+
+
Desired Output
Controller (Derived from Potential Field Method)
u(t)
Force Determination (Using Inverse Law)
+
Force
Flexible Structure with Embedded/Bonded Actuators
-
Output

Figure 2.4 Control Structure with model uncertainties

One way of dealing with these model uncertainties is to use the control law derived in Section 2.2 with some fixed estimate of the unknown parameters in place of the actual parameters. This leads to the approximate version of the force input, which is given as

$$\hat{\vec{F}}_i = \hat{M}_i u_i + \hat{C}_i \dot{Z}_i + \hat{K}_i Z_i + \hat{K}_i^D \tag{2.42}$$

where $\hat{M}_i$, $\hat{C}_i$, $\hat{K}_i$ and $\hat{K}_i^D$ are the unknown actual parameters. Analyzing the property of the system model derived, it is clear that the parameters appear linearly in the model. That is, the flexible surface dynamics can be written in the form given as

$$\vec{F}_i = W(Z_i, \dot{Z}_i, \ddot{Z}_i)\varphi \tag{2.43}$$

where $W(Z_i, \dot{Z}_i, \ddot{Z}_i)$ is an $n \times q$ matrix of known functions termed as regression matrix and φ is an $q \times 1$ vector of unknown constant parameters. This linear in the parameter property is very crucial in the design of the adaptive control law to be introduced later in the section.

The first step in the design of the adaptive control law is to rewrite (2.42) in the form given by

$$\hat{\vec{F}}_i = \hat{M}_i u - \hat{M}_i \ddot{Z}_i + \hat{M}_i \ddot{Z}_i + \hat{C}_i \dot{Z}_i + \hat{K}_i Z_i + \hat{K}_i^D \tag{2.44}$$

which is obtained by adding and subtracting $\hat{M}_i \ddot{Z}_i$ in (2.42), yielding

$$\hat{\vec{F}}_i = \hat{M}_i (u - \ddot{Z}_i) + W\hat{\varphi} \tag{2.45}$$

where $W\hat{\varphi} = \hat{M}_i \ddot{Z}_i + \hat{C}_i \dot{Z}_i + \hat{K}_i Z_i + \hat{K}_i^D$. Now rewriting the mass point position dynamics from (2.45) as

$$\ddot{Z}_i = u - \hat{M}_i^{-1}(\vec{F}_i - W\hat{\varphi}) \tag{2.46}$$

and on substitution of (2.43) yields

$$\ddot{Z}_i = u - \hat{M}_i^{-1}\tilde{\varphi} \tag{2.47}$$

where $\tilde{\varphi} = \varphi - \hat{\varphi}$ is the parameter error. The required task is now to determine a tuning law which would guarantee the convergence of parameter error $\tilde{\varphi}$, which is shown by the following theorem. The theorem shows the convergence result for the relative mass point, whereas the analysis will be same for the absolute mass points.

Theorem 2.4: Consider the relative mass point position estimate dynamics (2.47) for each point mass m_i in the mass-spring-damper system. Let the control input $u_i(t)$ be given as

$$u_i(t) = -\sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\frac{(Z_i - Z_j)}{\left\| Z_i - Z_j \right\|} - K_v \dot{Z}_i \tag{2.48}$$

and the parameter tuning give as

$$\dot{\hat{\varphi}} = -\Gamma W^T \hat{M}_i^{-1} \dot{Z}_i \tag{2.49}$$

Then the relative mass point position reaches steady-state values in the sense that the potential function V is minimized.

Proof: Define the Lyapunov function

$$L_1 = L + \frac{1}{2}(\tilde{\varphi}^T \Gamma^{-1} \tilde{\varphi}) \tag{2.50}$$

where $\Gamma = diag(\lambda_1, \lambda_2, \lambda_3, \ldots\ldots., \lambda_q)$ and $\lambda_i's$ are the positive scalar constants. Differentiating (2.50), yields

$$\dot{L}_1 = \dot{L} + \tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{2.51}$$

and on substitution of $\dot{L}$ form (2.17) gives

$$\dot{L}_1 = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \dot{r}_{ij} + \sum_{i=1}^{N} \dot{Z}_i^T \ddot{Z}_i + \tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{2.52}$$

which on further substitution for $\ddot{Z}_i$ from (2.47), yields

$$\dot{L}_1 = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \dot{r}_{ij} + \sum_{i=1}^{N} \dot{Z}_i^T u - \sum_{i=1}^{N} \dot{Z}_i^T \hat{M}_i^{-1} \tilde{\varphi} + \tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{2.53}$$

From (2.53), it is clearly seen that the first two terms on the right hand side of the equation the Lyapunov derivative function given in (2.19), which is rewritten in the form given as,

$$\dot{L}_1 = \dot{L} + \sum_{i=1}^{N} \dot{Z}_i^T \hat{M}_i^{-1} \tilde{\varphi} + \tilde{\varphi}^T \Gamma^{-1} \dot{\tilde{\varphi}} \tag{2.54}$$

which on further simplification can be rewritten as

$$\dot{L}_1 = \dot{L} - \tilde{\varphi}^T (W^T \hat{M}_i^{-1} \dot{Z}_i - \Gamma^{-1} \dot{\tilde{\varphi}}) \tag{2.55}$$

From Theorem 2.1, it has been shown that $\dot{L}$ is negative semi-definite, therefore in order for L_1 to be negative semi-definite

$$\dot{\tilde{\varphi}} = \Gamma W^T \hat{M}_i^{-1} \dot{Z}_i \tag{2.56}$$

which results in a tuning law given as,

$$\dot{\hat{\varphi}} = -\Gamma W^T \hat{M}_i^{-1} \dot{X}_i \tag{2.57}$$

since $\dot{\varphi}$ is equal to zero because the unknown parameters are constant. On substitution of (2.57) into (2.55), yields

$$\dot{L}_1 = \dot{L} \tag{2.58}$$

and it has been shown that $u_i \to 0$ as $t \to \infty$ and hence $\dot{Z}_i \to 0$, therefore $\dot{L}_1 \to 0$ as $t \to \infty$, which means the system is SISL.

2.4 Simulation Results

In this section, the results of morphing the flexible structure are presented with the proposed control law using the potential field method. Figure 2.5 shows the result when the surface is morphed to have the desired shape given by red star points, whereas the actual shape is given by the blue circle points.

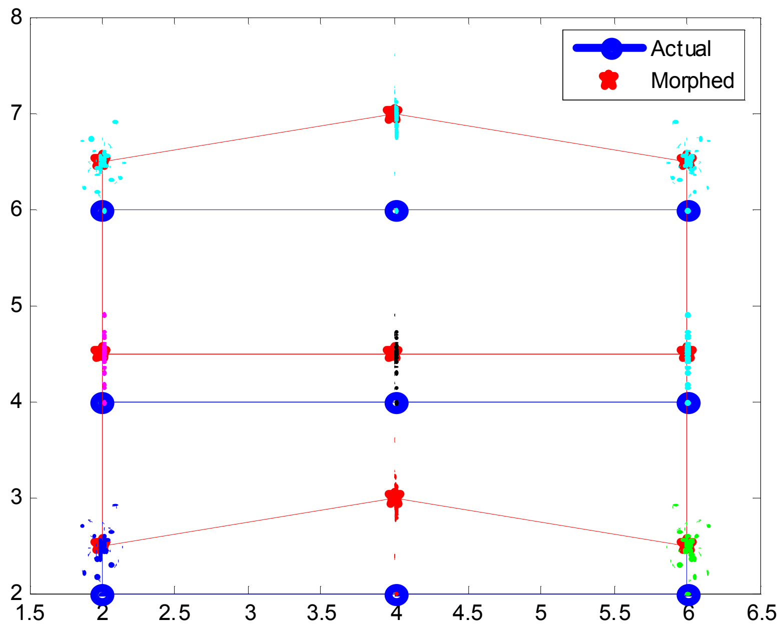


Figure2.5 Shape morphing result where red star shows the desired position and blue circle is the actual position of the mass points

Figure 2.6 shows the response of the position of the mass points under the influence of the control law given in (2.40) and (2.41).

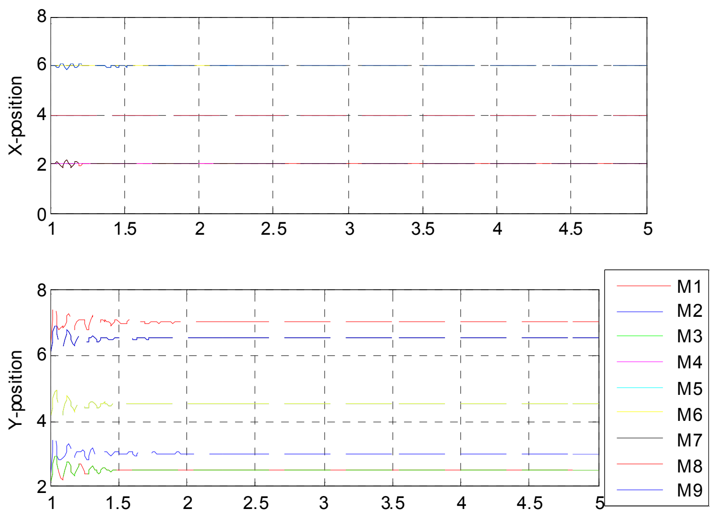


Figure 2.6 Response of the *x-y* positions of the mass points under the influence of the control law designed using the potential field method

2.5 Conclusion

In this chapter of the research work, an efficient algorithm for shape control of flexible structure is given. With the simulation results presented in the chapter the effectiveness of the algorithm is highlighted. The result in this chapter lays out the foundation for shape control of flexible structure. It becomes very important to select/place the microactuators for optimal shape control of flexible structure, since it aids in reducing the weight of the flexible structure and low power consumption. The following chapter presents the novel algorithm developed for the optimal selection/placement of the microactuators on the flexible structure.

CHAPTER 3

OPTIMAL SELECTION/PLACEMENT OF MICROACTUATORS FOR DISTRIBUTED CONTROL USING NEURAL NETWORKS

In the previous chapter, a control law was designed to achieve shape morphing where all the actuators are controlled. Practically, it is not viable to control all the actuators since energy is one of the major constraints, therefore it becomes necessary to develop an algorithm which can select the relevant actuators that need to be controlled in order to achieve the desired shape of the structure with minimum error. This chapter proposes an algorithm which selects the actuators which need to be controlled using the concept of neural networks. The overall block diagram of the scheme is shown in Figure 3.1

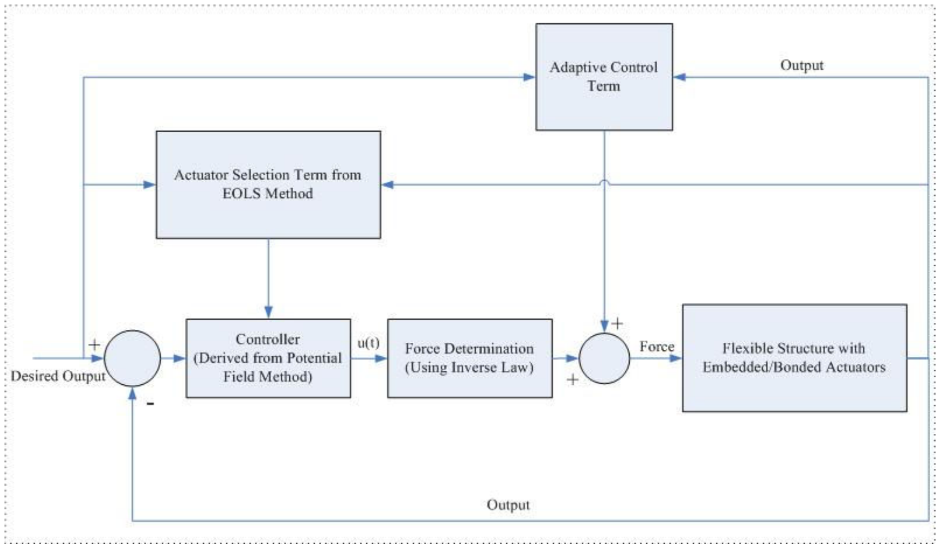

Adaptive Control Term
Output
Actuator Selection Term from EOLS Method
Controller (Derived from Potential Field Method)
u(t)
Force Determination (Using Inverse Law)
+
Force
Flexible Structure with Embedded/Bonded Actuators
+
Desired Output
-
+
Output

Figure 3.1 Block diagram with control law using potential field and actuator selection/placement

This chapter is organized in the following way: Section 3.1 discusses the problem to be dealt when morphing the flexible structure. Section 3.2 describes the system architecture which includes the description of the RBF NN, choice of the rdial basis function for morphing and also the modified RBF NN to be used for actual surface morphing. Section 3.3 describes the standard OLS method and how it is modified to determine the actuator positions through a novel method called as EOLS method proposed in this research. Section 3.4 presents the simulation results of EOLS method. Finally, we conclude the chapter in section 3.5.

3.1 Problem Formulation

RBF NN is a nonlinear interpolation tool, used to map the input vector space to the output vector space. Since the numbers of neurons required in the RBF NN are decided by the number and positions of centers, the design parameters in the RBF NN are the number of centers, positions of centers and the output weights. An RBF NN could be designed to map a 2D field from an arbitrary shape to a desired shape, by appropriately designing the NN parameters. In this research, the center positions of the NN represent the optimal position of the microactuators and the number of centers represents the actual number of microactuators to be used for morphing the flexible structure.

For example, to morph a 2D grid consisting of 100 points, to a desired shape as shown in Figure 3.2, only 10 points (actuators) can be actuated to bring the grid to the desired shape. These actuator positions and the number of actuators (10 points) are decided by the RBF NN. The actuator positions represented can be viewed as the center positions of the RBF NN. Therefore the problem of determining the actuator selection/placement becomes the question of how to best determine the center positions of the RBF NNs to provide the required deformation.

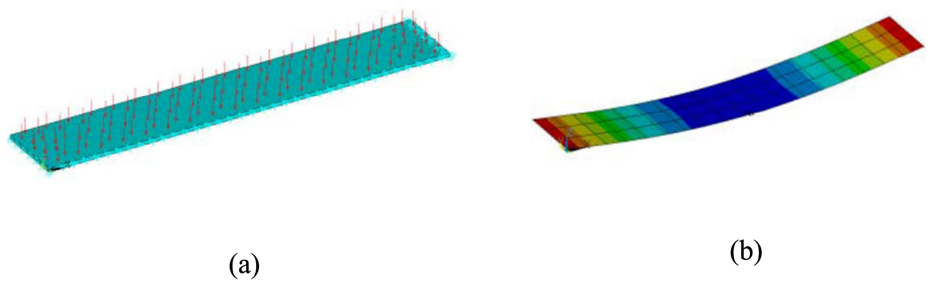


Figure 3.2 Morphing of a flexible structure (a) original shape of the plate (b) deformed shape of the plate

Various methods for center selection of RBF NN exist. One of the methods is the clustering method that consists of input-output clustering [24 - 26]. These input-output clustering algorithms involve methods such as k-means, recursive k-means, mean tracking clustering. These methods result in the selection of large number of hidden neuron centers. Another method based on genetic algorithm is proposed in [27], though it produces a smaller size of hidden centers and has a large computational cost.

One of the most commonly used methods to select the centers is the orthogonal least square (OLS) algorithm. The OLS algorithm is used as a tool to select both the center positions of the RBF and output weight matrix. One of the basic advantages of this method is that it avoids the inversion of matrices. In this research the standard OLS algorithm is modified to include the displacement and rotation effects, which are needed to fully describe the morphing of 2D flexible structures. This requires the determination of additional parameters which cannot be provided by the OLS method. To accomplish the complete deformation of the flexible structures, a novel method called Extended Orthogonal Least Square (EOLS) is introduced.

3.2 System Description

In this section an RBF NN is presented to estimate the number and positions of the microactuators. Before a new model of RBF NN is presented, the standard RBF NN in outlined to get familiarized with the changes to be made in the modified RBF NN.

This section also lays out the reason to choose Thin-Plate Spline as the radial basis function. The modified RBF NN developed here, forms the fundamental step to be used in the new proposed Extended Orthogonal Least Square (EOLS) method in the following section.

3.2.1. RBF Network Modeling

A standard RBF network performs a non-linear mapping between the input and the output vector space. It is used as an interpolation tool in the output vector space. A typical RBF architecture is shown in Figure 3.3.

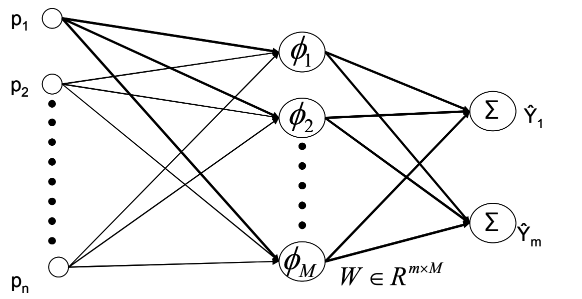


Figure 3.3 Radial basis Function Neural Network

The RBF mapping is given via the transformation

$$y_i = f_i(p) = \sum_{k=1}^{M} w_{ik}\phi(\|p - c_k\|) \tag{3.1}$$

where $p \in \Re^{n\times 2}$ is the input to the network, M is the number of neurons in the network, c_k is the RBF center, $\|.\|_2$ is the euclidean distance, w_i is the neural network weight, and $\phi(.)$ is the radial basis function. There are several choices of the RBF ($\phi(.)$) in the network. Some of the typical choices of RBF are

1) Cubic spline ($\phi(x) = x^3$),

2) Gaussian Function ($\phi(x) = \exp(-\frac{x^2}{\sigma^2})$), where σ is the spread parameter,

3) Thin Plate spline ($\phi(x) = x^2 \ln(x)$).

The choice of RBF is an important factor in setting up the mapping function. Therefore, the following discussion sets up the foundation on choosing the RBF.

3.2.2. Choice of Radial Basis Function

In this research Thin Plate Spline (TPS) is used as an interpolant function and forms the basis of the NN. The choice of TPS function lies on the fact that the TPS function represents the bending motion of a 2D flexible structure. The bending motion of a 2 dimensional structure is governed by a biharmonic equation given as

$$\Delta^2 g = 0 \tag{3.2}$$

where Δ is the Laplacian operator and g is the function to be solved. The solution of g is obtained by rewriting (3.2) as

$$\Delta(\Delta g) = 0 \tag{3.3}$$

where $\Delta = \nabla^2$ and ∇ is the gradient operator given as

$$\nabla = \hat{r}\frac{\partial}{\partial r} + \hat{\theta}\frac{\partial}{\partial \theta} + \hat{z}\frac{\partial}{\partial z} \tag{3.4}$$

Considering only (r, θ) and substituting (3.4) in (3.3), yields,

$$\Delta g = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}g\right) + \frac{1}{r^2}\left(\frac{\partial^2}{\partial \theta^2}g\right) \tag{3.5}$$

For the thin-plate spline the function g is only dependent on the (x, y) co-ordinates given as $g = f(r) = f(\|(x, y)\|_2)$. Therefore, $\frac{\partial}{\partial \theta}$ is zero and equation (3.5) is written as

$$\Delta g = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}g\right) \tag{3.6}$$

Now, integrating equation (3.6) twice, yields

$$g = C_1 \ln(r) + C_2 \tag{3.7}$$

To solve the second Laplacian operator, equation (3.7) is again integrated twice yielding

$$g = \frac{C_1}{4} r^2 \ln(r) - \left(\frac{C_1}{4} - \frac{C_2}{4} \right) r^2 + C_3 \ln(r) + C_4 \tag{3.8}$$

where C_1, C_2, C_3 and C_4 are constants. The equation (3.8) could be approximated to get the TPS deformation given as

$$g = r^2 \ln(r) \tag{3.9}$$

The RBF network explained in Section 3.1, is capable of estimating the actuator positions whenever there is bending, stretching or compression of a flexible structure. In order to estimate the actuator positions and unknown RBF parameters under the influence of tilting or just linear transformation together with bending, stretching or compression, the RBF network needs to be modified, as explained in the following section.

3.2.3. Modified RBF Network Modeling

Since the RBF network presented in Section 3.1 represents only the bending energy of the flexible structure, it is not sufficient to determine the actuator positions and the unknown RBF NN parameters under the effect of tilting and transformation of the flexible structure. When a physical flexible structure is merely tilted or changed from level to oblique, it does not bend. In tilting, energy forces work against gravity and not against the elasticity. In order to take into account the effect of tilting and transformation, the RBF network in equation (3.1) has to be modified to include an extra term known as the affine transformation. Therefore the new RBF model is given as

$$y_i = f_i(p) = \lambda(p) + \sum_{k=1}^{M} w_{ik}\phi(\|p - c_k\|) \tag{3.10}$$

where $\lambda(p)$ is the affine transformation term given as

$$\lambda(p) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = AX + B \tag{3.11}$$

The architecture of the modified RBF NN is shown in Figure 3.4.

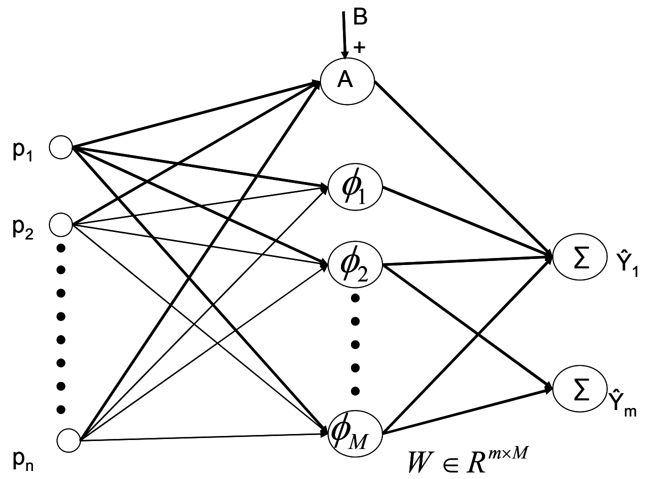


Figure 3.4 Modified Radial Basis Function Neural Network

The RBF given in Section 3.2 takes into account he effect of the normal and the in-plane transformation of the flexible structure. One of the concerns in RBF design is to select the centers appropriately from the data set. Proper selections of center positions yields in accurate performance of the interpolation and also in reduce size of the RBF network, since large RBF networks results in computational complexity and numerical ill-conditioning. One of the most common methods to select the center position is orthogonal least square (OLS) method mentioned in [28, 51]. A brief introduction to the OLS method is presented here in the following section in order to justify the use of

Extended Orthogonal Least Square (EOLS) method, which is the main contribution of the paper.

3.3 Extended Orthogonal Least Square Method

As mentioned earlier in Section 3.1, the actuator number and position basically represents the center positions of the RBF NN. Therefore the problem of actuator placement determination becomes the question of how to determine the center positions of the RBF NN. Therefore in this section, the method to estimate the center position using a novel Extended Orthogonal Least Square (EOLS) method is presented. is laid out.

Before presenting the EOLS method, the standard OLS method is briefly explained, to get a better understanding of EOLS method and how the actuator number and positions are determined.

