ADAPTIVE SAMPLING WITH MOBILE WSN

by

KOUSHIL SREENATH

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

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

DECEMBER 2005

ACKNOWLEDGEMENTS

Well, I'm finally writing the acknowledgments. Whoa!! I was so far away and the distant horizon was just that - a distant horizon. But a light shone steadily on, despite my misgivings, despite my wrong doings, despite my attempts at burning the lab down. And all I did was to follow the path it lit.

It has been a tremendous honor to work with two great professors, Dr. Popa and Dr. Lewis. It's a miracle I survived the assault. They were always kind when I did something, and kind even when I did nothing at all.

Mr. Jyotirmay Gadewadikar has always been there silently watching me for the past few years. I thank him for letting me make all the mistakes that I could. It was a pleasure shooting ideas with Dr. Vincenzo Giordano (His ideas were many, mine none.)

And to all my friends out there who kept asking me about the state of this thesis, it's been great fun. The Boston trips were many and this is the end, beautiful friend, THE END.

This work was supported in part by the Automation & Robotics Research Institute, the Army Research Office grants W91NF-05-1-0314, and M-47928-CI-RIP-05075-1, and the National Science Foundation grants IIS-0326595, and CNS-0421282.

November 3, 2005

ABSTRACT

ADAPTIVE SAMPLING WITH MOBILE WSN

Publication No. _____

Koushil Sreenath, M.S.

The University of Texas at Arlington, 2005

Supervising Professor: Dan Popa

The spatiotemporally varying network topology of mobile sensor networks makes it very suitable for applications such as reconstruction of environmental fields through sampling at locations that maximally reduce the largest uncertainty in the field estimate. Mobile sensor networks comprise of multiple heterogeneous resources and a deadlock-free resource scheduling in the presence of shared and routing resources becomes necessary to schedule the most efficient (cost / energy / time) resource for a task. Location information is imperative in sensor networks for most applications for localized sensing where localizing the network adaptively with no additional hardware is important.

Adaptive sampling approaches for spatially distributed static linear and Gaussian fields with mobile robotic sensors are formulated and experimentally validated. Resource scheduling algorithms for dispatching resources in a deadlock-free manner in systems with shared and routing resources are mathematically formulated and experimentally validated. Simultaneous and Adaptive localization algorithms for sensor network localization through simple geometric constraints are validated through simulations.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Spatiotemporally varying network topologies of mobile sensor networks make them suitable for applications such as reconstruction of environmental fields through adaptively sampling at locations that maximally reduce the largest uncertainty in the field estimate. Mobile sensor networks comprise of multiple heterogeneous resources and a deadlock-free resource scheduling in the presence of shared and routing resources becomes necessary to schedule the most efficient (cost / energy / time) resource for a task. Location information is imperative in sensor networks for most applications for localized sensing where localizing the network adaptively with no additional hardware is important.

This chapter introduces the research areas of the thesis, namely, Adaptive sampling, resource scheduling and sensor network localization.

1.1 Adaptive Sampling

Monitoring environmental parameters is a complex task of great importance in many areas, such as natural living environments, homeland security, industrial or laboratory hazardous environments (biologically, radioactively, or chemically contaminated), polluted/toxic natural environments, water treatment plants, nuclear stations, war zones, remote environments, such as deep space or underwater [1].

The capabilities and distributed nature of wireless sensor networks provide an attractive sampling approach for estimation of spatiotemporally distributed environmental phenomena. Sampling is a broad methodology for gathering statistical information about a phenomenon. Using densely deployed static sensor network to cover large sampling volumes is very expensive in time and resource costs and places heavy demands on energy consumption. Physical adaptation of a sensor network, either by adaptive sensor scheduling or through robotic mobility may be the only practical approach. This leads to adaptive sampling wherein sampling strategies temporally evolve with past measurements. Information-based approaches to processing and organizing spatially distributed, multimodal sensor data in a sensor network are discussed in [2, 3].