3.3.1. Standard Orthogonal Least Square Method

In the standard OLS method, the RBF network is seen as a linear regression model given as

$$\widetilde{d} = P\theta + e \tag{3.12}$$

where $\widetilde{d} \in \Re^{N\times 2}$ is the set containing the output measurement data and $P \in \Re^{N\times M}$ is the radial basis function matrix (regression matrix) given as

$$P = \begin{bmatrix} 0 & \phi(p_1,c_2) & . & . & \phi(p_1,c_M) \\ \phi(p_2,c_1) & 0 & . & . & \phi(p_2,c_M) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi(p_N,c_1) & \phi(p_N,c_2) & . & . & 0 \end{bmatrix} \tag{3.13}$$

$\theta \in \Re^{M \times 2}$ is the weight matrix given as,

$$\theta = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ . & . \\ . & . \\ w_{M1} & w_{M2} \end{bmatrix} \tag{3.14}$$

The standard OLS method allows the selection of RBF centers so that an adequate RBF network size could be obtained. In the OLS method, as mentioned in [28, 51], the problem of selecting a center is equivalent to the selection of significant regressors from a given data set. Each regressor in the data set forms a set of basis vector. At each step of the OLS method, a new center is selected by decomposing the regression matrix given by P, into a set of orthogonal basis vectors. On decomposition of the regression matrix (P) matrix, the calculation of each basis vector towards the maximization of the desired output variance is obtained.

Using the Gram-Schmidt orthogonalization method, the regression matrix P is decomposed as

$$P = WA \tag{3.15}$$

where $W \in \Re^{N \times M}$ is the orthogonal matrix and $A \in \Re^{M \times M}$ is an upper triangular matrix. Since the vector space spanned by the set of orthogonal basis vectors w_i is the same vector space spanned by the columns of the regression matrix P, the estimated model given in (3.12) can be written as

$$\hat{d} = WA\hat{\theta} = W\hat{g} \tag{3.16}$$

where $\hat{g} = A\hat{\theta}$. To estimate $\hat{g}$, the cost function is defined as

$$J_{\hat{g}} = \frac{1}{2}(\tilde{d} - W\hat{g})^T(\tilde{d} - W\hat{g}) \tag{3.17}$$

Now, using the necessary condition to achieve the global minimum of the quadratic function, $\hat{g}$ could be obtained as

$$\hat{g} = (W^T W)^{-1} W^T \tilde{d} \tag{3.18}$$

In order to choose the center position, the output energy is analyzed, which is given as

$$\begin{aligned} d^T d &= g^T W^T W g + e^T e \\ &= \sum_{i=1}^{M} g_i^{\,2} w_i^{\,T} w_i + e^T e \end{aligned} \tag{3.19}$$

where $\hat{g}_i$ is the i^{th} vector of $\hat{g}$, given as

$$\hat{g}_i = \frac{w_i^{\,T}\tilde{d}}{w_i^{\,T} w_i} \tag{3.20}$$

Since there is a one-to-one correspondence between the center positions (c_i) and the elements of the regressor vector $\hat{g}_i$, each term in $\sum_{i=1}^{M} g_i^{\,2} w_i^{\,T} w_i$ represents an increment in the energy due to the inclusion of the i^{th} center (c_i). Therefore, an error reduction ratio is defined to select a center (c_i) given as

$$[err]_i = \frac{g_i^{\,2} w_i^{\,T} w_i}{d^T d} \tag{3.21}$$

The selection of the RBF NN center position is performed in a forward regression manner. Therefore, at every step of the OLS method, an RBF center is selected so that the error reduction ratio is maximized. For the details on the step-by-step approach, please refer to [28, 51].

3.3.2. Extended Orthogonal Least Square Method

The EOLS method is designed to estimate the actuator positions, unknown RBF network weights and the unknown terms corresponding to the affine transformation. An outline of the EOLS method is given here together with the mathematical analysis. The model to be considered is an extension of the OLS model given in (3.12), which is given as

$$\widetilde{d} = P\theta + QB + e \tag{3.22}$$

where $\widetilde{d} \in \Re^{N\times 2}$ is the set containing the output measurement data, $P \in \Re^{N\times N}$ and $\theta \in \Re^{N\times 2}$ are same as defined in (3.13) and (3.14) respectively, $Q \in \Re^{N\times 3}$ is the input training set given as

$$Q = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ . & . & . \\ . & . & . \\ x_N & y_N & 1 \end{bmatrix} \tag{3.23}$$

and $B \in \Re^{3\times 2}$ is the matrix with affine transformation terms given as,

$$B = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ b_1 & b_2 \end{bmatrix} \tag{3.24}$$

The method of estimating the unknown weight matrix and the affine transformation term is divided into two steps. The two step procedure is based on the fact that the flexible structure has different effects during tilting and bending. The first step of the ELOS takes into account the effect of tilting only, whereas the second step takes into account both the tilting and bending of the flexible structure.

3.3.3. Determination of Affine Transformation Term

The affine transformation term is responsible for the motion corresponding to the tilting of the flexible structure. Since during the tilting of the flexible structure, the effect of the nonlinear terms in (3.22) is negligible, the model in (3.22) could be approximated as

$$\hat{d} = Q\hat{B} \tag{3.25}$$

In order to include the information of the RBF NN to update the affine transformation term, the performance index is defined as

$$\begin{aligned} J_{\hat{B}} &= \frac{1}{2} e_{\hat{B}}{}^{T} e_{\hat{B}} + \frac{1}{2} e_{\hat{B}}{}^{T} \alpha P_1^{-1} e_{\hat{B}} \\ &= \frac{1}{2} e_{\hat{B}}{}^{T} (I + \alpha P_1^{-1}) e_{\hat{B}} \end{aligned} \tag{3.26}$$

where $P_1 \in \Re^{M\times M}$ is the regression matrix formed from the selected center positions obtained from OLS method and given as

$$P_1 = \begin{bmatrix} 0 & \phi(p_1, c_2) & . & . & \phi(p_1, c_M) \\ \phi(p_2, c_1) & 0 & . & . & \phi(p_2, c_M) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi(p_M, c_1) & \phi(p_M, c_2) & . & . & 0 \end{bmatrix} \tag{3.27}$$

and α is the tuning parameter chosen as $\alpha = \frac{1}{\max(\lambda(P_1))}$, $\lambda's$ are the eigenvalues of the regression matrix (P_1), $e_{\hat{B}} = \tilde{d} - \hat{d} = \tilde{d} - Q\hat{B}$ is the error between the true and the estimated output and I is an identity matrix of dimension $\Re^{N\times N}$

Taking the gradient of (3.16) with respect to the unknown affine transformation term ($\hat{B}$), yields

$$\frac{\partial J_{\hat{B}}}{\partial \hat{B}} = \frac{1}{2}(-2\tilde{d}^T (I + \alpha P_1^{-1})Q + 2\hat{B}^T Q^T (I + \alpha P_1^{-1})Q) \tag{3.28}$$

Equating (3.28) to zeros by satisfying the necessary condition of global minimum and then solving for $\hat{B}$, yields

$$\hat{B} = (Q^T (I + \alpha P_1^{-1})Q)^{-1} Q^T (I + \alpha P_1^{-1})\tilde{d} \tag{3.28}$$

The reason for selecting the cost function which includes the regression matrix term is to update the affine transformation term each time a new center is selected, since there is one-to-one correspondence between the center position and the regression matrix. Since P is not a positive definite matrix, α is introduced to make the term $(I + \alpha P_1^{-1})$ positive definite.

3.3.4. EOLS Algorithm Description

In this section, a complete outline of the EOLS method is tabulated in order to determine the RBF neural network parameters and obtain the interpolant function given in (10). The EOLS algorithm is tabulated in table 3.1, given below.

Table 3.1 EOLS Algorithm

Step 1: For $k = 1$, $1 \le i \le N$ Initialize $w_1^{(i)} = p_i$ Compute $g_k^{(i)} = \frac{(w_1^{(i)})^T (d)}{(w_i^i)^T w_1^i}$ Compute the error reduction ratio of the i^{th} center as

$$[err]_1^i = \frac{(g_1^{(i)})^2 (w_1^i)^T w_1^i}{(d)^T (d)}$$

Select the center i_1 for which the $[err]_1^i$ is maximum and then select $w_1 = p_{i1}$

at the center $c_1 = c_{i1}$

Step 2: For $k \geq 2$, $1 \leq i \leq N$ & $i \neq i_1$, $i \neq i_2$, ……, $i \neq i_{k-1}$

Compute $\alpha_{jk}^i = \dfrac{w_j^T p_i}{(w_j^T w_j)}$, $1 \leq j < k$

$$w_k^{(i)} = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^i w_j$$

$$g_k^{(i)} = \frac{(w_k^{(i)})^T (d)}{(w_k^i)^T w_k^i}$$

$$[err]_k^i = \frac{(g_k^{(i)})^2 (w_k^i)^T w_k^i}{(d)^T (d)}$$

Select the center i_k for which the $[err]_k^i$ is maximum

and then select $w_k = w_k^{(ik)} = p_{ik} - \sum_{j=1}^{k-1} \alpha_{jk}^i w_j$

Step 3: After selection of the center positions, form P_1 given by (27). Use (29) to determine $\hat{B}$ given as

$$\hat{B} = (Q^T (I + \alpha P_1^{-1}) Q)^{-1} Q^T (I + \alpha P_1^{-1}) \tilde{d}$$

3.4 EOLS Algorithm Application Examples

In this section, the results of morphing the flexible structure are presented and compared with the principal wrap method presented in [46]. Figure 3.5 and 3.6 shows the result of morphing with 10 and 15 actuators respectively, selected using the EOLS method. Figure 3.7 plots the error and compares the error between the EOLS method and the Principal Wrap (PW) method. From the plot it is clearly seen that the EOLS method performs efficiently and reaches the desired shape more accurately. EOLS method gives the flexibility of choosing the actuator position, whereas for the PW method the actuator positions have to be known *a priori*.

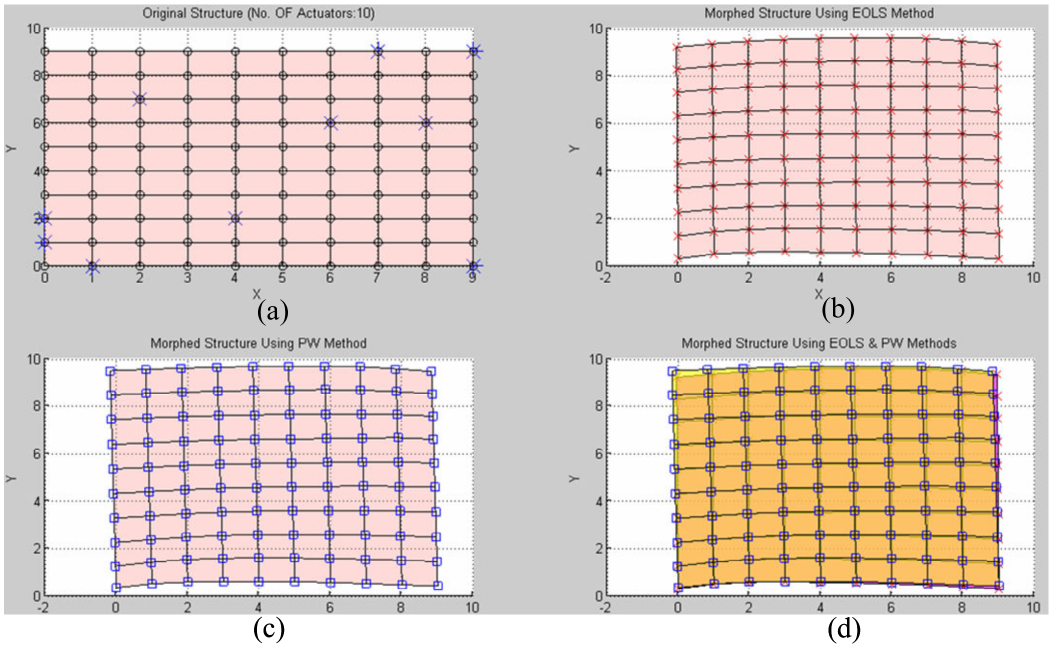


Figure 3.5 Morphing result using 10 Actuators (a) original grid points (b) morphing using EOLS Method (c) morphing using PW method (d) comparison of EOLS and PW method

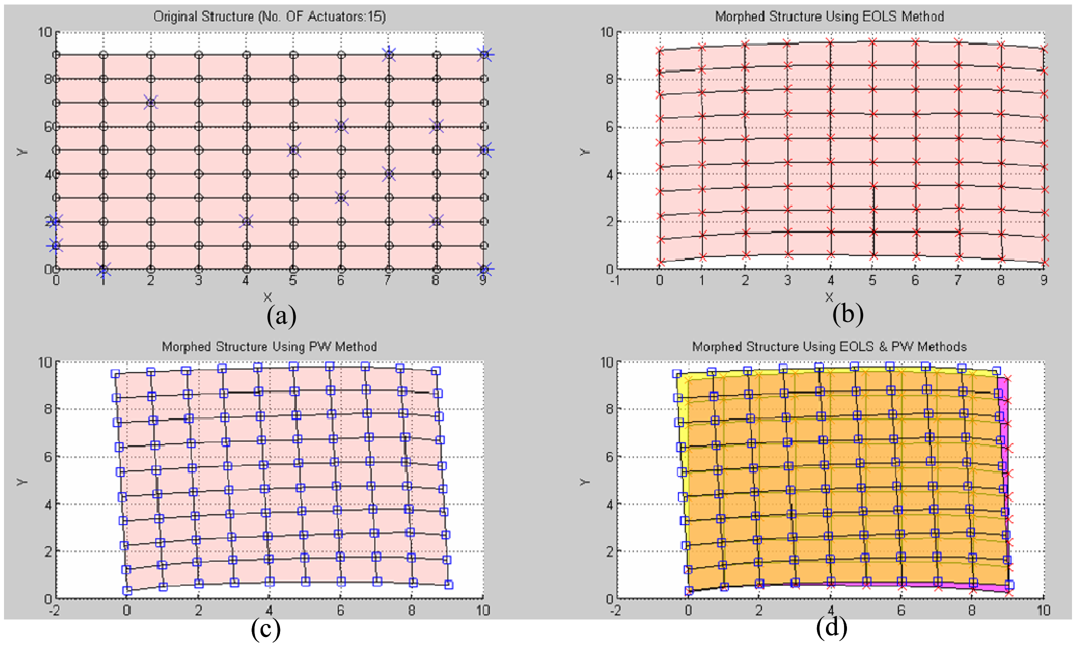


Figure 3.6 Morphing result using 15 Actuators (a) original grid points (b) morphing using EOLS Method (c) morphing using PW method (d) comparison of EOLS and PW method

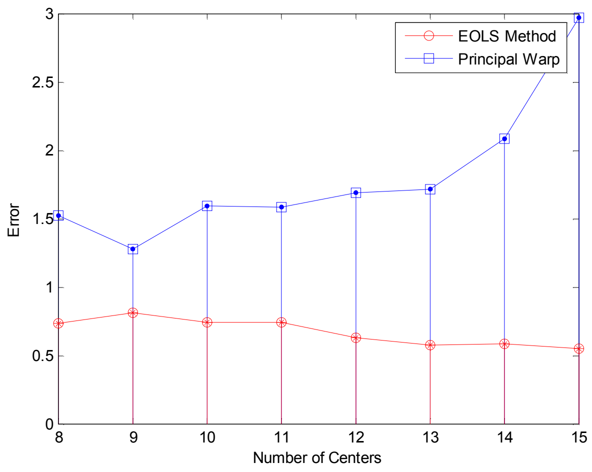


Figure 3.7 Error plot comparing the morphing error between EOLS method and Principal Wrap Method

3.5 Conclusion

An efficient algorithm for obtaining the optimal actuator placement method is presented, together with the mathematical analysis. The extended orthogonal least square (EOLS) method developed takes into account the factors of tilting, linear transformation and bending of a flexible structure. The simulation results obtained validates the high efficiency of the EOLS method by comparing it with the principal warp method. More work is being pursued to include the dynamics and the stiffness of the physical flexible structure. This chapter completes the overall control scheme given in Figure 3.1, where the optimal selection/placement of the microactuators is done through the EOLS algorithm and the control input is derived as given in Chapter 2.

Until now we have developed algorithms for shape control of flexible structure and optimal selection/placement of the microactuators on the flexible structure. It becomes imperative even to obtain the desired shape for the flexible structure, therefore the next chapter presents the expression classification algorithm to obtain the desired shape for the flexible structure. A twp-stage neural network algorithm is developed for the expression classification procedure.

CHAPTER 4

TWO STAGE NEURAL NETWORK FOR EXPRESSION CLASSIFICATION

This chapter is organized in the following manner, Section 4.1 gives an overview of the system architecture and highlights the steps necessary to carry out the expression classification process. Section 4.2 discusses the feature (landmark grid) extraction and the normalization process. Section 4.3 presents the first stage of the neural network to estimate the principal components for dimension reduction. Section 4.4 discusses the second stage of neural network for expression classification. In Section 4.5, experimental results are presented on the image database obtained from Yale University [42]. The algorithm performance is presented via a confusion matrix and receiver operating characteristic (ROC) curves.

4.1 System Architecture

This section presents the system architecture being used for the face expression recognition methodology. The key steps involved in the methodology are 1) Normalization 2) Landmark grid extraction 3) Estimation of principal components using neural network for the landmark grids 4) Expression classification using a neural network.

The proposed approach makes use of an architecture that consists of a two stage neural network to classify different face expressions. The scheme employed in the approach is shown in Figure 4.1, and is detailed in subsequent sections and outlined below.

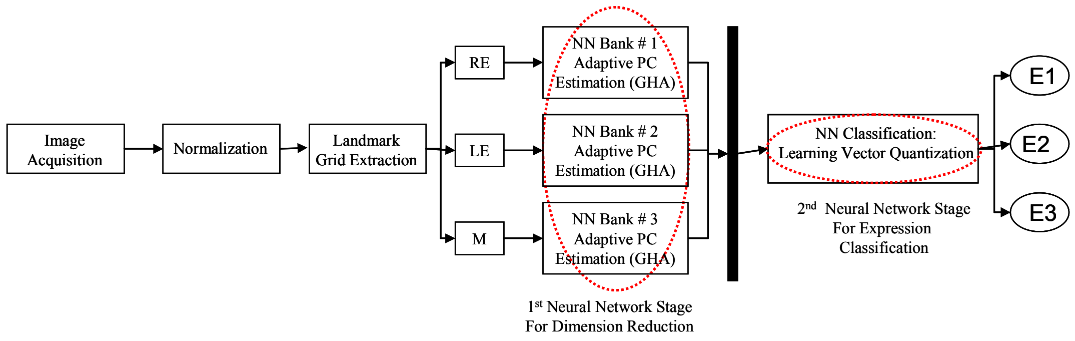


Image Acquisition
Normalization
Landmark Grid Extraction
RE
LE
M
NN Bank # 1 Adaptive PC Estimation (GHA)
NN Bank # 2 Adaptive PC Estimation (GHA)
NN Bank # 3 Adaptive PC Estimation (GHA)
1st Neural Network Stage For Dimension Reduction
NN Classification: Learning Vector Quantization
2nd Neural Network Stage For Expression Classification
E1
E2
E3

Figure 4.1 System Architecture

The second step in the approach is data pre-processing which involves the normalization of images in order to make it independent of factors like lighting, position and scale. Details involving image normalization are given in Section 4.2.1.

The first step of the methodology involves the extraction of landmark points and grids. The landmark points are used in a normalization procedure, whereas the landmark grids are used as inputs to the first stage of the neural network. Since most of the face image consists of skin, it becomes redundant to input the entire face image. Therefore, only the grids around the key feature points (i.e. eyes, mouth) are considered as input to the system. This step reduces the number of inputs required for the first stage of the neural network. The method used to obtain the grid around the landmark points is explained in section 4.2.1.

The third step given in the proposed approach involves the dimension reduction of the image using Principal Component Analysis (PCA) by the first stage of the neural network. This stage consists of a bank of three neural networks for three landmark grids. Each neural network bank is based on a Gradient Hebbian Learning rule, which is used to estimate the principal components of the landmark grids instead of using an original matrix algebra method. This feature gives us the flexibility of updating the principal components online as new faces comes in. If one were to use the original matrix algebra method then this would require the task of obtaining the mean and co-variance of the image set, which must be performed offline. Once the principal components are estimated, the projection of the actual landmark grid is taken to reduce the dimension of the data. Details on the use of a neural network to estimate the principal component and hence obtain a reduced dimension dataset are given in section 4.3.

The last and the main step of classifying different expressions are obtained from the second stage of neural network. The second stage of neural network is based on supervised competitive learning. Supervised Learning Vector Quantization (LVQ) network is trained using kohonen learning rule to classify various expressions. The use of LVQ network to obtain classification is explained in section 4.4.

4.2 Data Preprocessing

One of the important aspects in facial expression technique has been data preprocessing. A lot of research has been done to optimize the amount of data to be processed in order to classify various expressions. Computation time for classification increases if the whole face data is considered. To encounter this problem, the image data is processed to reduce the dimension of the data such that only relevant information is considered for classification. The reduced dimension data is obtained using principal component analysis method. Instead of using the standard matrix algebra based principal component analysis method, the neural network technique is used to estimate the principal components.

Before the details on the use of neural network is presented, the methods pertaining to making the face image light, pose and scale independent and landmark grid extraction are presented. This section begins with the steps necessary to carry out the normalization and landmark grid extraction process.

This section presents the step necessary to carry out the landmark grid extraction and normalization process. Key attribute selection procedure is performed on the images to extract the landmark points and grids. Images taken under different conditions of lighting, pose, and scale effect the performance of the classification algorithm. In order to avoid these effects the images are normalized with respect lighting, pose and scale.

4.2.1. Normalization to Avoid Lighting Effects

The images have to be consistent in terms of lighting, scale, pose and image size for them to be classified correctly. Variations in any of these factors could lead into incorrect expression classification. To avoid the effect of inconsistent lighting, the images are normalized with respect to the intensity values using histogram equalization method. Histogram equalization method is a non-linear mapping which assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities. For more on the histogram equalization method, see [43].

4.2.2. Landmark Point Extraction

In this section, the method to obtain the landmark points (eyes, mouth forehead) and grids around them is explained. Using standard image processing techniques such as horizontal, vertical gradient and projection methods, circular Hough transform [44] and probability distribution of skin color [44], the position of eyes and mouth are obtained. The position of the forehead point is obtained geometrically by locating it in between the eyes. Complete formulation of this procedure is given in [44, 45]. The landmark feature points obtained using this procedure is shown in Figure 4.2

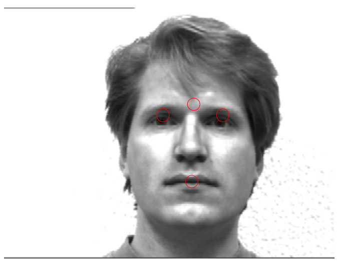


Figure 4.2 Image with landmark points

4.2.3. Normalization to Avoid Scale and Pose Inconsistencies

The landmark grid points obtained from the above section are used to avoid scale and pose inconsistencies in the image. Variation in scale leads to inconsistent head size and mouth expression. Scale invariance and constant image size are achieved by interpolating the face images to a standard position. The interpolation is performed using the Thin Plate Spline (TPS) method defined in [46], which takes into account the effect of transition, scaling, and rotation. To begin with the TPS transformation, four landmark points (eyes, mouth, forehead) on the face image are determined as mentioned above, and are then mapped to a standard pixel position of these landmark points of a 2D face model, which are pre-allocated. TPS transformation helps in achieving exact geometric orientation so that the key facial features are located in the same region of every image, which is useful in landmark grid extraction process.

The TPS transformation is given as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \sum_{i=1}^{n} w_i U(\|P_i - (x, y)\|) \tag{4.1}$$

where $\begin{bmatrix} x' \\ y' \end{bmatrix}$ is the mapping of the point $\begin{bmatrix} x \\ y \end{bmatrix}$, A, B and w_i are of size $(2,2)$, $(2,1)$, $(2,n)$, respectively. P_i is the position of the landmark points, n is the number of landmark points which in our case is 4 and $U(r) = r^2 \log(r^2)$ where $r^2 = x^2 + y^2$. Initially A, B, w_i are unknowns, whose co-efficient are determined by knowing the mapping of four landmark points to a standard pixel position. The method to determine the unknowns is explained in [46]. Once these unknowns are determined, the other pixel positions whose mapping is unknown are mapped using the transformation mentioned in (1). The result of TPS transformation applied on a face image is shown in Figure 4.3. The green squares shown in fig. 4.3 (a) are the standard position of the landmark points, which are pre-determined and the red circles are the feature (landmark) points detected using landmark point and grid extraction method described in 4.2.1. The results of applying the TPS interpolation method are shown in fig. 4.3 (a) and 4.3 (b).

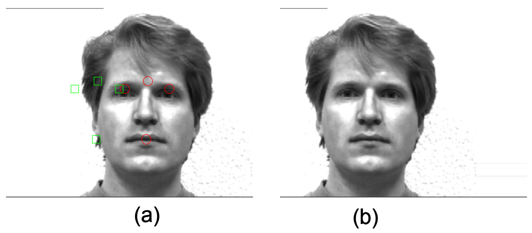


Figure 4.3 Thin Plate Spline Transformation (a) original image (b) after TPS transformation

4.2.4. Landmark Grid Extraction

Once the landmark feature points are located and normalized to avoid scale and pose inconsistencies, a grid of pixel size $(35,50)$, $(35,50)$, $(30,80)$ is formed around the right eye, left eye and mouth region, respectively. This grid size was selected by hit and trial for the image data obtained from Yale University [42]. The result of this process is shown in Figure 4.4. The grid size may vary depending upon what image size is being considered for expression analysis.

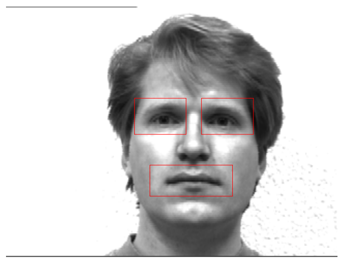


Figure 4.4 Image with landmark grids

4.3 First Stage Neural Network for Principal Component Estimation

The normalization and landmark grid extraction process yields the data to be analyzed for principal component estimation. The estimation of principal components is done by the Hebbian learning rule neural network. The details on the use of neural network to estimate principal components are explained further in this section. Before the neural network estimation process is mentioned, a brief introduction to the principal component method is given.

Principal component analysis (PCA) is one of the oldest and well-known techniques employed in image processing. PCA also known as Karhunen-Loeve Transform (KLT) is used for dimensionality reduction. PCA is a statistical method used for data compression by determining a linear transformation matrix $W \in \Re^{m \times n}$ $(m < n)$, which compresses the data $X \in \Re^{n \times 1}$ to yield a lower dimension data given as

$$y = WX \tag{4.2}$$

where $y \in R^{m \times 1}$

The concept of PCA is to reduce the number of features representing a data by discarding the ones which have small variance and retains only those terms that have large variance. PCA method uses standard matrix algebra method in calculating the eigenvectors of the co-variance matrix formed by analyzing the image data. Only those eigenvectors are chosen which gives the maximum information about the data. These chosen eigenvectors form the W matrix. The details on obtaining the transformation matrix W are given in [48, 49]

To show the advantage of neural network over statistical based approaches, an example to calculate the principal components is shown using the landmark grids obtained above. The dimension of a landmark grid (for ex. landmark grid corresponding to mouth) extracted from the face image is $[P,Q]^T = [35,80]^T$ and the total number of landmark grids are $M = 21$ (i.e. total number of face images are $M = 21$). All the landmark grids extracted are converted into 1D vector given by τ_i, corresponding to the landmark grid i. The dimension of τ_i is $\{N,1\}$ where $N = P \times Q = 2800$. The mean of the landmark grids is obtained by

$$\psi = \frac{1}{M}\sum_{k=1}^{M}\tau_k = \frac{1}{M}\left[\tau_1 + \tau_2 + \tau_3 + ... + \tau_M\right] \tag{4.3}$$

and subtracting the mean from the original landmark grid yields the difference image given as

$$\phi_i = \tau_i - \psi \tag{4.4}$$

where $i = 1,2,3,....,M$. The difference landmark grids obtained in (4.4) are accumulated into a set given as

$$A = \left[\phi_1\ \phi_2\ \phi_3 ....\phi_M\right] \tag{4.5}$$

where the dimension of A is $[2800,21]$. Now, the co-variance matrix C is obtained for the difference images obtained in (5), given as

$$C = \frac{1}{M}\sum_{n=1}^{M}\phi_n\phi_n^{\ T} = AA^T \tag{4.6}$$

and then the eigenvectors of C are computed to form the basis of the transformation matrix W. Since the dimension of the co-variance matrix C is $\{N,N\}$, it becomes computationally inefficient to obtain the N eigenvectors. Since the number of possible eigenvectors that would yield the maximum information about the dataset, is equal to the number of face images in the training set, therefore only M eigenvectors are taken into account to form the W matrix. To compute M eigenvectors, following method is adopted, where a new matrix is constructed, given as

$$L = A^T A \tag{4.7}$$

The dimension of L is $\{M, M\}$ and let its eigenvalues be μ_i and eigenvectors be ϑ_i. Therefore we get

$$L\vartheta_i = \mu_i \vartheta_i \tag{4.8}$$

Substituting for L in (4.8), yields

$$A^T A \vartheta_i = \mu_i \vartheta_i \tag{4.9}$$

Multiplying (4.9) by A gives

$$AA^T A \vartheta_i = A\mu_i \vartheta_i \tag{4.10}$$

Since $C = AA^T$, therefore $A\vartheta_i$ are the eigenvectors of matrix C. Through this way, only M eigenvectors are computed instead of computing N eigenvectors. Following the process mentioned from equations (4.3)-(4.10), a transformation matrix W is constructed for dimension reduction.

The approach mentioned in equations (4.3)-(4.10) is based on batch processing, where all the information needs to be known *a priori* for the construction of W matrix. If a new face is added to the data set, a new matrix must be formed by adding that face to the others, and the entire computation for the PCs mentioned in (4.3)-(4.10) must be redone. In order to avoid this problem, neural networks are used in stage 1 to estimate the principal components of the image data, instead of the original matrix algebra method, based on batch computation. The advantage of using NN instead of batch matrix computation to calculate the principal component, is that it gives the flexibility of online adaptive reformulation. That is, the PCs are modified as each new face is presented to the NN based only on the new face data. By contrast, using batch computation methods, one must form a matrix that contains all the faces in the dada set.

Using NN for PC estimation does not require having all the information at once i.e. batch processing is avoided. As the images are assimilated one by one, the PC are updated sequentially. If a new face appears to the system architecture shown in Figure 4.1, its size is reduced by the PC estimated earlier after going through normalization and landmark grid extraction process. This reduced image size of the new face is used for expression classification and at the same time, the earlier estimated PC are updated with this new face image.

From Section 4.2.2, the landmark grids obtained in the form of matrix are converted into vectors. These landmark grid vectors are the inputs to this stage of neural network. Instead of combing all the different landmark grid vectors into one feature vector and using one neural network, the proposed algorithm uses three banks of neural network for each landmark grid as shown in Figure 4.5. The advantage of using different NN banks is that the system becomes robust, i.e. even if the left and right eye muscle movement are similar for two different expressions, the information obtained from the principal component of mouth region could be used to classify the expressions correctly.

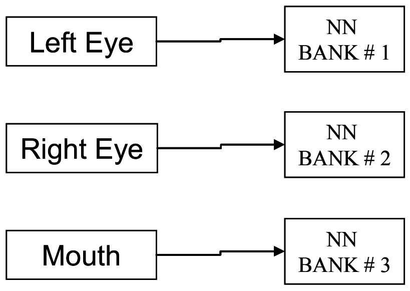


Figure 4.5 NN bank for each landmark vector

The Structure of neural network deployed in NN banks is shown in Figure. 4.6 and Sanger's rule based on Generalized Hebbian Algorithm (GHA) is used to estimate the principal component of the landmark region. The weight update equation for the neural network is given as

$$w_{ij}(k+1) = w_{ij}(k) + \mu(k)y_i(k)\left[x_j(k) - \sum_{h=1}^{i} w_{hj}(k)y_h(k)\right] \tag{4.11}$$

where $i = 1,2,.....,m$ and $j = 1,2,.....,n$, m is the number of principal components to be estimated and n is the number of input vectors (i.e. the length of the landmark grid vectors). The weight update equation in (4.11) is derived by defining a performance index given as

$$\begin{aligned} J_1(X) &= \frac{1}{2}(X - \hat{X})^T(X - \hat{X}) \\ &= \frac{1}{2}(X - W^T WX)^T(X - W^T WX) \end{aligned} \tag{4.12}$$

and minimizing the performance index defined above (4.12) with respect to W and using the steepest descent approach [50], the weight update equation is given as,

$$W(k+1) = W(k) + \mu\frac{\partial J_1(X)}{\partial W} \tag{4.13}$$

The weight update equation defined in (4.11) is written in the scalar form of (4.13), using the symmetricity of yy^T. In [51, 52], further variations of the update rules are given.

The reason of using GHA is that it extracts the m actual principal eigenvectors as those obtained from original matrix algebra method

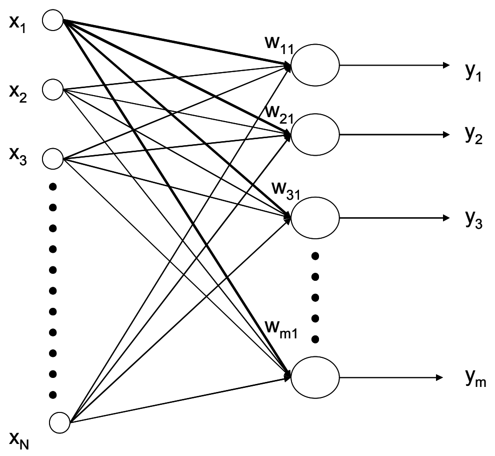


Figure 4.6 NN structure for estimating PC

4.4 Second Stage Neural Network for Expression Classification

In this section, a second stage neural network is presented to classify different expressions such as happy, sad, fear, anger, surprise, disgust or normal, where the output of the first stage neural network becomes the input to the second stage neural network. The classification step only requires distinguishing between different classes (happy, sad, fear, anger, surprise, disgust or neutral) by modeling the class boundaries. This type of learning is called as discriminative learning [53] and it does not require estimating the class feature densities. To achieve discriminative learning a supervised learning vector quantization network is used. LVQ is a hybrid network and uses a self organizing map approach. It is based on winner-take-all policy and uses training vector to distinguish the different categories of the input. Since LVQ network tends to have shorter training time then the backpropogation or RBF, and processing time being an important constraint in the approach mentioned, LVQ network proves to be advantageous over backpropogation and RBF. In reference [54], the performance of these networks have been compared

The supervised LVQ network is trained using standard kohonen learning rule. More details on this rule are given in [51]. In LVQ network each neuron in the first layer is assigned to a class and then each class is assigned to one neuron in the second layer. The second stage neural network is a supervised learning vector quantization network which is shown in Figure 4.7

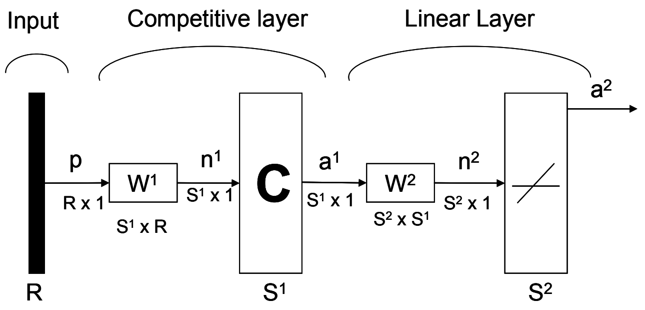


Figure 4.7 LVQ Network

The first layer of the LVQ network is the competitive layer followed by a second layer of linear network. The output of the first layer of network is given as

$$a^1 = compet(\|{}_i w^1 - p\|) \tag{4.14}$$

where p is the inputs with a dimension of $R \times 1$ elements and W^1 are the weights of the first layer of neural network. The output of the second layer of network is given as

$$a^2 = W^2 a^1 \tag{4.15}$$

which has S^2 number of neurons, equal to the number of classes and W^2 are the weights of the second layer of neural network. The supervised LVQ network is trained using standard Kohonen learning rule given in [51].

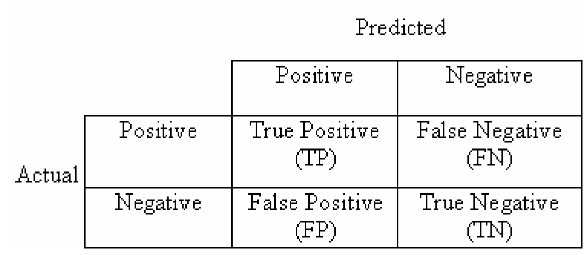
In LVQ network each neuron in the first layer is assigned to a class and then each class is assigned to one neuron in the second layer.

4.5 Performance Analysis

In this section the confusion matrix and the receiver operating characteristics are tabulated and plotted to choose the number of eigenvectors, to look at the effect of normalization process and to look at the generalization of the algorithm. Before the experimental results are presented, the terms associated with confusion matrix and ROC is explained.

4.5.1. Confusion Matrix

Confusion matrix is a visualization tool used in supervised learning for performance evaluation. Figure 4.8 shows an example of the confusion matrix and the terms associated with it are explained below.



		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 4.8 Confusion Matrix

True Positive is the number of correct predictions that's an instance is positive i.e. hits; False Positive is the number of incorrect predictions that an instance is positive i.e. false alarm; False Negative is the number of incorrect predictions that an instance is negative i.e. misses; True Negative is the number of correct predictions that an instance is negative i.e. correct rejections. For more details on the confusion matrix, see [55]. Two of the important terms associated with confusion matrix are True Positive rate (Sensitivity) = $\frac{TP}{TP+FN}$ and False Positive Rate (Specificity)= $\frac{FP}{FP+TN}$ which are used in to visualize the performance through receiver operating characteristics (ROC), explained in the next section.

4.5.2. Receiver Operating Characteristics

It is a plot of the classifier's true positive rate (sensitivity) against the false positive rate (specificity) explained in 4.5.1. High sensitivity means that the classifier identifies most of the positive samples and the desired performance of the classifier is good. High specificity means that the classifier identifies most of the negative samples and the performance of the classifier is not good.

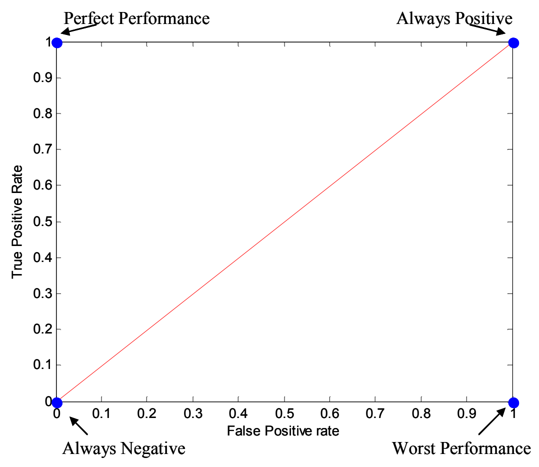


Figure 4.9 Receiver Operating Characteristic

As shown in Figure 4.9, the best performance is achieved near the point $(0,1)$ and the worst performance near the point $(1,0)$. In our case the multi-class task learning vector quantization is reduced to the binary decision i.e. either it belongs to a desired class or it doesn't belongs to it, no matter how many other classes there are. The results for the two stage neural network architecture proposed in this literature are visualized using ROC which is calculated by tabulating confusion matrix. For the different experiments conducted, the confusion matrix is tabulated and then the sensitivity and specificity are calculated to plot the ROC. For more details on ROC, see [56]

4.5.3.Experimental Results

The performance analysis of the proposed algorithm is performed by looking at the confusion matrix and by plotting receiver operating characteristics (ROC). The image database is taken from [42] and consists of 11 face image datasets, each of the 11 facial images having 4 expressions- normal, happy, sad, surprise. In this study we only used the images for normal, happy, and surprised expressions. We conducted two different tests to validate the use of the proposed algorithm in this paper. The first test experiment was conducted for only classification test and the second one for generalization test.

4.5.3.1 Classification Test

For the classification test, all the 11 images per expression (normal, happy, surprised) were taken into account for training (i.e. 33 images). For each expression, classification performance was tested for different number of eigenvectors estimated using 1st stage of NN.

The performance of the classification test is tabulated and plotted using confusion matrix and ROC,. respectively. For the choice of 3 number of eigenvectors, there are $C_3^{11} = 165$ combinations possible. Out of 165 combinations, 20 combinations of choosing 3 eigenvectors out of 11 were taken and classification step was performed for all the 11 image dataset (33 images). Due to the space constraint, the confusion matrix shown in Figure 4.8, is shown only for the 7 eigenvectors case and for only 1 combination in Table 4.1. For other choice on the number of eigenvectors, ROC is plotted as shown in Figure. 4.10.

Table 4.1 Confusion Matrix corresponding to 7 eigenvectors for Training Dataset

	Normal	Happy	Surprised
Normal	11	0	0
Happy	0	11	0
Surprised	0	0	11

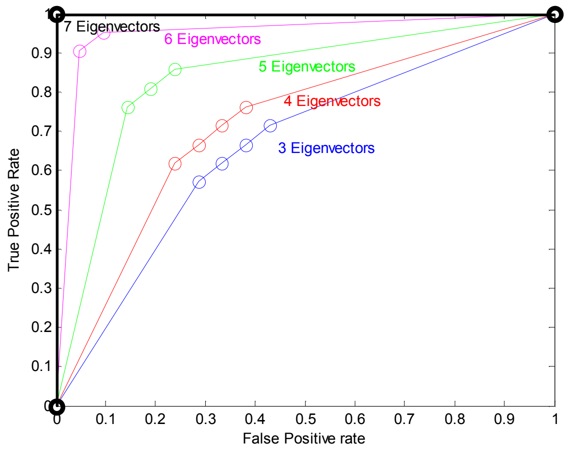


Figure 4.10 Classification Performance to choose number of eigenvectors

From the results shown in Figure 4.10 choice of 7 number of eigenvectors yields a classification rate of 100%.

4.5.3.2 Generalization Test

The generalization test was conducted to validate the versatility of the proposed algorithm. For the generalization test 7 images per expression (normal, happy, surprised) were taken into account for training (i.e.21 images). The remaining 4 image datasets (i.e. 12 images) were used to check the performance of the proposed 2 stage neural network.

Out of $C_7^{11} = 330$ combinations possible to choose for training and testing image set, 20 different combinations of 7 training image-set and 4 testing image-set were chosen. Due to the space constraint, the confusion matrix for testing image dataset in shown in table 4 for only 1 combination out of the chosen 20 different combinations. The ROC is plotted in fig. 4.11 for all the 20 combinations chosen.

Table 4.2 Confusion Matrix for Testing Dataset

	Normal	Happy	Surprised
Normal	4	0	0
Happy	0	3	1
Surprised	0	0	4

Figure 4.11 also presents the generalization result of linear discriminant analysis (LDA) algorithm proposed in [33, 57] With the neural nwtwork (NN) algorithm proposed in this paper, classification rate of 91.67% is achieved as compared to the classification rate of 83.33% by the LDA algorithm proposed in [33].

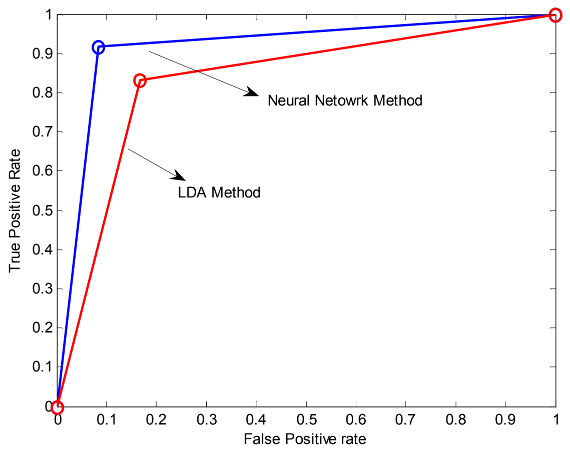


Figure 4.11 Classification performance with generalization

Figure 4.12 shows the performance of the algorithm in terms of the classification rate with and without normalization of the images. When normalization in terms of lighting, scale and pose is performed then we achieve a classification rate of 91.67% for testing image dataset as compared to a classification rate of 85.71% with no normalization.

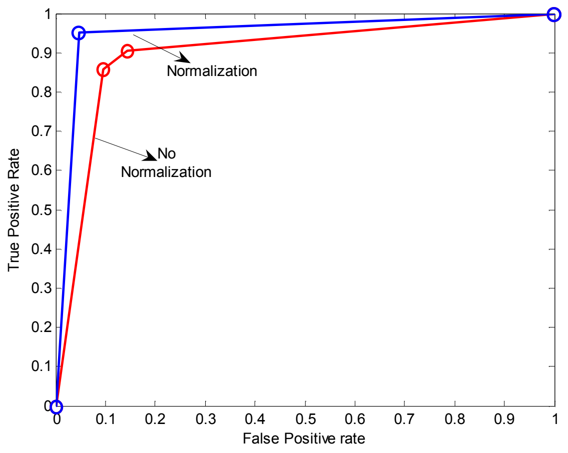


Figure 4.12 Classification performance with and without Normalization

4.6 Conclusion

Automating the analysis of facial expressions is important to achieve dexterity in non-verbal communication. In this paper, an approach based on neural network is presented for facial expression classification. Some of the important characteristics of the proposed algorithm are the following:

• The presented approach provides a simpler and easy to implement method for facial expression classification

• This paper provides a method which estimates and recursively updates the principal components to increase the computational efficiency.

The proposed algorithm also takes into account various factors such as lighting, scale, and pose inconsistencies to improve the performance of the facial expression classification.

This chapter presented the method for expression classification, which forms the basis to obtain the desired shape for flexible structure to morph. Until now, we have developed an algorithm which can obtain the desired shape using expression classification algorithm and then achieve the desired shape using the shape control algorithms presented in Chapter 2 and 3. In order to make the process of obtaining the desired shape of an object, it is important to have a method which takes into account the effect of any pose variations of the object. Therefore, the next chapter lays out the foundation to consider the pose of an object.

CHAPTER 5

POSE ESTIMATION OF 3D OBJECTS USING SEMI-DECOUPLED EXTENDED KALMAN FILTER

In Chapter 4, a two stage neural network algorithm was proposed to classify expressions. In order to achieve a robust classification procedure , the pose of the human face needs to be determined. Therefore, in this chapter an approach towards obtaining the pose of a 3D object (human head) is proposed. The chapter is organized in the following way: in Section 5.1, the transformation used for pose estimation is discussed. Section 5.2 presents the semi-decoupled EKF method for position and orientation vectors with only three non-collinear feature points. In Section 5.3, experimental results are presented highlighting the performance of the semi-decoupled EKF over a coupled 6D filter. Finally, the conclusion is presented in Section 5.4.

5.1 System Description

In this section, an introduction to the pose estimation algorithm is given which includes the approach to be adopted for pose estimation. A set of feature points on the images acquired by humanoid robot vision system are used to estimate the pose of the human. The feature points selected for this purpose are left eye, right eye and the location of the mouth. These feature points are extracted using standard image processing techniques such as horizontal and vertical gradient methods, circular Hough transform and probability distribution of skin color [76 - 78]. The relative coordinates of the feature points (left eye, right eye, mouth) are nominally known *a priori*. In our previous work [79], we presented an efficient two stage neural-network based approach for feature extraction and expression classification.

In this research work we test a target pose-estimation algorithm from face/feature recognition. For this purpose, we use a simple 3D object as target instead of a human. A much simpler image processing is performed on colored markers placed on the 3D object. Figure 5.1 shows the geometry of the system where the camera and the rigid object coordinate frames are represented. The feature point coordinates are related to the image frame (row, column) through a standard transformation written as:

$$\begin{bmatrix} c \\ r \end{bmatrix} = \frac{1}{Z^c} \begin{bmatrix} fx & 0 \\ 0 & fy \end{bmatrix} \begin{bmatrix} X^c \\ Y^c \end{bmatrix} + \begin{bmatrix} c_0 \\ r_o \end{bmatrix} \tag{5.1}$$

where $[c, r]$ are the image coordinates of the feature point, $[X^c, Y^c, Z^c]$ are the feature point coordinates in the camera frame, (fx, fy) are the intrinsic camera parameters defining the focal length and pixel dimensions and $[c_0, r_0]$ is the principal point of the camera frame in the image plane.

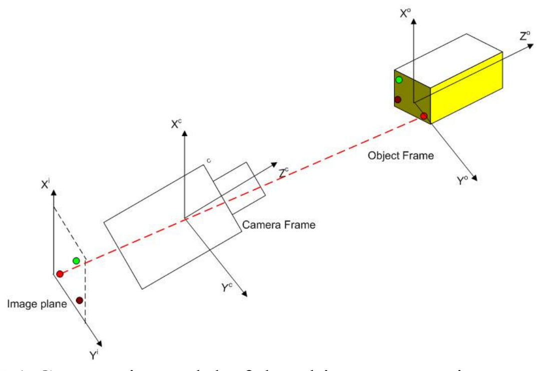


Figure 5.1 Geometric model of the object-camera-image reference frame

The feature point coordinates are determined in the camera frame using the transformation, given as:

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = R\left\{ \begin{bmatrix} X^o \\ Y^o \\ Z^o \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right\} \tag{5.2}$$

where $[X^o, Y^o, Z^o]$ is the known coordinate vector of the feature points in the object frame, $[X, Y, Z]$ is the unknown translation vector of the object and R is the rotation matrix of unknown orientation angles (roll (ϕ), pitch (θ) yaw (φ)) of the object, given by:

$$R = \begin{bmatrix} C_\phi C_\theta & C_\phi S_\theta S_\varphi - S_\phi C_\varphi & C_\phi S_\theta C_\varphi + S_\phi S_\varphi \\ S_\phi C_\theta & S_\phi S_\theta S_\varphi + C_\phi C_\varphi & S_\phi S_\theta C_\varphi - C_\phi S_\varphi \\ -S_\theta & C_\theta S_\varphi & C_\theta C_\varphi \end{bmatrix} \tag{5.3}$$

where $C_\phi = Cos(\phi)$; $S_\theta = Sin(\theta)$; $C_\varphi = Cos(\varphi)$

The transformation given in (5.1) and (5.2) forms the basis to estimate the position ($[X, Y, Z]$) and orientation (roll(ϕ), pitch(θ), yaw(φ)) vectors. These unknown parameters are estimated using a semi-decoupled EKF method proposed in the following section with respect to the camera frame.

5.2 Position and Orientation Estimation Using Semi-Decoupled Extended Kalman Filter

In this section, a semi-decoupled EKF approach is presented for 6D pose estimation. We call this estimator “semi-decoupled” because two separate filters are designed, one for estimation of position and the other one for orientation vectors, as shown in Figure 5.2.

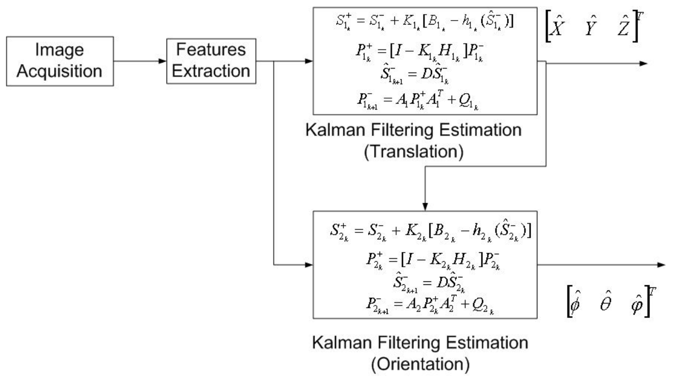


Image Acquisition
Features Extraction
$S^+_{1_k} = S^-_{1_k} + K_{1_k}[B_{1_k} - h_{1_k}(\hat{S}^-_{1_k})]$
$P^+_{1_k} = [I - K_{1_k}H_{1_k}]P^-_{1_k}$
$\hat{S}^-_{1_{k+1}} = D\hat{S}^-_{1_k}$
$P^-_{1_{k+1}} = A_1P^+_{1_k}A_1^T + Q_{1_k}$
Kalman Filtering Estimation (Translation)
$[\hat{X} \ \hat{Y} \ \hat{Z}]^T$
$S^+_{2_k} = S^-_{2_k} + K_{2_k}[B_{2_k} - h_{2_k}(\hat{S}^-_{2_k})]$
$P^+_{2_k} = [I - K_{2_k}H_{2_k}]P^-_{2_k}$
$\hat{S}^-_{2_{k+1}} = D\hat{S}^-_{2_k}$
$P^-_{2_{k+1}} = A_2P^+_{2_k}A_2^T + Q_{2_k}$
Kalman Filtering Estimation (Orientation)
$[\hat{\phi} \ \hat{\theta} \ \hat{\varphi}]^T$

Figure 5.2 Semi-Decoupled Kalman filter approach for object position and orientation vector (pose) estimation

For the estimation of position vector over a sampling step, no knowledge of orientation angles is required i.e. $\phi = 0$, $\theta = 0$, $\varphi = 0$. Therefore, the rotation matrix (R) for position vector EKF block is equal to identity. Whereas, for the orientation vector EKF block, the full nonlinear rotation matrix (R) given in (5.3) is used. The process of imaging 3D objects from single camera 2D images makes the transformation in (1) sensitive to depth information. Since estimation of orientation vector deals with the nonlinearity of the rotation matrix, it is sensitive to the position of the object, therefore the current estimates of the position vector are used in the orientation vector EKF block.

Another advantage of the proposed algorithm over the coupled EKF approach is that the matrix dimensions used in computations is reduced, and therefore improves the efficiency of the pose estimation process. The semi-decoupled approach also makes the estimation process more robust to large errors in initial condition, as shown by the experimental results.

As features used are image measurement points for the Kalman filter, we use only three non-collinear points, instead of 4, 5 or more used by others. It has been shown in [80, 81] that with three feature points, a given 3D object with known dimensions can be uniquely located in a 3D space with at most 4 distinct ambiguous solutions. In our case this ambiguity does not exist if the initial triangle pose estimate is close to the correct pose, or if the initial triangle pose estimate is roughly perpendicular to the camera line of sight. This is obvious because of the EKF state update and propagation from close to a correct estimate would yield the correct pose while the object is undergoing motion.

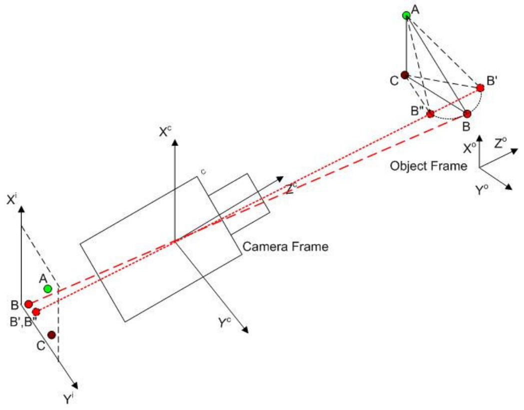


Figure 5.3 Illustration of three non-collinear points producing different pixel measurements as the object moves between ambiguous poses

For example, consider the triangle shown in Figure 5.3, with three vertices as the selected feature points. Points B' and B'' are produced by the rotation along $AC-axis$ by an angle φ. Even though these two points would yield same pixel measurements, the estimation process yields the incremental rotations the object has undergone from B' to B''. The whole trajectory is still ambiguous, but if we can distinguish B' to B'' at the beginning of the estimation process, we never lose track of the right pos. Because we use only 3 feature points, we do not require having the feature points in any particular special configuration to estimate the correct pose of the object, as is the case in other papers (for instance [82]).

The experimental results presented in Section V also demonstrate that we can track three non-collinear feature points through the proposed semi-decoupled EKF approach with high performance and convergence better than a full 6D coupled filter, such as the one in [67, 68].

The Kalman filter requires a dynamical model and a measurement model. For the dynamical model, it is assumed that the object moves with a constant velocity over a defined sampling time, which is certainly the case at low speeds, typical of natural human motions during conversations with a robot. The measurement model for the EKF is given by (5.1) and (5.2), as detailed in the following subsections.

5.2.1. Position Vector Estimation

For the estimation of position vector, the position dynamical model is defined as

$$\ddot{S}_1 = 0 \tag{5.4}$$

where $S_1 = \begin{bmatrix} X & Y & Z & \dot{X} & \dot{Y} & \dot{Z} \end{bmatrix}^T$. Converting the continuous model in (5.4) to the discrete time model, yields

$$S_{1_{k+1}} = A_1 S_{1_k} + \gamma_{1_k} \tag{5.5}$$

where γ_k is the disturbance noise of the discrete time dynamical model described as Gaussian with zero mean and a noise covariance of Q_1 and the plant matrix is given as

$$A_1 = \begin{bmatrix} O_3 & T \times I_3 \\ O_3 & O_3 \end{bmatrix} \tag{5.6}$$

where O_3 and I_3 are the 3x3 zero and identity matrices respectively, and T is the sampling time. The measurement model for estimation of position vector is defined by (5.1) and (5.2), where the rotation matrix R is taken to be *identity.* The Extended Kalman filter is defined by the following equations

Model:

$$S_{1_{k+1}} = A_1 S_{1_k} + \gamma_{1_k} \tag{5.6}$$

$$B_{1_k} = h_k(S_{1_k}) + \vartheta_{1_k} \tag{5.7}$$

where $B_{1_k} = \begin{bmatrix} c & r \end{bmatrix}^T$, $h_{1_k} = \frac{1}{Z^c}\begin{bmatrix} fx & 0 \\ 0 & fy \end{bmatrix}\begin{bmatrix} X^c \\ Y^c \end{bmatrix}$ and ϑ_{1_k} is the measurement noise with the covariance given by R_{1_k}.

Gain:

$$K_{1_k} = P_{1_k}^- H_{1_k}^T [H_{1_k} P_{1_k}^- H_{1_k}^T + R_{1_k}]^{-1} \tag{5.8}$$

where $H_{1_k} = \left.\frac{\partial h_k}{\partial S_{1_k}}\right|_{S_{1_k}^-}$ $H_{1_k} = \begin{bmatrix} fx\frac{1}{Z^c} & 0 & -fx\frac{X^c}{(Z^c)^2} & 0 & 0 & 0 \\ 0 & fy\frac{1}{Z^c} & -fy\frac{Y^c}{(Z^c)^2} & 0 & 0 & 0 \end{bmatrix}$

Update:

$$S_{1_k}^+ = S_{1_k}^- + K_{1_k}[B_{1_k} - h_{1_k}(\hat{S}_{1_k}^-)] \tag{5.9}$$

$$P_{1_k}^+ = [I - K_{1_k} H_{1_k}] P_{1_k}^- \tag{5.10}$$

Propagation:

$$\hat{S}^{-}_{1_{k+1}} = D\hat{S}^{-}_{1_k} \tag{5.11}$$

$$P^{-}_{1_{k+1}} = A_1 P^{+}_{1_k} A_1^T + Q_{1_k} \tag{5.12}$$

This position vector estimate from this Kalman filter is now fed into a second filter used for estimating the orientation vector in the following subsection.

5.2.2. Orientation Vector Estimation

For the estimation of the orientation vector, we define the dynamical model as:

$$\ddot{S}_2 = 0 \tag{5.13}$$

where $S_2 = \begin{bmatrix}\phi & \theta & \varphi & \dot{\phi} & \dot{\theta} & \dot{\varphi}\end{bmatrix}^T$. Converting the continuous model in (5.13) to the discrete time model, yields

$$S_{2_{k+1}} = A_2 S_{2_k} + \gamma_{2_k} \tag{5.14}$$

where γ_{2_k} is the disturbance noise of the discrete time dynamical model described as Gaussian with zero mean and a noise covariance of Q_{2_k} and the plant matrix is $A_2 = A_1$. The Extended Kalman filter update and propagation equations remain the same as before, with the difference being in the states to be used for estimation. The Jacobian matrix for the orientation vector estimation is given as

$$H_{2_k} = \left.\frac{\partial h_k}{\partial S_{2_k}}\right|_{S^{-}_{2_k}} \tag{5.15}$$

where
$$H_{2_k} = \begin{bmatrix} \frac{fx}{(Z^c)^2}\left(\frac{Z^c\partial X^c}{\partial\phi} - \frac{X^c\partial Z^c}{\partial\phi}\right) & \frac{fx}{(Z^c)^2}\left(\frac{Z^c\partial X^c}{\partial\theta} - \frac{X^c\partial Z^c}{\partial\theta}\right) & \frac{fx}{(Z^c)^2}\left(\frac{Z^c\partial X^c}{\partial\varphi} - \frac{X^c\partial Z^c}{\partial\varphi}\right) & 0 & 0 & 0 \\ \frac{fy}{(Z^c)^2}\left(\frac{Z^c\partial Y^c}{\partial\phi} - \frac{Y^c\partial Z^c}{\partial\phi}\right) & \frac{fy}{(Z^c)^2}\left(\frac{Z^c\partial Y^c}{\partial\theta} - \frac{Y^c\partial Z^c}{\partial\theta}\right) & \frac{fy}{(Z^c)^2}\left(\frac{Z^c\partial Y^c}{\partial\varphi} - \frac{Y^c\partial Z^c}{\partial\varphi}\right) & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial X^c}{\partial\phi} = -(S_\phi C_\theta)(X^o + X) + (-S_\phi S_\theta S_\varphi - C_\phi C_\varphi)(Y^o + Y) + (-S_\phi S_\theta C_\varphi + C_\phi S_\varphi)(Z^o + Z) \tag{5.16}$$

$$\frac{\partial X^c}{\partial\theta} = -(C_\phi C_\theta)(X^o + X) + (C_\phi C_\theta S_\varphi)(Y^o + Y) + (C_\phi C_\theta C_\varphi)(Z^o + Z) \tag{5.17}$$

$$\frac{\partial X^c}{\partial\varphi} = (C_\phi S_\theta C_\varphi + S_\phi S_\varphi)(Y^o + Y) + (-C_\phi S_\theta S_\varphi + S_\phi C_\varphi)(Z^o + Z) \tag{5.18}$$

$$\frac{\partial Y^c}{\partial\phi} = (C_\phi C_\theta)(X^o + X) + (C_\phi S_\theta S_\varphi - S_\phi C_\varphi)(Y^o + Y) + (C_\phi S_\theta C_\varphi + S_\phi S_\varphi)(Z^o + Z) \tag{5.19}$$

$$\frac{\partial Y^c}{\partial\theta} = -(S_\phi S_\theta)(X^o + X) + (S_\phi C_\theta S_\varphi)(Y^o + Y) + (S_\phi C_\theta C_\varphi)(Z^o + Z) \tag{5.20}$$

$$\frac{\partial Y^c}{\partial\varphi} = (S_\phi S_\theta C_\varphi - C_\phi S_\varphi)(Y^o + Y) + (-S_\phi S_\theta S_\varphi - C_\phi C_\varphi)(Z^o + Z) \tag{5.21}$$

$$\frac{\partial Z^c}{\partial \phi} = 0 \tag{5.22}$$

$$\frac{\partial Z^c}{\partial \theta} = (C_\theta)(X^o + X) + (-S_\theta S_\varphi)(Y^o + Y) + (-S_\theta C_\varphi)(Z^o + Z) \tag{5.23}$$

$$\frac{\partial Z^c}{\partial \varphi} = (C_\theta C_\varphi)(Y^o + Y) + (-C_\phi S_\theta)(Z^o + Z) \tag{5.24}$$

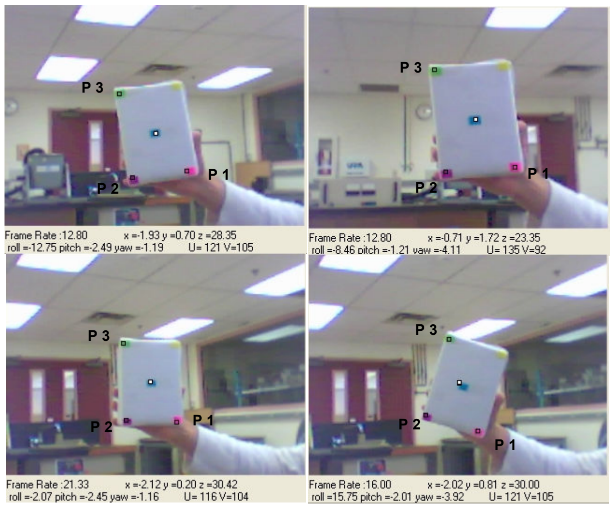
5.3 Implementation Results of Pose Estimation Using Semi-Decoupled Extended Kalman Filter

The performance of the proposed algorithm is demonstrated in this section through several experiments we conducted. The first set of experiment compares the performance of the semi-decoupled EKF with the 6D coupled filter when the camera is stationary. The second set of experiments combines the pose estimator and visual tracker.

In the experiments conducted, the camera parameters values are $fx = -300$, $fy = 300$, $c_0 = 83$, $r_0 = 113$, $T = 0.04$. The sampling time T is based on the measured frames per second rate, which was 26 fps. Figure 5.4 shows the object that was used for pose estimation and tracking, while undergoing different sets of motion.

Three non-collinear feature points (P_1, P_2, P_3) were used and their nominal coordinates are given (in inches) as $P_1 = [-3,2.1,0]$, $P_2 = [-3,-2.1,0]$, $P_1 = [3,-2.1,0]$.

In this case, the object moves while the camera is stationary. Several sets of motions of the object with various initial conditions of the unknown parameters were conducted, to compare the performance of the semi-decoupled EKF method over the coupled EKF method. The result shown here had an object motion sequence involving translation along *y-axis, z*-axis and rotation around *z-axis.* Since the estimation process is sensitive to the depth information, which in this case is the z-coordinate, the initial conditions used for the unknown parameters are varied to a large variation in z-coordinate. The initial pose estimates are $[X,Y,Z]^T = [1,1,2]^T$ $[\phi,\theta,\varphi]^T = [0,0,0]^T$, far away in Z from the actual pose $[X,Y,Z]^T = [0,0,20]^T$ and $[\phi,\theta,\varphi]^T = [0,0,0]^T$



```
Frame Rate :12.80          x =-1.93 y =0.70 z =28.35
 roll =-12.75 pitch =-2.49 yaw =-1.19        U= 121 V=105
```

```
Frame Rate :12.80          x =-0.71 y =1.72 z =23.35
 roll =-8.46 pitch =-1.21 yaw =-4.11        U= 135 V=92
```

```
Frame Rate :21.33          x =-2.12 y =0.20 z =30.42
 roll =-2.07 pitch =-2.45 yaw =-1.16        U= 116 V=104
```

```
Frame Rate :16.00          x =-2.02 y =0.81 z =30.00
 roll =15.75 pitch =-2.01 yaw =-3.92        U= 121 V=105
```

Figure 5.4 Experimental setup showing the three features points in different poses (red, violet, green)

The estimation results are shown in appendix A. From the estimated pixel coordinates ($\begin{bmatrix} c & r \end{bmatrix}^T$) for each feature point, an error function is defined as,

$$e = (c_m^2 + r_m^2)^{1/2} - (c^2 + r^2)^{1/2} \tag{5.16}$$

where $\begin{bmatrix} c_m & r_m \end{bmatrix}^T$ are obtained from the measurement set.

Figure 5.5 plots the error function defined in (5.16) for the semi-decoupled EKF approach (solid line) and the coupled EKF approach (dotted line) and the Frobenius norm is calculated for semi-decoupled (229.12) and coupled EKF method (667.24).

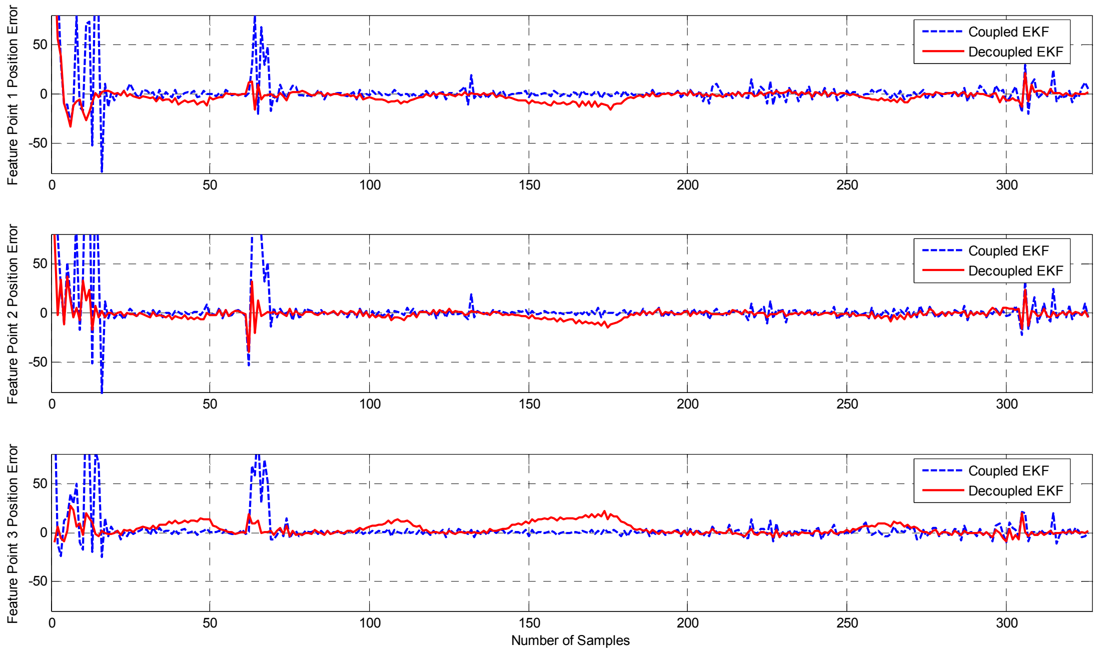


Figure 5.5 Error plot of estimated pixel output comparing Decoupled Kalman filter (3D+3D in solid red line) and Coupled Kalman filter (6D, in dashed blue line)

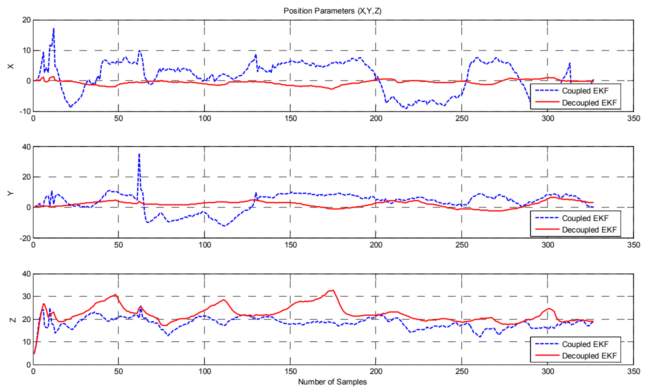


Figure 5.6 Position estimates comparing Decoupled Kalman filter (3D+3D in solid red line) and Coupled Kalman filter (6D, in dashed blue line) performance

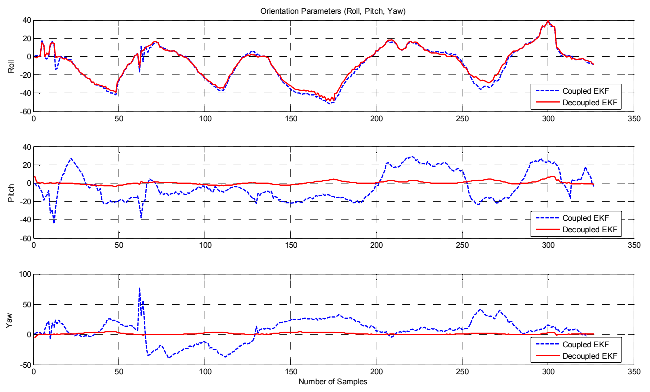


Figure 5.7 Orientation estimates comparing Decoupled Kalman filter (3D+3D in solid red line) and Coupled Kalman filter (6D, in dashed blue line) performance

These results clearly illustrate the robustness of the semi-decoupled EKF approach over the coupled EKF approach. Figure 5.6 and Figure 5.7 plots the position and orientation estimates obtained using semi-decoupled and coupled approach. From the motion conducted of the object, the estimates obtained from the coupled EKF (dotted line) approach have larger variations as compared to the semi-decoupled EKF approach (solid line).

5.4 Conclusion

An efficient pose estimation was proposed and implemented using a decoupled EKF. Two filters, one for position, one for orientation of the object are used instead of a single 6D filter. Experimental results clearly show improved tracking accuracy, reduced computational load, and better robustness of the semi-coupled filter approach.

This chapter concludes the first part on the use of the potential field method for shape control of flexible structure. A complete system starting from shape control of flexible structure to optimally selecting/placing the microactuators to acquiring the desired shape under object movement is developed. The next chapter uses the concept of potential field method in the field of wireless sensor network, where the main concern is to find the position of the sensor nodes with only the range measurement available.

CHAPTER 6

A POTENTIAL FIELD APPROACH FOR DYNAMIC LOCALIZATION OF WIRELESS SENSOR NETWORKS

This chapter is organized in the following way: section 6.1 derives the position estimate dynamical model and the potential field function to be used for the relative localization of a stationary UGS network. Section 6.1.4 discusses the algorithm used for relative localization and simulation results are presented and in section 6.1.5, experimental results are given. Section 6.2 extends the idea to absolute localization and presents the dynamical estimator model and a modified potential field function used for absolute localization. Section 6.2.3 discusses the algorithm used for absolute localization and simulation results are presented and in section 6.2.5, experimental results are given. Finally the conclusion is presented in Section 6.4

6.1 Virtual Node Dynamics for Relative Localization

In this section we present a novel method for relative localization of a network of stationary unattended ground sensors (UGS). It is assumed that distance (i.e. range) measurements between sensor nodes are available, specifically each sensor node measures the distance to at least three other nodes. The method uses a dynamical model for position estimates of each node that is driven by a fictitious virtual force based on range errors. These virtual dynamics have states which are the estimates of the relative positions, and reach a steady-state value that provides an optimal (in a sense to be made precise herein) estimate of the relative positions of all nodes in the network.

The UGS nodes do not physically move, but the virtual dynamics capture the available range information to dynamically compute the UGS relative position estimates. A certain potential field is introduced to generate optimal position locations in a least-squares sense. The potential field is used as a Lyapunov function and a Lyapunov proof shows how to generate appropriate virtual forces based on the gradient of the potential field.

6.1.1. System Description

Here, we describe the virtual dynamics used for generating position estimates of the stationary UGS nodes based on range information. The position estimate for the i^{th} sensor node is given by

$$X_i = [x_i \quad y_i]^T \tag{6.1}$$

where, x_i and y_i are the *x-y* coordinates for the UGS node position estimate. The position estimation dynamics are given as

$$\ddot{X}_i = \vec{f}_i \tag{6.2}$$

where, $\vec{f}_i = \begin{bmatrix} f_i^x & f_i^y \end{bmatrix}^T$ is the virtual force in the *x* and *y* directions to be specified. The state variable description form for the position estimate of the i^{th} UGS node is by

$$\begin{bmatrix} \dot{X}_i \\ \ddot{X}_i \end{bmatrix} = \begin{bmatrix} O_2 & I_2 \\ O_2 & O_2 \end{bmatrix} \begin{bmatrix} X_i \\ \dot{X}_i \end{bmatrix} + \begin{bmatrix} O_2 \\ I_2 \end{bmatrix} \begin{bmatrix} f_i^x \\ f_i^y \end{bmatrix} \tag{6.3}$$

where, $O_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

6.1.2. Potential Field for Optimal Position Estimation for Relative Localization

A potential field is now introduced to determine the virtual force $\vec{f}_i$ in (6.2) so that the position estimates reach a steady-state value that is an optimal estimate for the actual UGS node relative positions. Define a potential field as

$$V_{ugs} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{2} K_{ij} (r_{ij} - \bar{r}_{ij})^2 \tag{6.4}$$

where $r_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ is the estimated range and $\bar{r}_{ij}$ is the actual measured range between i^{th} and j^{th} UGS nodes.

Define the potential function for a single UGS node i by

$$V_{i_{ugs}} = \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{2} K_{ij} (r_{ij} - \bar{r}_{ij})^2 \tag{6.5}$$

The gradient of the potential with respect to the sensor node state is given by

$$\frac{\partial V_{i_{ugs}}}{\partial X_i} = \vec{\nabla} V_{i_{ugs}} = \vec{\nabla} \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{1}{2} K_{ij} (r_{ij} - \bar{r}_{ij})^2 = \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \vec{\nabla} (r_{ij} - \bar{r}_{ij}) \tag{6.6}$$

and on further simplification, the gradient is written as

$$\frac{\partial V_{i_{ugs}}}{\partial X_i} = \sum_{\substack{j=1 \\ i \neq j}}^{N} K_{ij} (r_{ij} - \bar{r}_{ij}) \left[\frac{x_i - x_j}{\left\| X_i - X_j \right\|} \hat{x} \quad \frac{y_i - y_j}{\left\| X_i - X_j \right\|} \hat{y} \right] \tag{6.7}$$

where $\left\| X_i - X_j \right\| = r_{ij}$

Theorem 6.1: Consider the position estimate dynamics (6.2) for each sensor node i in the network. Let the virtual force for i^{th} sensor node be given as

$$\vec{f}_i = -\sum_{\substack{j=1 \\ i\neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\frac{(X_i - X_j)}{\|X_i - X_j\|} - K_v \dot{X}_i \tag{6.8}$$

Then the position estimates reach steady-state values that provide optimal estimates of the actual relative localization of the nodes in the sense that V_{ugs} is minimized.

Proof: Define the Lyapunov function

$$L = V_{ugs} + \sum_{i=1}^{N} \frac{1}{2} \dot{X}_i^T \dot{X}_i \tag{6.9}$$

Differentiate to obtain

$$\dot{L} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i\neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\dot{r}_{ij} + \sum_{i=1}^{N} \dot{X}_i^T \ddot{X}_i \tag{6.10}$$

One can compute

$$\dot{r}_{ij} = \frac{\dot{X}_i^T (X_i - X_j)}{\|X_i - X_j\|} \tag{6.11}$$

and on substituting (6.2) and (6.11), we obtain

$$\dot{L} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i\neq j}}^{N} K_{ij}(r_{ij} - \bar{r}_{ij})\frac{\dot{X}_i^T (X_i - X_j)}{\|X_i - X_j\|} + \sum_{i=1}^{N} \dot{X}_i^T \vec{f}_i \tag{6.12}$$

Further, substituting the force (6.8) yields

$$\dot{L} = -\sum_{i=1}^{N} \dot{X}_i^T K_v \dot{X}_i \tag{6.13}$$

clearly, $\forall \; K_v > 0$, $\dot{L} \le 0$ and the vector $\begin{bmatrix} X_i & \dot{X}_i \end{bmatrix}^T$ is bounded which shows that the position estimate dynamics is SISL. Evaluating $\ddot{L}$ yields

$$\ddot{L} = -2\sum_{i=1}^{N} \dot{X}_i^{\,T} K_v \vec{f}_i \tag{6.14}$$

and on substituting (6.8) we obtain

$$\ddot{L} = -2\sum_{i=1}^{N} \dot{X}_i^{\,T} K_v \left\{ -\sum_{j=1}^{N} K_{ij}(r_{ij} - \bar{r}_{ij}) \frac{(X_i - X_j)}{\left\| X_i - X_j \right\|} - K_v \dot{X}_i \} \right\} \tag{6.15}$$

Using the result obtained from Lyapunov analysis that the vector $\begin{bmatrix} X_i & \dot{X}_i \end{bmatrix}^T$ is bounded yields that $\ddot{L}$ in (6.15) is also bounded. By Barbalat's Lemma [105] we deduce that $\dot{L} \to 0$ as $t \to \infty$, which yields $\dot{X}_i \to 0$ as $t \to \infty$. Therefore (6.2) shows that $\vec{f}_i$ goes to zero $\forall i$. Finally (6.7) and (6.8) show that $\frac{\partial V_{ugs}}{\partial X_i} \to 0, \forall i$, so V_{ugs} reaches a minimum.■

6.1.3. Relative Localization Scheme for UGS Networks

Using the system defined in (6.2) and the force control input defined as in (6.8) the wireless sensor network can be relatively localized. However, due to a nonlinear mapping between the range (r_{ij}) and the x-y coordinates, there are local minima in Lyapunov function in (6.9), which could lead to incorrect position estimates for the nodes. Though the Lyapunov term defined in (6.9) is positive definite in terms of range information, the Lyapunov derivative in (6.13) is independent of the range information term and is only dependent on the velocity of position estimates defined in (6.1). In this section, we study the problem of local minima and propose an algorithm to ensure that the network always converges to a unique solution.

6.1.3.1 Study of Local Minima

The next results show that a unique equilibrium point exists in a 3 sensor node configuration, but there exist local minima in a 4 sensor node configuration when the virtual force input is given by (6.8). Further in the section, we develop an algorithm that always guarantees a unique equilibrium configuration for position estimate system in (6.2), for any number of UGS nodes.

Lemma 6.1: Let there be given 3 stationary UGS nodes (not in a straight line) and range measurements between all nodes. Then the position estimate dynamics in (6.2) with the virtual force in (6.8) converges to a unique steady-state value, which provides an optimal estimate for the relative position in terms of minimum potential field in (6.4).

Proof: Assuming the first node to be at the origin and the second node to be along the *x-axis,* the co-ordinates of first and second nodes are given as $(0,0)$ and $(r_{12},0)$ respectively. Referring to Figure 6.1 the co-ordinates of the third node could be found which are

$$x_3 = r_{13}\cos(\theta); \quad \mathrm{y}_3 = r_{13}\sin(\theta) \tag{6.16}$$

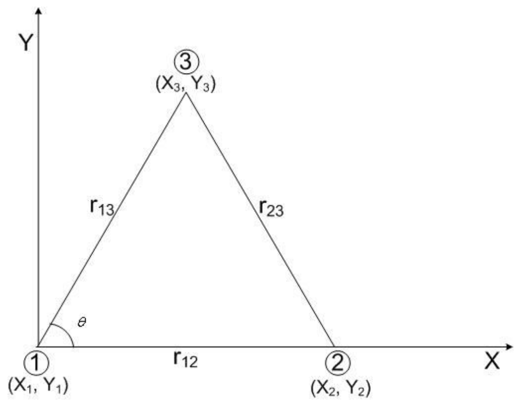


Figure 6.1 Three UGS Node case

Using (6.8) to write down the force equation for node 1 we get

$$f_1^x = -\frac{(r_{12}-\bar{r}_{12})}{r_{12}}(x_1-x_2) - \frac{(r_{13}-\bar{r}_{13})}{r_{13}}(x_1-x_3) \tag{6.17}$$
$$f_1^y = -\frac{(r_{12}-\bar{r}_{12})}{r_{12}}(y_1-y_2) - \frac{(r_{13}-\bar{r}_{13})}{r_{13}}(y_1-y_3)$$

and on substituting the co-ordinate values obtained in (16) yields

$$f_1^x = -(r_{12}-\bar{r}_{12}) - (r_{13}-\bar{r}_{13})\cos(\theta) \tag{6.18}$$
$$f_1^y = -(r_{13}-\bar{r}_{13})\sin(\theta)$$

From the proof of theorem 6.1, for the equilibrium point the forces $\vec{f}_1$ go to zero. Therefore, for $f_1^y = 0$, $r_{13} = \bar{r}_{13}$ is the only solution since $\sin(\theta) \neq 0$ under the constraint that the 3 nodes are not in a straight line. Looking back at (6.18) and in order to get $f_1^x = 0$, $r_{12} = \bar{r}_{12}$ is the only solution, which leads us to the unique solution of relative localization for 3 nodes. ■

Lemma 6.2: Let there be given 4 stationary UGS (no 3 of which are in a straight line) and range measurements between all nodes. Then the position estimate dynamics in (6.2) with the virtual force in (6.8) may not converge to a unique steady-state value.

Proof: Referring to Figure 6.2 and writing the co-ordinates for the 4 node case we get

$$\begin{aligned} &x_1 = 0, y_1 = 0;\ x_2 = r_{12}, y_2 = 0; \\ &x_3 = r_{13}\cos(\theta);\quad y_3 = r_{13}\sin(\theta) \\ &x4 = r_{14}\cos(\theta');\quad y_4 = r_{14}\sin(\theta') \end{aligned} \tag{6.19}$$

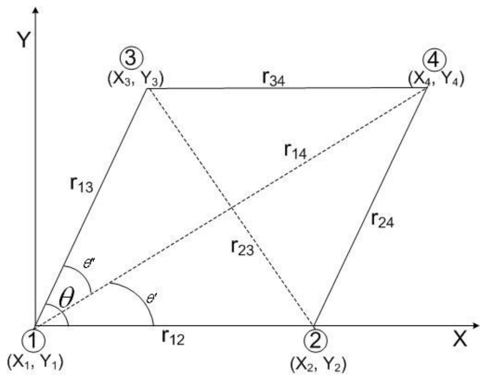


Figure 6.2 Four UGS Node case

Using (6.8) to write down the force equation for node 1 we get

$$\begin{aligned} f_1^{\,x} &= -\frac{(r_{12}-\bar{r}_{12})}{r_{12}}(x_1-x_2)-\frac{(r_{13}-\bar{r}_{13})}{r_{13}}(x_1-x_3)-\frac{(r_{14}-\bar{r}_{14})}{r_{14}}(x_1-x_4) \\ f_1^{\,y} &= -\frac{(r_{12}-\bar{r}_{12})}{r_{12}}(y_1-y_2)-\frac{(r_{13}-\bar{r}_{13})}{r_{13}}(y_1-y_3)-\frac{(r_{14}-\bar{r}_{14})}{r_{14}}(y_1-y_4) \end{aligned} \tag{6.20}$$

and on substituting the coordinates values from (6.19) yields

$$\begin{aligned} f_1^{\,x} &= -(r_{12}-\bar{r}_{12})-(r_{13}-\bar{r}_{13})\cos(\theta)-(r_{14}-\bar{r}_{14})\cos(\theta') \\ f_1^{\,y} &= -(r_{13}-\bar{r}_{13})\sin(\theta)-(r_{14}-\bar{r}_{14})\sin(\theta') \end{aligned} \tag{6.21}$$

The total force applied on node 1 is given as

$$f_1^{\,2} = (f_1^{\,x})^2 + (f_1^{\,y})^2 \tag{6.22}$$

and on substitution of (6.21) in (6.22) and simplifying the terms, we get

$$f_1^2 = a^2 + b^2 + c^2 + 2ab\cos(\theta) + 2ac\cos(\theta') + 2bc\cos(\theta'') \quad (6.23)$$

where

$$a = r_{12} - \bar{r}_{12}; \quad b = r_{13} - \bar{r}_{13}; \quad c = r_{14} - \bar{r}_{14} \quad (6.24)$$

However, the force equation in (6.23) can also be obtained with the different configuration of nodes shown in Figure 6.3, in which the co-ordinates for the four nodes are given as

$$\begin{aligned} &x_1 = 0,\ y_1 = 0;\ x_2 = r_{12},\ y_2 = 0; \\ &x_3 = r_{13}\cos(\theta); \quad y_3 = r_{13}\sin(\theta) \\ &x4 = r_{14}\cos(\theta'); \quad y_4 = -r_{14}\sin(\theta') \end{aligned} \quad (6.25)$$

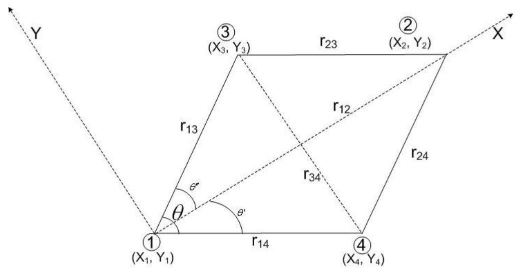


Figure 6.3 Alternate configuration for four UGS nodes

Thus, there are 2 minimum configurations for the sensor node to fall into local minima, one of which has $r_{ij} \neq \bar{r}_{ij}$. ■

6.1.4. Relative Localization Algorithm

Existence of local minima as mentioned in Lemma 6.2, with the potential field function defined in (6.4) can be resolved by the addition of only one node at a time to the sensor network when $N \geq 4$. According to Lemma 6.1, the first 3 nodes to arise in the UGS network attain a unique steady-state value for their position estimate. The final coordinates of these first 3 UGS nodes, already relatively localized, are used to calculate the initial starting point for the next UGS node to be added in the network.

The trilateration method [104] is used to obtain the initial starting point for the new UGS node. Use of only the trilateration process is inefficient for least-squares localization of large wireless sensor nodes since, when a node is added, all other nodes may have to change their positions to reduce the potential occurred due to the measurement error. Here, trilateration is only used to get an initial position estimate for the new UGS node. Once the initial position estimate has been obtained, our algorithm is used to relatively localize the network with the control input in (6.8). This allows the relative position estimates of all the nodes in the network to be adjusted each time a node is added.

The starting point for UGS nodes when $N \geq 4$ is found using

$$\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} 2(x_{j_1} - x_{j_3}) & 2(y_{j_1} - y_{j_3}) \\ 2(x_{j_2} - x_{j_3}) & 2(y_{j_2} - y_{j_3}) \end{bmatrix}^{-1} \begin{bmatrix} \bar{r}_{j_3 j_4}{}^2 - \bar{r}_{j_1 j_4}{}^2 + x_{j_1}{}^2 - x_{j_3}{}^2 + y_{j_1}{}^2 - y_{j_3}{}^2 \\ \bar{r}_{j_3 j_4}{}^2 - \bar{r}_{j_2 j_4}{}^2 + x_{j_2}{}^2 - x_{j_3}{}^2 + y_{j_2}{}^2 - y_{j_3}{}^2 \end{bmatrix} \tag{6.26}$$

where, (x_j, y_j) denotes the current estimate of the location of node j and $(x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2}), (x_{j_3}, y_{j_3})$ denotes the position estimates of the already localized nodes j_1, j_2, j_3 respectively.

6.1.5. Relative Localization Simulation Results

The relative localization algorithm mentioned above is simulated with seven UGS nodes in the sensor network, where each UGS node is assumed to have range measurement information with at least 3 other UGS nodes. The algorithm localizes the UGS nodes relative to each other and yields an optimal relative position estimates. The results are shown in Figure 6.4, where the estimates are plotted as a functions of time given some incorrect initial conditions. One node is added at a time, then all the estimates in the network are readjusted. The figure shows that the estimates converge to the true relative locations of the UGS.

Table 6.1 Relative Localization Algorithm

1	Initialize $N = 3$ (number of UGS nodes in the network to start with the localization process)
2	Dynamically localize the network with $N = 3$ using (6.8) with range measurement information.
3	Increment N by 1 to keep a count of the number of sensor nodes
4	Initialize the next sensor node position using (6.26).
5	Dynamically localize the network with the new sensor node using (6.8)
6	Repeat steps 3-5 until all the nodes have been localized

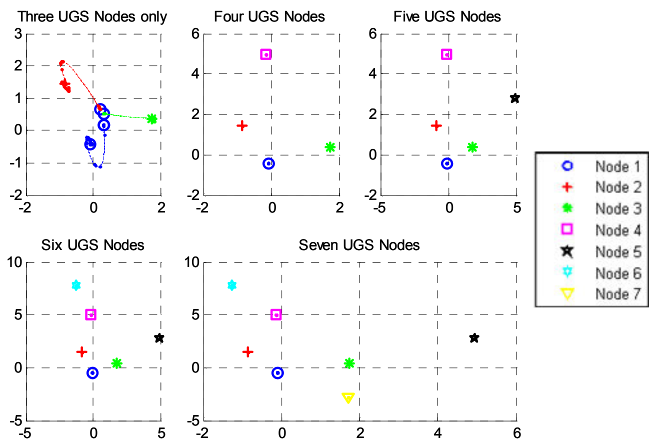


Figure 6.4 Relative Localization of 7 UGS nodes

6.1.6. Relative Localization Experimental Results

The relative localization algorithm mentioned in Table 6.1 is implemented using the relative localization hardware mentioned in Section 6.1.5. The implementation starts by measuring the distance of 3 nodes and localizing it as shown in Figure 6.5 (a). For the simplicity of demonstration of the algorithm in real time scenario, the range measurement reading are averaged over a time interval in order to take into account the effect of noise. Upon addition of one node at a time, all the estimates in the network are readjusted. When the range measurement of 4th sensor node is available, the sensor node reconfigures itself to include the 4th sensor node as shown in Figure 6.5(b). Similar process is repeated when 5th sensor node range measurement becomes available, as shown in Figure 6.5(c). The results for localizing the sensor nodes relatively are shown in Figure 6.6. The co-ordinates obtained in the Figure 6.6 are relative to each other and the relative localization algorithm results tells about the orientation of the sensor nodes and provides the co-ordinates in relative sense to the internal co-ordinate frame.

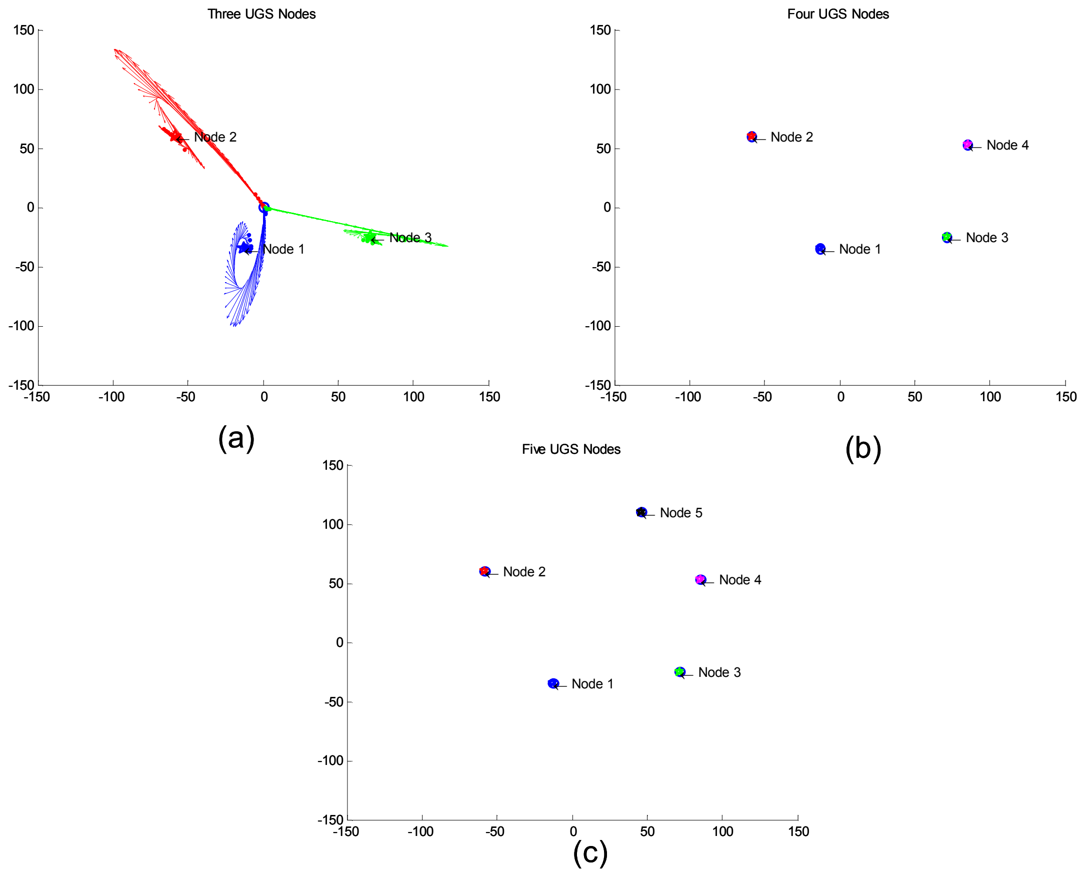


Figure 6.5 Relative Localization of 5 UGS nodes (a) with 3 nodes (b) with 4 nodes (c) with 5 nodes

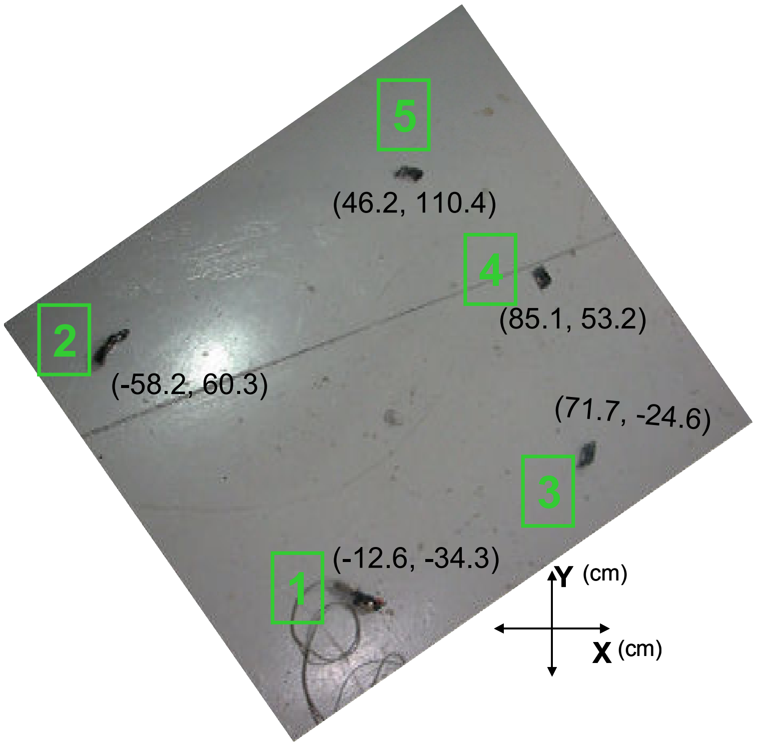


Figure 6.6 Relative Localization of 5 Cricket nodes (all units are in cm)

Figure 6.7 shows the effect of different values of gain K_v on the convergence of position estimates. For higher values of K_v the oscillations are reduced but at the same time the convergence time of the position estimates increases. The gain value K_v is selected as a trade off between the convergence time and the oscillations. In the experimental results presented here, the gain values used are $K_{ij} = 1$ and $K_v = 1$.

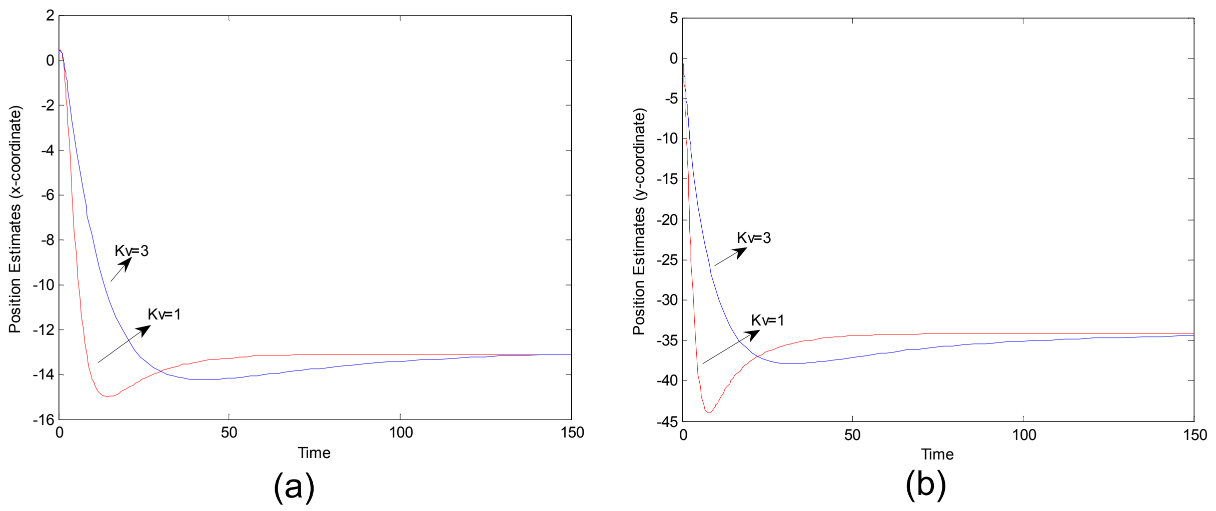


Figure 6.7 Effect of different values of Kv on position estimates convergence (a) on x-co-ordinates (b) on y-co-ordinates

6.2 Virtual Node Dynamics for Absolute Localization

In this section we discuss the absolute localization of the UGS nodes with the help of UAV. The UAV are assumed to have their absolute position information through on-board GPS or through contact with ground control station (GCS). To the potential field defined in (6.4) for relative localization, we add a second potential field for absolute localization. The modified potential field is used as a Lyapunov function and a Lyapunov proof shows how to generate appropriate virtual forces based on the gradient of the modified potential field.

6.2.1. System Description

For the absolute localization algorithm the following assumptions are being made for UAVs, which are

1) UAVs have an altitude hold autopilot [91].

2) UAVs are operated in hover mode, so that they move over the stationary UGS network and assume a fixed position until the localization algorithm to be presented has been executed.

Here, we describe the virtual dynamics used for position estimates of the stationary UGS nodes based on range information and absolute position information of UAV. Let the total number of UGS nodes and UAV in the air-ground sensor network be N indexed by a set X_i; $i = \{1,2,3,.......N\}$. Let the number of UAV with absolute position information be m, indexed by a set $X_{i_p}{}^{a}$; $\{p = 1,2,....., m\}$ such that $X_{i_p}{}^{a} \subset X_i$ and the UGS nodes with no absolute position information be indexed by X_{i_p}; $\{p = m+1,....., N\}$ such that $X_{i_p}{}^{ugs} \subset X_i$.

The position estimate for UGS nodes with no absolute position information is given as

$$X_{i_p} = \begin{bmatrix} x_{i_p} & y_{i_p} \end{bmatrix}^T \tag{6.27}$$

where x_{i_p} and y_{i_p} are the *x-y* coordinates of the UGS node position estimates. The position estimation dynamics are given as

$$\ddot{X}_{i_p} = \vec{f}_{i_p} \tag{6.28}$$

where, $\vec{f}_{i_p} = \begin{bmatrix} f_{i_p}{}^{x} & f_{i_p}{}^{y} \end{bmatrix}^T$ is a virtual force in the x and y directions to be specified.

The position estimate for UAV with absolute position information is given as

$$X_{i_p}{}^{a} = \begin{bmatrix} x_{i_p}{}^{a} & y_{i_p}{}^{a} \end{bmatrix}^T \tag{6.29}$$

where ${x_{i_p}}^a$ and ${y_{i_p}}^a$ are the *x-y* coordinates of the UAV position estimates. The position estimation dynamics are given as

$$\ddot{X}_{i_p}{}^a = \vec{f}_{i_p}{}^a \tag{6.30}$$

where,

$$\vec{f}_{i_p}{}^a = \left[f_{i_p}{}^{ax} \quad f_{i_p}{}^{ay} \right]^T \tag{6.31}$$

is a virtual force in the x and y directions to be specified

6.2.2. Potential Field for Optimal Position Estimation with Absolute Position Information

The potential field defined in (6.4) is now modified to incorporate the absolute position information available for UAV. A new term is added to the already existing potential field for UGS nodes in (6.4) to obtain the potential field for UAVs, which is given as

$$V_{uav} = \frac{1}{2}\sum_{p=1}^{m}\sum_{j=1}^{N} K_{i_p j}{}^a (r_{i_p j}{}^a - \bar{r}_{i_p j}{}^a)^2 + \frac{1}{2}\sum_{p=1}^{m} K_{i_p}{}^a \left\| e_{i_p}{}^a \right\|^2 \tag{6.32}$$

where $e_{i_p}{}^a = \left[(x_{i_p}{}^a - \bar{x}_{i_p}{}^a)^2 + (x_{i_p}{}^a - \bar{y}_{i_p}{}^a)^2 \right]^{1/2}$ and $\bar{X}_{i_p}{}^a = \left[\bar{x}_{i_p}{}^a \quad \bar{y}_{i_p}{}^a \right]$ is the known absolute position of UAV ${i_p}^a$. Therefore, the new potential field for the air-ground sensor network with UAVs and UGS nodes is now defined as

$$V_p = V_{uav} + V_{ugs} \tag{6.33}$$

$$V_p = \frac{1}{2}\sum_{p=1}^{m} K_{i_p}{}^{a}\left\|e_{i_p}{}^{a}\right\|^2 + \frac{1}{2}\sum_{p=1}^{m}\sum_{j=1}^{N} K_{i_p j}{}^{a}(r_{i_p j}{}^{a} - \bar{r}_{i_p j}{}^{a})^2 + \frac{1}{2}\sum_{p=m+1}^{N}\sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \bar{r}_{i_p j})^2 \tag{6.34}$$

where $r_{i_p j}{}^{a} = [(x_{i_p}{}^{a} - x_j)^2 + (y_{i_p}{}^{a} - y_j)^2]^{1/2} = \left\|X_{i_p}{}^{a} - X_j\right\|$.

Now, define the potential field for a single UAV $i_p{}^a$ with absolute position measurement by

$$V_{i_p}{}^{a} = \frac{1}{2}K_{i_p}{}^{a}\left\|e_{i_p}{}^{a}\right\|^2 + \frac{1}{2}\sum_{\substack{j=1\\ j\neq i_p}}^{N} K_{i_p j}{}^{a}(r_{i_p j}{}^{a} - \bar{r}_{i_p j}{}^{a})^2 \tag{6.35}$$

The potential V_{i_p} for a single UGS node i_p without absolute position measurement is same as in (6.5). The gradient of the potential with respect to the UAV node states $i_p{}^a$ is given by

$$\frac{\partial V_{i_p}{}^{a}}{\partial X_{i_p}{}^{a}} = K_{i_p}{}^{a}(X_{i_p}{}^{a} - \bar{X}_{i_p}{}^{a}) - \sum_{j=1}^{N} K_{i_p j}{}^{a}(r_{i_p j}{}^{a} - \bar{r}_{i_p j}{}^{a})\frac{(X_{i_p}{}^{a} - X_j)}{\left\|X_{i_p}{}^{a} - X_j\right\|} \tag{6.36}$$

where as, the gradient of the potential $\left(\frac{\partial V_{i_p}}{\partial X_{i_p}}\right)$ with respect to the UGS node state i_p is same as in (6.7).

Based on the first assumption we consider only *x* and *y* positions of the UAVs and the second assumption allows us to consider UAV as a sensor node with absolute position information until the absolute localization algorithm has been executed.

Theorem 6.2: Consider the position estimate dynamics in (6.28) for each UGS node and (6.30) for each UAV with absolute position information. Let the virtual force for the UGS node and UAVs be given respectively as

$$f_{i_p} = \sum_{j=1}^{N} K_{i_p j}(r_{i_p j} - \bar{r}_{i_p j}) \frac{(X_{i_p} - X_j)}{\left\| X_{i_p} - X_j \right\|} - K_v \dot{X}_{i_p} \tag{6.37}$$

$$f_{i_p}{}^a = -K_{i_p}{}^a (X_{i_p}{}^a - \bar{X}_{i_p}{}^a) - \sum_{j=1}^{N} K_{i_p j}{}^a (r_{i_p j}{}^a - \bar{r}_{i_p j}{}^a) \frac{(X_{i_p}{}^a - X_j)}{\left\| X_{i_p}{}^a - X_j \right\|} - K_v{}^a \dot{X}_{i_p}{}^a \tag{6.38}$$

Then the position estimates reach steady-state values that provide optimal estimates of the actual absolute localization of the nodes in the sense that V_p is minimized.

Proof: Define the Lyapunov function as

$$L_p = V_p + \frac{1}{2} \sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} \dot{X}_{i_p}{}^a + \frac{1}{2} \sum_{p=m+1}^{N} \dot{X}_{i_p}{}^T \dot{X}_{i_p} \tag{6.39}$$

Differentiate to obtain

$$\dot{L}_p = \dot{V}_p + \sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} \ddot{X}_{i_p}{}^a + \sum_{p=m+1}^{N} \dot{X}_{i_p}{}^T \ddot{X}_{i_p} \tag{6.40}$$

One can compute

$$\begin{aligned} \dot{V}_p = {} & \sum_{p=1}^{m} K_{i_p}{}^a \dot{X}_{i_p}{}^{a^T} (X_{i_p}{}^a - \bar{X}_{i_p}{}^a) + \sum_{p=1}^{m} \sum_{j=1}^{N} K_{i_p j}{}^a (r_{i_p j}{}^a - \bar{r}_{i_p j}{}^a) \dot{X}_{i_p}{}^{a^T} \frac{(X_{i_p}{}^a - X_j)}{\left\| X_{i_p}{}^a - X_j \right\|} \\ & + \sum_{p=m+1}^{N} \sum_{j=1}^{N} K_{i_p j} (r_{i_p j} - \bar{r}_{i_p j}) \dot{X}_{i_p}{}^T \frac{(X_{i_p} - X_j)}{\left\| X_{i_p} - X_j \right\|} \end{aligned} \tag{6.41}$$

and on substitution of (6.30), (6.32) and (6.46) in (6.45) we get

$$\dot{L}_p = \sum_{p=1}^{m} K_{i_p}{}^{a} \dot{X}_{i_p}{}^{a^T} (X_{i_p}{}^{a} - \bar{X}_{i_p}{}^{a}) + \sum_{p=1}^{m}\sum_{j=1}^{N} K_{i_p j}{}^{a} (r_{i_p j}{}^{a} - \bar{r}_{i_p j}{}^{a}) \dot{X}_{i_p}{}^{a^T} \frac{(X_{i_p}{}^{a} - X_j)}{\left\| X_{i_p}{}^{a} - X_j \right\|}$$
$$+ \sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} f_{i_p}{}^{a} + \sum_{p=m+1}^{N}\sum_{j=1}^{N} K_{i_p j} (r_{i_p j} - \bar{r}_{i_p j}) \dot{X}_{i_p}{}^{T} \frac{(X_{i_p} - X_j)}{\left\| X_{i_p} - X_j \right\|} + \sum_{p=m+1}^{N} \dot{X}_{i_p}{}^{T} f_{i_p} \tag{6.42}$$

Further substituting the force inputs mentioned in (6.37) and (6.38) yields

$$\dot{L}_p = -\sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} K_v{}^{a} \dot{X}_{i_p}{}^{a} - \sum_{p=m+1}^{N} \dot{X}_{i_p}{}^{T} K_v \dot{X}_{i_p} \tag{6.43}$$

clearly, $\forall\ (K_v, K_v{}^{a}) > 0$, $\dot{L}_p \le 0$ and the vector $\begin{bmatrix} X_{i_p} & \dot{X}_{i_p} \end{bmatrix}^T$ and $\begin{bmatrix} X_{i_p}{}^{a} & \dot{X}_{i_p}{}^{a} \end{bmatrix}^T$ is bounded which shows that the system is SISL. Evaluating $\ddot{L}_p$ yields

$$\ddot{L}_p = -2\sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} K_v{}^{a} \ddot{X}_{i_p}{}^{a} - 2\sum_{p=m+1}^{N} \dot{X}_{i_p}{}^{T} K_v \ddot{X}_{i_p} \tag{6.44}$$

which on substitution of (6.28) and (6.30) gives

$$\ddot{L}_p = -2\sum_{p=1}^{m} \dot{X}_{i_p}{}^{a^T} K_v{}^{a} f_{i_p}{}^{a} - 2\sum_{p=m+1}^{N} \dot{X}_{i_p}{}^{T} K_v f_{i_p} \tag{6.45}$$

Using the result from the Lyapunov analysis that the vector $\begin{bmatrix} X_{i_p} & \dot{X}_{i_p} \end{bmatrix}^T$ and $\begin{bmatrix} X_{i_p}{}^{a} & \dot{X}_{i_p}{}^{a} \end{bmatrix}^T$ is bounded also yields that $\ddot{L}_p$ is also bounded. By Barbalat's Lemma [105] we deduce that $\dot{L}_p \to 0$ as $t \to \infty$, which yields $\dot{X}_{i_p}{}^{a} \to 0$ and $\dot{X}_{i_p} \to 0$ as $t \to \infty$. Therefore (6.28) shows that $\vec{f}_{i_p}$ goes to zero $\forall i_p$ with no absolute position information and (6.30) shows that $\vec{f}_{i_p}{}^{a}$ goes to zero $\forall i_p$ with absolute position information. Finally

(6.37) and (6.38) shows that $\frac{\partial V_{i_p}}{\partial X_{i_p}} \to 0$ and $\frac{\partial V_{i_p}{}^a}{\partial X_{i_p}{}^a} \to 0$ respectively, so V_p reaches a minimum. ■

6.2.3. Absolute Localization Algorithm

We now present an algorithm to dynamically localize the UGS network and uniquely determine the absolute position estimates of all the UGS nodes with the aid of at least three UAV with GPS. Before we propose the algorithm, let us look at a potential problem of the UGS network being inverted (i.e. upside down) in its configuration.

In any network, if only one node has GPS information, the network can be uniquely localized with respect to an absolute coordinate frame modulo a rotation and an inversion (i.e. flip the net over). If two nodes have GPS, then the rotational uncertainty is removed, but the net can still be 'upside down'. Adding a third node with GPS removes this final uncertainty, and results in a correctly localized net in absolute coordinates.

To solve the 'upside down' problem for the sensor network, there must be at least 3 UAVs with absolute position information available. Unfortunately, when the third node with absolute position information is added to the WSN, the estimated positions in the WSN may already correspond to the inverted situation.

The upside down problem can be confronted as follows. According to our algorithm, one UAV is added at a time to the air-ground sensor network. Let $i^{th};j^{th};k^{th}$ be the 3 UAVs to be added in that particular order to the network. Before the k^{th} UAV is added, its initial position co-ordinates obtained from (6.26) using only range measurement is compared with its absolute position information and the error is defined as

$$\varepsilon = \left\| X_{i_k}{}^{a} - \overline{X}_{i_k}{}^{a} \right\| \tag{6.46}$$

If $\varepsilon > \varepsilon_M$, (where ε_M is the known maximum range error depending on the types of system used), then the network is assumed to have improper position estimates due to the estimated network being inverted. To flip the estimated network positions upside down to the correct configuration, the orthogonal projection of all the UGS nodes already added to the network is taken on the line formed by the known positions of i^{th} and j^{th} UAVs. The projection of the UGS nodes across that line [87] is given as

$$\begin{bmatrix} x^o \\ y^o \end{bmatrix} = \frac{1}{a^2+b^2} \begin{bmatrix} b^2-a^2 & -2ab \\ -2ab & a^2-b^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 2ac \\ 2ac \end{bmatrix} \tag{6.47}$$

where, (x, y) are the initial points and (x^o, y^o) are the final points after projection and

$$a = y_j - y_i \,;\, b = x_i - x_j \,;\, c = (x_j - x_i)y_i - (y_j - y_i)x_i \tag{6.48}$$

Table 6.2 Absolute Localization Algorithm

```
1     Relatively Localize the UGS nodes using Algorithm 1 for
      relative localization mentioned in section 6.1.4.
2     Initialize the UAV position using (6.26) from the range
      measurement information available
  2.1 Increment $i$ by 1 to keep a count of the number of UAVs with
      absolute position information
      2.1.1 **if** $i = 3$
      2.1.2 use (6.46) to calculate $\varepsilon$
            2.1.2.1 **if** $\varepsilon > \varepsilon_M$
            2.1.2.2 Take orthogonal projection using (6.47) of UGS
                    nodes on the line formed with first 6 UAVs having
                    absolute position information
            2.1.2.3 Dynamically localize the air-ground sensor
                    network with (6.37) as the control input for UGS
                    nodes and (6.38) as the control input for the
                    thirdUAV.
            2.1.2.4 **end if**
      2.1.3 **else**
      2.1.4 Dynamically localize the air-ground sensor network with
            (6.37) as the control input for UGS nodes and (6.38) as the
            control input for UAV.
      2.1.5 **end if**
7     Repeat step 2 until all the 3 UAVs have been added
```

6.2.4. Absolute Localization Simulation Results

Here, we present simulation results for absolute localization. The air-ground sensor network model used for simulation consists of 7 UGS nodes and a single UAV with on-board GPS. Once the UGS nodes have been relatively localized using the algorithm mentioned in Section 6.1.4, the UAV hover over the terrain to absolutely localize the sensor network. The UAV stops at 3 different positions and at each position it measures the distance with at least 3 UGS nodes as shown in Figure 6.8.

Figure 6.9 shows the virtual movement of the UGS nodes when the UAV stops at 2 different positions. The UGS nodes move in a way to obtain its absolute coordinates. When the UAV moves to a third position, it estimates its position using (6.26) and then compares it with its position obtained from GPS. If the error obtained is more then ε_M then all the UGS nodes are reflected on the line formed by the points when UAV stopped at position 1 and 2 which is shown in Figure 6.10.

The reflection is done in order to solve the upside-down problem as mentioned in Section 6.2.3. Figure 6.11 shows the plot for the potential field function versus the number of iterations taken before the potential field converges to zero. The plot shows that the absolute localization algorithm estimates the UGS nodes position such that the desired distance is achieved.

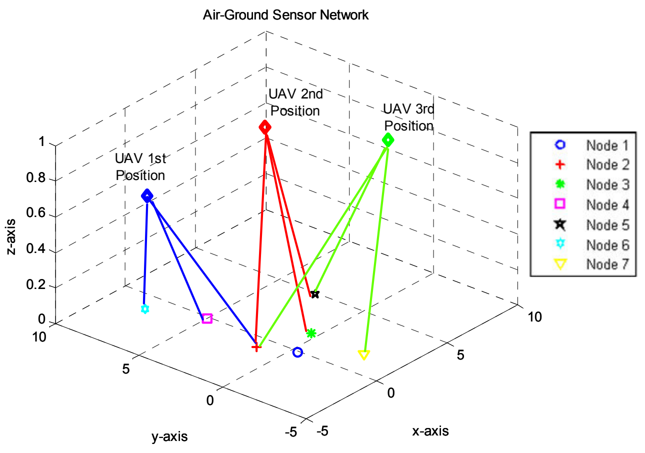


Figure 6.8 Air-Ground Sensor Network

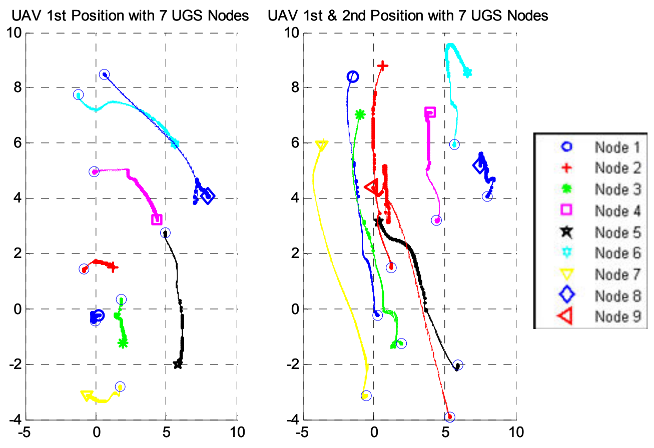


Figure 6.9 UAV 1 and 2 with 7 UGS nodes

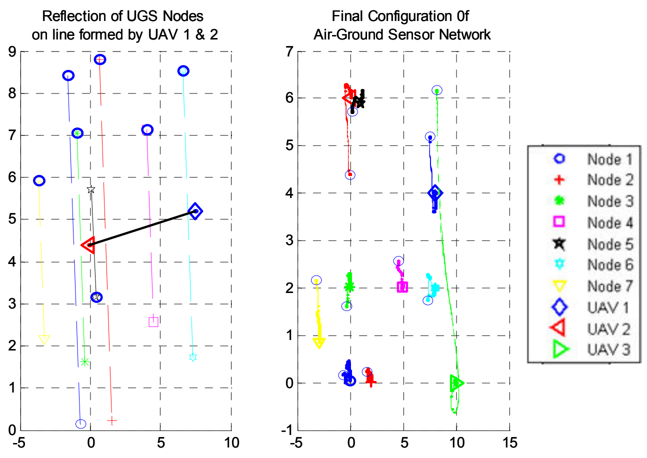


Figure 6.10 Final Configuration of UGS nodes

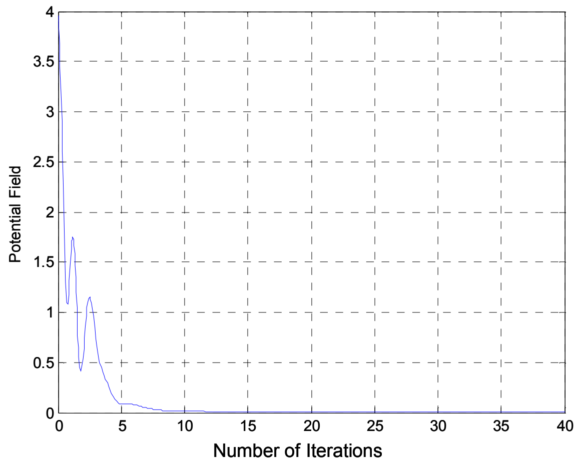


Figure 6.11 Potential field Function Plot

6.2.5. Absolute Localization Experimental Results

Here, we present the real time implementation result for absolute localization. Once the UGS nodes have been relatively localized as mentioned in Section 6.1.4, the UAV hovers over the terrain to absolutely localize the sensor network. The UAV stops at 3 different positions and at each position it measures the distance with atleast 3 UGS nodes as shown in Figure 6.12.

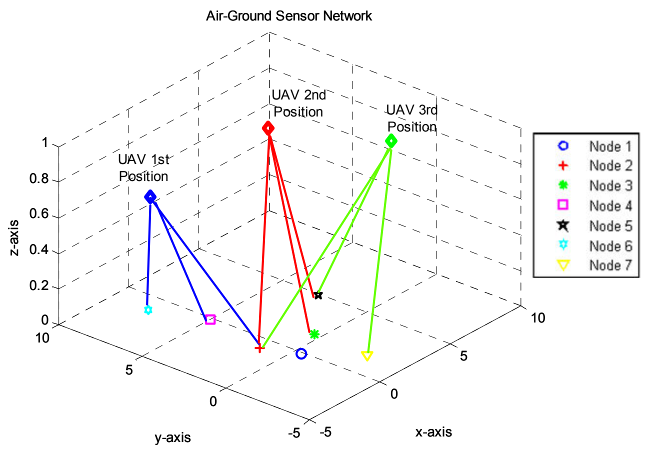


Figure 6.12: Air Ground Sensor Network Model

Figure 6.13 shows the virtual movement of the UGS nodes when the UAV stops at 3 different known locations to absolutely localize the UGS nodes. Figure 6.14 shows the final absolute co-ordinates of the UGS nodes when the 3 UAV hovering positions are given as $(34.3,138.6)$, $(167.3,35.9)$, $(-94.4,-42.93)$ cm respectively.

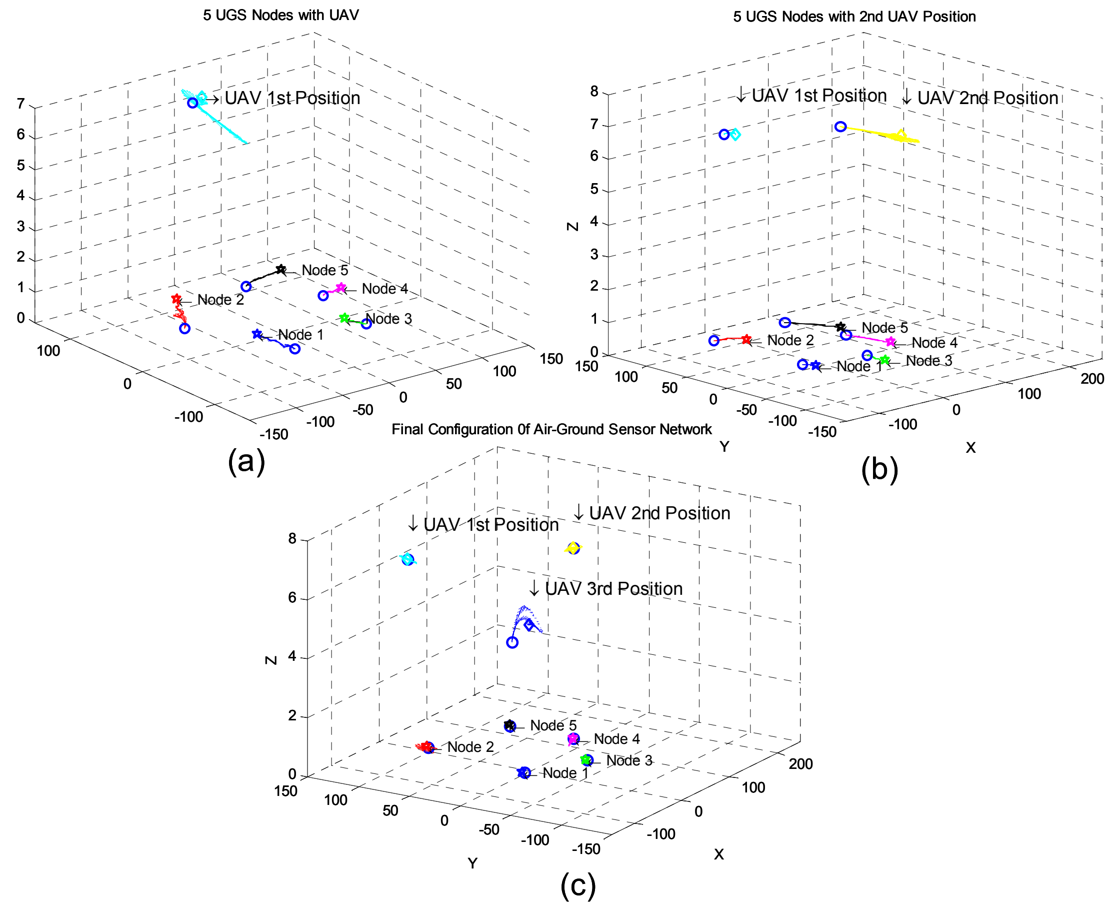


Figure 6.13: UAV with 5 UGS nodes (a) with UAV's 1st position (b) with UAV's 2nd position (c) with UAV's 3rd position

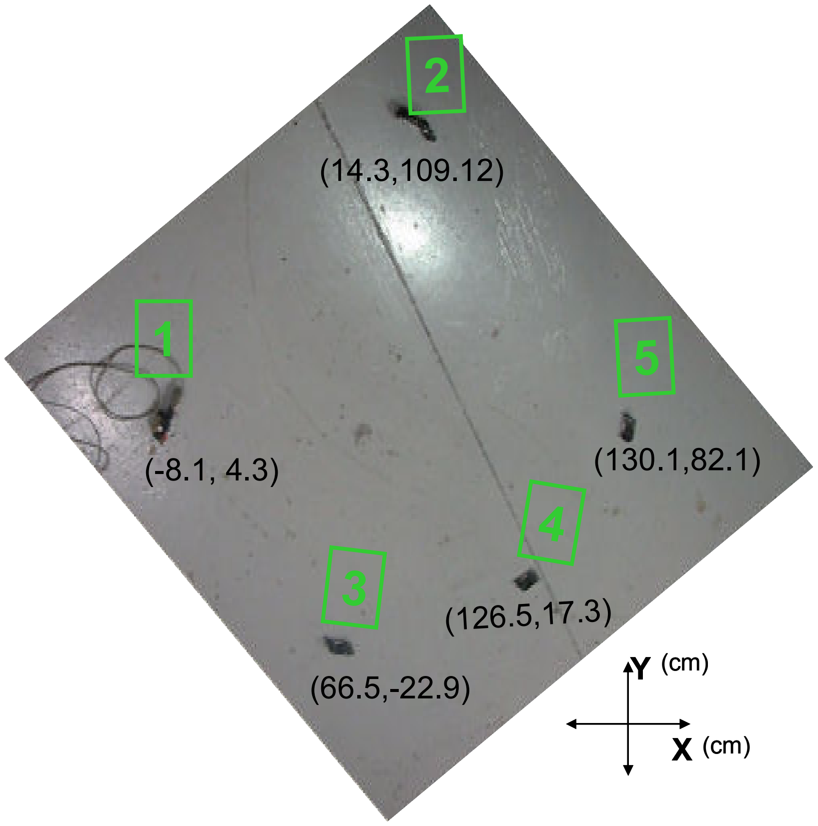


Figure 6.14: Final configuration of UGS nodes with absolute position (all units are in cm)

6.3 Extended Kalman Filter for Relative and Absolute Localization

In this section an extended Kalman Filter (EKF) is presented to take the effect of noise during real time implementation. The EKF is combined with the localization algorithm developed earlier in the literature. Figure 6.15 lays out the structure of the overall scheme

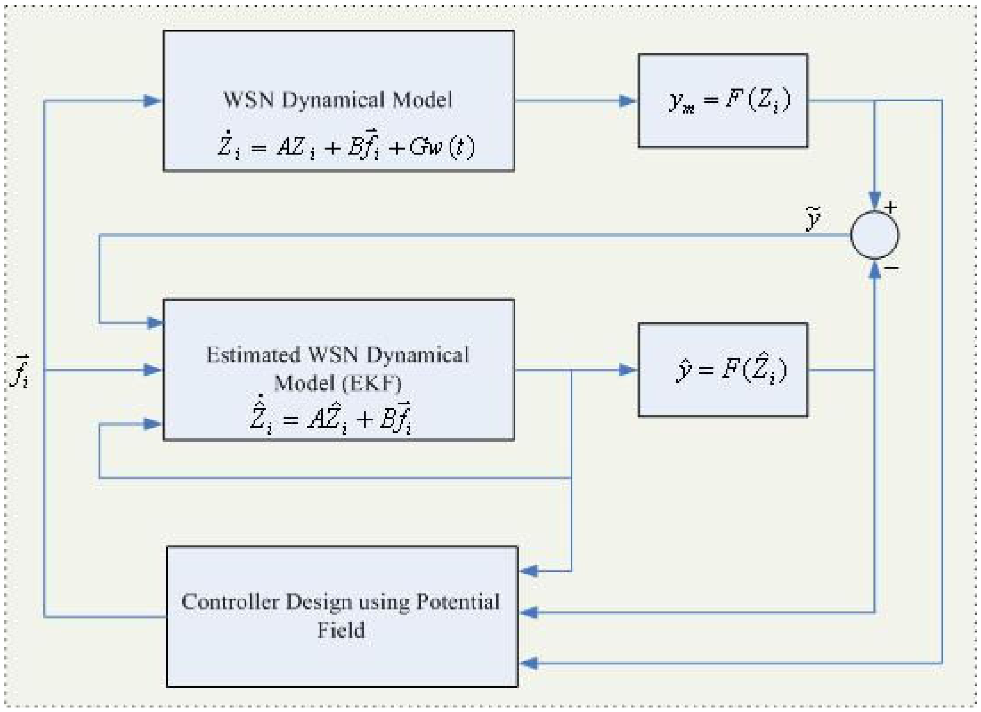

WSN Dynamical Model
$\dot{Z}_i = AZ_i + B\vec{f}_i + Gw(t)$
$y_m = F(Z_i)$
$\tilde{y}$
+
−
$\vec{f}_i$
Estimated WSN Dynamical Model (EKF)
$\dot{\hat{Z}}_i = A\hat{Z}_i + B\vec{f}_i$
$\hat{y} = F(\hat{Z}_i)$
Controller Design using Potential Field

Figure 6.15 UAV Localization scheme combined with EKF

For Kalman filter implementation, a system model and a measurement model is required. The system model is derived as

<u>System Model</u>

$$X_i = [x_i \quad y_i]^T \tag{6.49}$$

where, x_i and y_i are the *x-y* coordinates for the UGS node position estimate. The position estimation dynamics are given as

$$\ddot{X}_i = \vec{f}_i \tag{6.50}$$

where, $\vec{f}_i = \left[f_i^x \quad f_i^y\right]^T$ is the virtual force in the *x* and *y* directions to be specified. The state variable description form for the position estimate of the i^{th} UGS node is by

$$\begin{bmatrix} \dot{X}_i \\ \ddot{X}_i \end{bmatrix} = \begin{bmatrix} O_2 & I_2 \\ O_2 & O_2 \end{bmatrix} \begin{bmatrix} X_i \\ \dot{X}_i \end{bmatrix} + \begin{bmatrix} O_2 \\ I_2 \end{bmatrix} \begin{bmatrix} f_i^x \\ f_i^y \end{bmatrix} \tag{6.51}$$

where, $O_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$\dot{X}_i = AX_i + Bf_i + Gw(t) \tag{6.52}$$

where $w(t) \sim N(0,Q)$

Measurement Model

$$Z = h(X_i, X_j) = r_{ij} = \sqrt{(x_i - x_j)^{\wedge 2} + (y_i - y_j)^{\wedge 2}} \tag{6.53}$$

where Z is the set of range measurements and $v(t) \sim N(0,R)$.

Setting up the Kalman Filter yields the following equations for gain, system update and propagation.

Gain

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{6.54}$$

where $H_k = \left. \dfrac{\partial h(X_i, X_j)}{\partial X_i} \right|_{X_{i_k}^-}$

System Update

$$\hat{X}_{i_k}^+ = \hat{X}_{i_k}^- + K_k [Z - \hat{Z}] \tag{6.55}$$

Covariance Update

$$P_k^+ = (I - K_k H_k) P_k^- \tag{6.56}$$

<u>System Propagation</u>

$$\dot{\hat{X}}_i = A\hat{X}_i + Bf_i \tag{6.57}$$

<u>Covariance Propagation</u>

$$\dot{P} = AP + PA^T + GQG^T \tag{6.58}$$

Simulating the EKF for 3 node case yields the following equations for measurement equation and gain matrix

$$Z = \begin{bmatrix} r_{12} & r_{13} & r_{21} & r_{23} & r_{31} & r_{32} \end{bmatrix} \tag{6.59}$$

$$H_k = \begin{bmatrix} \frac{x_1 - x_2}{r_{12}} & \frac{y_1 - y_2}{r_{12}} & 0 & 0 & -\frac{x_1 - x_2}{r_{12}} & -\frac{y_1 - y_2}{r_{12}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{x_1 - x_3}{r_{13}} & \frac{y_1 - y_3}{r_{13}} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{x_1 - x_3}{r_{13}} & -\frac{y_1 - y_3}{r_{13}} & 0 & 0 \\ \frac{x_1 - x_2}{r_{21}} & \frac{x_1 - x_2}{r_{21}} & 0 & 0 & -\frac{x_1 - x_2}{r_{21}} & -\frac{y_1 - y_2}{r_{21}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{x_2 - x_3}{r_{23}} & \frac{y_2 - y_3}{r_{23}} & 0 & 0 & -\frac{x_2 - x_3}{r_{23}} & -\frac{y_2 - y_3}{r_{23}} & 0 & 0 \\ \frac{x_1 - x_3}{r_{31}} & \frac{y_1 - y_3}{r_{31}} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{x_1 - x_3}{r_{31}} & -\frac{y_1 - y_3}{r_{31}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{x_2 - x_3}{r_{32}} & \frac{y_2 - x_3}{r_{32}} & 0 & 0 & -\frac{x_2 - x_3}{r_{32}} & -\frac{y_2 - x_3}{r_{32}} & 0 & 0 \end{bmatrix} \tag{6.60}$$

The simulation results are shown in the Figure 6.16 and Figure 6.17. The results shown below highlights the effectiveness of the EKF implemented on the localization method developed in this literature. The blue lines in the plot indicate the measurements taken, whereas the red lines indicate the estimated obtained from the EKF method. The convergence of the estimate with time indicates the effectiveness of the EKF method proposed in the literature.

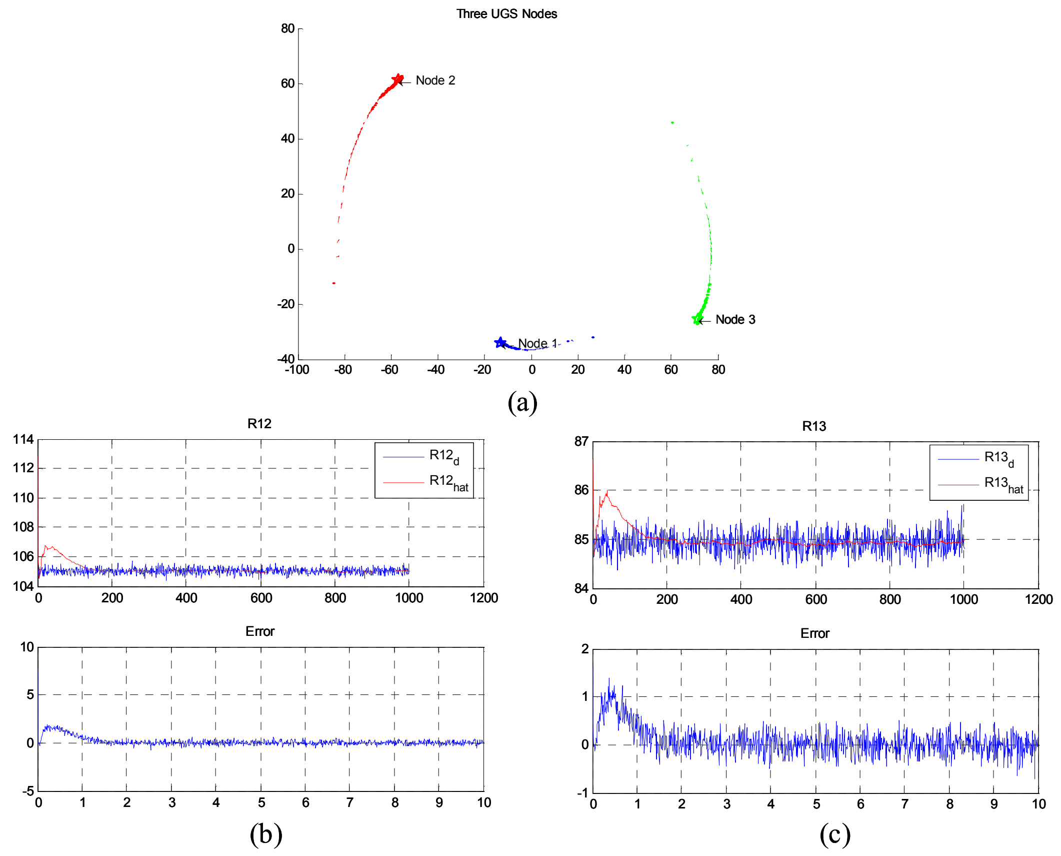


Figure 6.16: Plot of the range estimates (a) final configuration of 3 sensor nodes (b) range estimate r12 (red) converging to r12d (blue) (c) range estimate r13 (red)converging to r13d (blue)

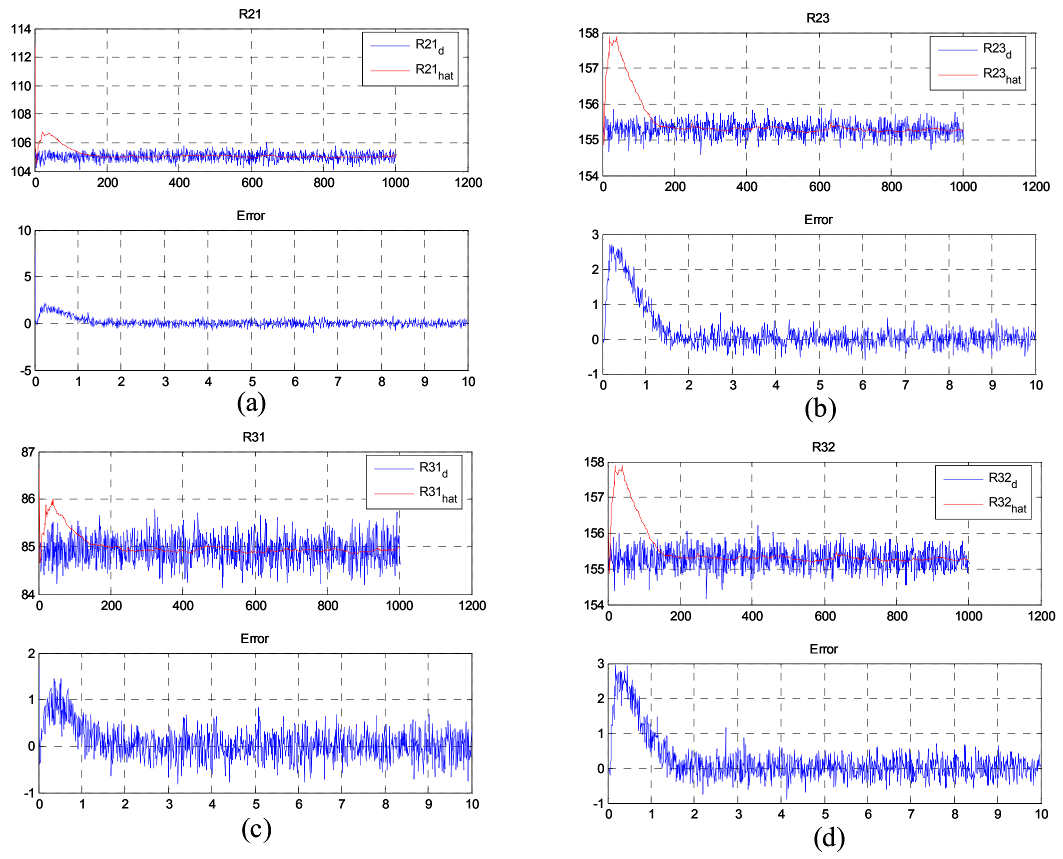


Figure 6.17: Plot of the range estimates (a) range estimate r21 (red)converging to r21d (blue) (b) range estimate r23 (red)converging to r23d (blue) (c) range estimate r31 (red)converging to r31d (blue) (d) range estimate r32 (red)converging to r32d (blue)

6.4 Conclusion

Efficient algorithms for relative and absolute localization are presented based on potential field methods, together with the mathematical analysis. The algorithms mentioned are successfully implemented on a real-time hardware. The results obtained validate the efficiency of the relative and absolute localization algorithm proposed. The algorithms presented take care that the system does not fall into the local minima. The algorithm also takes into account the problem of the network being in an inverted configuration during absolute localization.

EKF is also developed for relative and absolute localization the sensor nodes and is effectively simulated to show the effectiveness of the proposed method.

REFERENCES

[1] Ge, S.S., Cui, Y.J., “Dynamic motion planning for mobile robots using potential field method,” Autonomous Robots, vol. 13, pp. 207-222, 2002.

[2] Kumar, V., Zefran, M, Ostrowski, J, “Motion planning and control of robots,” Handbook of industrial robots, J. Wiley and sons, Oct. 1997.

[3] Khatib, O., “Real-Time obstacle avoidance for manipulators and mobile robots,” International Journal of Robotic Research, vol. 5, no. 1 pp. 90, 1986.

[4] Ge, S.S., Cui, Y.J., “Path planning of mobile robots using new potential functions,” Proc. Of Asian Control Conference, Shanghai, 2000.

[5] Rimon, E., Koditchek, D.E., “Exact robot navigation using artificial potential functions,” IEEE Transactions on Robotics and Automation, vol. 8, no, 5, 1992.

[6] Kim, J., “Real-Time obstacle avoidance using harmonic potential functions,” IEEE Transactions on Robotics and Automation, 1992.

[7] Volpe, R., Khosla, P., “Manipulator control with superquadratic artificial potential functions: Theory and experiments,” IEEE transactions on Systems, Man,, and Cybernatics, vol. 20, no. 6, 1990.

[8] Ge, S.S., Cui, Y.J., “New Potential functions for mobile robots path planning,” IEEE Transactions on Robotics and Automation, vol. 16, no, 5, pp. 615, 2000.

[9] Kim, J.O., Khosla, P., “Real-Time obstacle avoidance using harmonic potential field functions,” IEEE Conference on Robotics and Automation, Sacramento, CA, pp.790-796, 1991.

[10] Connolly, C.I., “Applications of harmonic functions to robotics,” Proc. of IEEE International Symposium on Intelligent Control, pp. 498-502, 1992.

[11] Hu. Y.R., Vukovich G., “Active Robust shape control of flexible structures,” Journal of Sound and Vibration, vol. 15, issue 7, pp. 807-820, 2005.

[12] Cummings, R.E., Pourboghart F., Maruboyina H.K., Abrate, S., Dhali, S., “Architecture for distributed actuation and sensing using smart piezoelectric elements”, SPIE Conference on Smart Structures and Integrated Systems, vol. 3329, 1998.

[13] Hughes D., Wen J.T., Beseler J., Chow J., “Modeling, Identification and control of flexible structures,” Computer Integrated Manufacturing, 1990., Proceedings of Rensselaer's Second International Conference, vol., no., pp.412-417, 21-23 May 1990.

[14] Akella, P., Chen, X., Cheng, W., Hughes, D., Wen, J.T., “Modeling and control of smart structures with bonded piezoelectric sensors and actuators,” Smart Mater. Struct. 3 344-353, 1994.

[15] Leleu, S., Abou-Kandil, H. Bonnassieux, Y., “Piezoelectric actuators and sensors location for active control of flexible structures” Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, 2000. IMTC 2000.

[16] Sadri, A., M. Wright, J. R., Wynne, R. J., “Modelling and optimal placement of piezoelectric actuators in isotropic plates using genetic algorithms” in Journal of Smart Materials And Structures, 1999, Vol 8; Number 4, pages 490-498.

[17] Han, J.H., Lee, I., “Optimal Placement Of Piezoelectric Sensors And Actuators For Vibration Control of a Composite Plate Using Genetic Algorithms” in Journal of Smart Materials And Structures, 1999, Vol 8; pages 257-267.

[18] Tong, D., Williams, R. L., Agrawal, S. K., “Optimal Shape Control of Composite Thin Plates with Piezoelectric Actuators”, In Journal Of Intelligent Material Systems And Structures, 1998, Vol 9; Number 6, pages 458-467.

[19] Quek, S.T., Wang, S.Y., Ang, K.K, “Vibration Control Of Composite Plates Via Optimal Placement Of Piezoelectric Patches”, in Journal of Intelligent Material Systems and Structures, 2003, Vol 14, pages 229-245.

[20] Padual, S.L., Kincaid, R.K., “Optimization Strategies for Sensors and Actuators Placement”, National Aeronautics and Space Administration, NASA/TM-1999-209126, Langely research, Langely Virginia 23681, USA.

[21] Onada, J., Hanawa, Y., “Actuator Placement optimization by genetic and improved Simulated Annealing Algorithms”, AIAA Journal, pages 1167-1169.

[22] Nguyn, Q., Tong, L., “Shape control of smart composite plate with non-rectangular piezoelectric actuators”, in Journal of Composite Structures, 2004, Vol. 66, Issues, 1-4.

[23] Da Mota Silva, S., Ribeiro, R., Dias, J. R., Vaz, M., “Shape control of structures with PZT actuators using genetic algorithms”, Proceedings of 13th Inter. Conf. on Composite Materials, Beijing, 2001.

[24] Z. Uykan and C. Güzelis, “Input–output clustering for determining the. centers of radial basis function network”, in Proc. Turkish Symposium on Signal Processing and its Applications, Antaalya, Turkey, April 1996, pages 677-682.

[25] Uykan, Z. Guzelis, C. Celebi, M. E. Koivo, H. N., “Analysis of Input-Output Clustering for Determining Centers of RBFN”, in IEEE Transactions On Neural Networks, 2000, VOL 11; PART 4, pages 851-858.

[26] Chen, S., Mulrew, B., Grant, P.M., “A clustering Technique for Digitial Communications Channel Equalization Using Radial basis Function Networks”, in IEEE Transaction on Neural Networks, 1993, Vol 4, pages 570-579.

[27] Zhao, W., Huang, D.S., “The Structure Optimization of Radial Basis Probabilistic Neural Networks Based on Genetic Algorithms”, in Proceedings of World Congress on Computational Intelligence, 2004, Hawaii, pages 1086-1091.

[28] Chen, S., Cowan, C.F.N., Grant, P.M., “Orthogonal least squares learning algorithm for radial basis function networks”, in IEEE Transaction on Neural Networks, 1991, Vol. 2, pages 302-309.

[29] Mehrabian, A., “Communication without words”, Psychology today, vol. 2, no. 4, pp 52-55, 1968.

[30] Ekman, P., Friesen, W., “Facial Action coding system”, Palo Alto, CA,: Consulting Physcologists Press, 1978.

[31] Lien, J.J.; Kanade, T.; Cohn, J.F.; Ching-Chung Li, “Automated facial expression recognition based on FACS action units”, Proceedings, Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998. Page(s): 390 – 395.

[32] Pantic, M.; Patras, I, “Dynamics of facial expression: recognition of facial actions and their temporal segments from face profile image sequences”, IEEE Transaction on System, Man and Cybernatics, 36(2006), 433-439.

[33] Khan, M.M.; Ward, R.D.; Ingleby, M., “Infrared Thermal Sensing of Positive and Negative Affective States”, Robotics, Automation and Mechatronics, IEEE Conference on, page(s), Dec 2006.

[34] Anderson, K.; McOwan, P.W., “A real-time automated system for the recognition of human facial expressions”, IEEE Transaction on System, Man and Cybernatics, 36(2006), 96-105.

[35] Zheng, W., Zhou, X., Zou, C., Zhao, L., “Facial Expression Recognition Using Kernel Canonical Correlation Analysis (KCCA)”, IEEE Transaction on Neural Networks, 17(2006), 233-238.

[36] Feitosa, R. Q., Vellasco, M. M. B. R., Andrade, D. V., Maffra, S. A. R. S., Oliveira, D. T., “Facial Expression Classification Using RBF and Back-Propagation Neural Networks”, Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2000) and the 6th International Conference on Information Systems Analysis and Synthesis (ISAS'2000), Orlando, USA, August 2000, pp. 73-77.

[37] Lisetti, C.L., Rumelhart, D.E., “Facial Expression Recognition using a Neural Network” In Proceedings of the 11th International Flairs Conference. AAAI Press, 1998.

[38] Padgett, C, Cottrell, G.. “Representing face images for emotion classification.” in M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems, volume 9, Cambridge, MA, 1997. MIT Press.

[39] Ma, L., Khorasani, K., Van Blyenburgh P, “Facial expression recognition using constructive feedforward neural networks Systems, Man and Cybernetics, Part B, IEEE Transactions on, page(s): 1588-1595 Volume: 34, Issue: 3, June 2004.

[40] Rosenblum, M., Yacoob, Y., Davis, L.S., “Human expression recognition from motion using a radial basis function network architecture”, IEEE Transactions On Neural Network, 7(5):1121–1138, 1996.

[41] Ham, F.M. and Acharyya, R., “A Universal Neural Network-Based Infrasound Event Classifier”, C.H. Chen Editor: chapter 3, Signal and Image Processing for Remote Sensing, pp. 33-54, CRC press, Taylor & Francis.

[42] Yale image database http://cvc.yale.edu/projects/yalefaces/yalefaces.html.

[43] Jain, A. K., “Fundamentals of digital image processing”, Prentice Hall; US Ed edition 1988.

[44] Kraiss, K.F., Advanced Man-Machine Interaction*:* Fundamentals and Implementation. Springer, 2006.

[45] Ryu, Y. and Oh, S., “Automatic Extraction of Eye and Mouth Fields from a Face Image Using Eigenfeatures and Ensemble Networks” Applied Intelligence 17, 2 (Jul. 2002), 171-185.

[46] Bookstein, F. L. “Principal Warps: Thin-Plate Splines and the Decomposition of Deformations” IEEE Trans. Pattern Anal. Mach. Intell. 11, 6 (Jun. 1989), 567-585.

[47] Turk, M. A. and Pentland, A. P., “Face recognition using eigenfaces.” In IEEE Conference on Computer Vision and Pattern Recognition, pages 586—591.

[48] Turk, M. A. and Pentland, A. P. “Eigenfaces for recognition” Journal of Cognitive Neuroscience, 3(1), 1991.

[49] Sanger, T., “Optimal Unsupervised Learning in a Single Layer Linear Feedforward neural Network”, Neural Networks 12,459-473, 1989.

[50] Hagan, M.T, Demuth, H.B, Beale, M.H., “Neural Network Design”, 1st Edition PWS Publishing Co. , 1995.

[51] Ham, F. M., Kostanic, I., and Ham, F. M., “Principles of Neurocomputing for Science and Engineering” 1st. McGraw-Hill Higher Education, 2000.

[52] Haykin, S., Neural Netowrks: A Comprehensive Foundation, Prentice Hall, 2nd Edition, 1998.

[53] Rubinstein, Y.D.,Hastie, T., “Discriminative vs informative learning” In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pages 49--53, 1997.

[54] Hawickhorst, B. A., Zahorian, S. A., Rajagopal, R., "A Comparison of Three Neural Network Architectures for Automatic Speech Recognition," Intelligent Engineering Systems Through Artificial Neural Networks , Vol. 5, Fuzzy Logic and Evolutionary Programming, pp. 221-226, St. Louis, Missouri, November 1995.

[55] Chen, C., “Signal and image processing for remote sensing” CRC Press Technology & Industrial Arts 648 pages ISBN 0849350913, 2006.

[56] Trees, H., “Detection, Estimation, and Modulation Theory, Part I”, Wiley Interscience, ISBN 9780471095170, 2002.

[57] Yang, J. Yu, H., Kunz, W., “An Efficient LDA Algorithm for Face Recognition” 6th International Conference on control, Automation, Robotics and Vision (ICARCV2000).

[58] Muller-Schneiders, S.; Jager, T.; Loos, H.S.; Niem, W., “Performance evaluation of a real time video surveillance system,” Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on. 15-16 Oct. 2005., pp. 137-143.

[59] Oh, P. Y.; Stanciu, R. I., “Visual Servoing to Help Camera Operators Track Better,” in Proc. International Conf. Intelligent Robots and Systems, Beijing, China, Oct. 2006, pp. 7-7.

[60] Chaumette, F.; Hutchinson, S., “Visual Servo Control . I. Basic Approaches,” Robotics & Automation Magazine, IEEE, vol. 13, issue 4, Dec. 2006, pp. 82-90.

[61] Chaumette, F.; Hutchinson, S., “Visual Servo Control . II. Advanced Approaches [Tutorial],” Robotics & Automation Magazine, IEEE, vol. 14, issue 1, Mar. 2007, pp.109-118.

[62] Malis, E.; Chaumette, F.; Boudet, S., “2D ½ Visual Servoing Stability Analysis with Respect to Camera Calibration Errors,” Proceedings of the 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Victoria, B.C., Canada , Oct. 1998, pp. 691-697.

[63] Mittrapiyanuruk, P.; DeSouza, G.N.; Kak, A.C., “Calculating the 3D-pose of rigid-objects using active appearance models,” Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on , vol.5, no., pp. 5147-5152 Vol.5, 26 April-1 May 2004.

[64] Wunsch, P.; Winkler, S.; Hirzinger, G., “Real-time pose estimation of 3D objects from camera images using neural networks,” Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on , vol.4, no., pp.3232-3237 vol.4, 20-25 Apr 1997.

[65] Abidi, M.A.; Chandra, T., “Pose estimation for camera calibration and landmark tracking,” Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on , vol., no., pp.420-426 vol.1, 13-18 May 1990.

[66] Shakunaga, T., “An Object Pose Estimation System Using A Single Camera,” Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on , vol.2, no., pp.1053-1060, 7-10 Jul 1992.

[67] Wang, J.; Wilson, W.J., “3D relative position and orientation estimation using Kalman filter for robot control,” Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on , vol., no., pp.2638-2645 vol.3, 12-14 May 1992.

[68] Ying Kin Yu; Kin Hong Wong; Chang, M.M.Y., “Recursive three-dimensional model reconstruction based on Kalman filtering,” Systems, Man and Cybernetics, Part B, IEEE Transactions on , vol.35, no.3, pp. 587-592, June 2005.

[69] Lippiello, V.; Siciliano, B.; Villani, L., “Position and orientation estimation based on Kalman filtering of stereo images,” Control Applications, 2001. (CCA '01). Proceedings of the 2001 IEEE International Conference on , vol., no., pp.702-707, 2001.

[70] Ficocelli, M.; Janabi-Sharifi, F., “Adaptive filtering for pose estimation in visual servoing,” Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on , vol.1, no., pp.19-24 vol.1, 2001.

[71] Kay, Y.; Lee, S., “A robust 3-D motion estimation with stereo cameras on a robot manipulator,” Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on , vol., no., pp.1102-1107 vol.2, 9-11 Apr 1991.

[72] Lippiello, V.; Siciliano, B.; Villani, L., “3D pose estimation for robotic applications based on a multi-camera hybrid visual system,” Robotics and

Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on , vol., no., pp. 2732-2737, May 15-19, 2006.

[73] Lewis, F.L., Optimal Estimation, New York: John Wiley & Sons, 1986.

[74] Hutchinson, S.; Hager, G.D.; Corke, P.I., "A tutorial on visual servo control," Robotics and Automation, IEEE Transactions, vol.12, no.5, pp.651-670, Oct 1996.

[75] Jun-Ho Oh; David Hanson; Won-Sup Kim; Young Han; Jung-Yup Kim; Ill-Woo Park, "Design of Android type Humanoid Robot Albert HUBO," Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference vol., no., pp.1428-1433, Oct. 2006.

[76] Kraiss, K.F., Advanced Man-Machine Interaction: Fundamentals and Implementation. Springer, 2006.

[77] Ryu, Y. and Oh, S., "Automatic Extraction of Eye and Mouth Fields from a Face Image Using Eigenfeatures and Ensemble Networks" Applied Intelligence 17, 2 (Jul. 2002), 171-185.

[78] Fischler, M. A. and Bolles, R. C. 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,". Commun. ACM 24, 6 (Jun. 1981), 381-395.

[79] Dang, P., Lewis, F., .Stephanou, H. Ham, F., "Facial Expression Recognition using a Two Stage Neural Network", Proc. Mediterranean Conf. Control & Automation, Athens, Greece, June 2007.

[80] Wu, Y. and Hu, Z. 2006. "PnP Problem Revisited". *J. Math. Imaging Vis.* 24, 1 (Jan. 2006), 131-141.

[81] Cheng Li; Bin Liang; Wenfu Xu, "Autonomous Trajectory Planning of Free-floating Robot for Capturing Space Target," Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference , vol., no., pp.1008-1013, Oct. 2006.

[82] Singh, S.K.; Pieper, S.D.; Popa, D.O.; Guinness, J.; "Control and coordination of head, eyes and facial expressions of virtual actors in virtual environments," Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on 3-5 Nov. 1993 Page(s):335 – 339.

[83] Lewis, F.L., Dawson, D., Abdallah, C., "Robot Manipulator Control," Marcel Dekker, 2nd edition, 2004, ISBN 0-8247-4072-6.

[84] Lewis, F.L., Jagannathan, S., Yesildirek, Y., "Neural Network Control of Robot Manipulators and Nonlinear Systems," London, U. K.: Taylor and Francis, 1999.

[85] Ahmed AA, Shi H, Shang Y (2005) SHARP: a new approach to relative localization in wireless sensor networks. Proc. Of Distributed Computing Systems Workshops:892- 898.

[86] Bachrach J, Taylor C (2005) Localization in sensor networks. Handbook of Sensor Networks : Algorithms and Architectures, vol. 1.

[87] Bronshtein IN, Semendyayev KA, Musiol G, Muehlig H, Mühlig H (2003) Handbook of mathematics. Springer; 4th edition.

[88] Bulusu N, Heidemann J, Estrin D (2001) Gps-less low cost outdoor localization for very small devices. Proc. of 21st conference on distributed computing systems, Phoenix, AZ.

[89] Bulusu N, Heidemann J, Estrin D (2001) Adaptive Beacon Placement. Proc. of the the 21st International Conference on Distributed Computing Systems, p. 489.

[90] Bulusu N, Heidemann J, Estrin D (2005) Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine, vol. 7, pp. 28-34.

[91] Castillo P, Lozano R, Dzul A (2005) Stabilization of a Mini Rotorcraft with Four Robots. IEEE control system Magazine, pp: 45-55.

[92] Corke P, Peterson R, Rus D(2004) Coordinating Aerial Robots and Sensor Networks for Localization and Navigation. Proc. of 7th Int. Symp. on Distributed Autonomous Robotic Systems, Toulouse.

[93] Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. IEEE International Conference on Robotics and Automation. Vol. 2, pp:500 – 505.

[94] Kim J, Sukkarieh S (2003) Airborne simultaneous localization and map building. IEEE International Conference on Robotics and Automation. Vol. 1, pp:406 – 411.

[95] Kim J, Sukkarieh S (2004) SLAM aided GPS INS navigation GPA denied and unknown environments" Proc. Of international symposium on GNSS/GPS.

[96] Kim J, Ong LL, Nettleton E, Sukkarieh S (2004) Decentralized approach to unmanned aerial vehicle navigation: without the use of the global positioning system and preloaded maps. Proc. of the I MECH E Part G Journal of Aerospace Engineering, vol. 218, iss. 6, pp. 399-416(18).

[97] Langelaan J, Rock S (2004) Navigation of small UAVs operating in forests. Proc. of AIAA guidance, navigation and control conference.

[98] Luiz C, Grocholsky B, Keller JF, Kumar V, Taylor CJ (2004) "Experiments in Multirobot Air-Ground Coordination. IEEE International Conference on Robotics and Automation, New Orleans, LA.

[99] Moses RL, Krishnamurthy D, Patterson RM (2002) A self-localization method for wireless sensor networks. EURASIP Journal on Applied Signal Processing..

[100] Patwari N, Ash JN, Kyperountas S, Hero AO, Moses RL, Correal NS (2005) Locating the nodes: cooperative localization in wireless sensor networks. Signal Processing Magazine, IEEE, vol. 22, pp. 54-69.

[101] Paul AS, Wan EA (2005) Dual kalman filters for autonomous terrain aided navigation in unknown environments. Proc.of IEEE International Joint Conference on Neural Networks. Vol. 5, pp:2784 -2789.

[102] Savarese C, Rabaey JM, Beutel J (2001) Locationing in distributed ad-hocwireless sensor networks. Proc. Of IEEE International Conference on Acoustics, Speech, and Signal Processing.

[103] Semic S, Sastri S (2001) Distributed localization in wireless sensor networks. Tech. Rep. UCB/ERL M02/26, UC Berkeley..

[104] Shi Q, Huo H, Fang T, Li D (2005) Using space distance intersection for node location computation in wireless sensor networks. International workshop on sensor network and applications.

[105] Slotine J, Li W Applied Non-Linear Control, Prentice Hall.

[106] Van Blyenburgh P (1999) UAVs: An Overview. Air & Space Europe, vol. 1, issue 5, pp. 43-47(5) Elsevier Science.

[107] Z ou Y, Chakrabarty K (2003) Sensor deployment and target localization based on virtual forces. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. Vol. 2, pp:1293 – 1303.

BIOGRAPHICAL INFORMATION

Pritpal Dang received his Bachelor’s degree in Electrical Engineering from S. R. K. N. Engineering College, Nagpur, India in 2002. He received his Master’s of Science in Electrical Engineering in 2004 during which he worked as Research Assistant in Automation & Robotics Research Institute (ARRI). His research was concentrated on prototyping and implementation intelligent control algorithms for underactuated electromechanical systems. He then worked as a visiting instructor in Alcorn State University, MS, where he taught control system and robotics courses and developed a control system laboratory for undergraduate classes and also had been working as research associate for his PhD in ARRI. His research work includes distributed control, sensor networks and Electro-Mechanical systems.