Field estimation using the Kullback-Leibler distance as a measure of the approximation error is shown in [4], where sample density is adaptively varied over the search space depending on the state uncertainty. Adaptive Sensing [5] presents an energy-efficient topology configuration method for environment monitoring using densely deployed wireless sensor networks where redundant nodes are transitioned into passive mode as auxiliary nodes for later use. Backcasting [6], detects correlations in an environmental field during the initial preview sampling stage and this information is used for refined sampling where only a small subset of sensors are adaptively activated, thereby reducing the demands placed on energy consumption.

Environmental phenomena may appear as single or multiple events and may migrate within the environment. Hence for accurate determination of space- and time-

varying variables, we require the sensing to be spatiotemporally distributed. Robotics technology provides the possibility of mobile sensing nodes in a distributed sensor network using prior research in localization of mobile robots [7-9], localization of sensor networks [10, 11], and cooperative environment mapping approaches such as SLAM [12], and CML [13]. Robotic Sensor Agents [1], presents a wide variety of intelligent, autonomous robotic platforms for monitoring the environment. Deployment algorithms for sensor networks with mobile nodes are discussed in [14-16].

Mobile sensor agents are most suited to implement adaptive sampling strategies. A bacterial motion for detecting, seeking and tracking of an environmental phenomenon is presented in [17]. NIMS [18], presents an adaptive sampling approach for monitoring of spatiotemporal variation of atmospheric climate phenomena in a forest environment by mapping environmental variables of temperature, humidity, and solar illumination. Environmental prediction [19], uses the ensemble transform Kalman filter (ET KF) for designing flight tracks along which GPS dropwindsondes are deployed from the aircraft and provide vertical profiles of pressure, temperature, humidity and wind as they drift down on a parachute. Various estimation techniques are presented in [20] for predictive modeling in oceanography and meteorology. Optimal sample selection using singular value decomposition (SVD) of the parameter variance space is introduced in [21], where linear regression of the estimators is performed for maximization of various norms of the variance matrix. Concurrent localization and estimation of a field using multiple autonomous underwater vehicles (AUV) is presented in [22]. An extended Kalman filter based sampling approach for estimation

of parameterized fields is introduced in [23] samples are chosen to minimize the state uncertainty, represented by the covariance matrix.

This thesis considers the problem of estimation of a spatially stationary color field using mobile robotic sensors equipped with a color sensing module. A color field is chosen so that the truth model is always known easily and can be used for determining the level of accuracy in our estimation algorithms. Most environmental fields can be modeled and projected onto a two-dimensional topographical color map which can then be used for estimation. Extensive simulations and experimental results are presented.

1.2 Resource Scheduling

Mobile wireless sensor networks comprise of multiple heterogeneous resources capable of performing diverse tasks such as measuring, manipulating, moving, sensing, etc. In mobile sensor networks, a strong one-to-many mapping between a resource and the tasks that the resource can perform occurs. This mapping can be statically assigned resulting in shared resources, or dynamically assigned resulting in both shared and routing resources. Shared resources arise when multiple tasks contend for a single shared resource, while routing resources arise when multiple resources contend to perform a single task. The use of shared or routing resources is a major problem occurring in discrete event (DE) systems, including manufacturing systems, computer systems, communication systems, highway/vehicle systems, and others [24]. Failure to suitably assign, dispatch, or schedule resources in the presence of shared or routing resources, can cause serious deleterious effects on system performance, resulting in

extreme cases in system deadlock. The need then arises for deadlock prevention, deadlock avoidance, or deadlock detection and recovery.

Deadlock avoidance algorithms have been used in various scenarios such as robotic cells [25, 26], e-commerce driven manufacturing systems [27], process control such as semiconductor fabrication [28], communication network routing [29], computer operating systems, etc. The implementation of deadlock avoidance policies in autonomous distributed robotic systems such as mobile sensor networks has not been still thoroughly investigated. Preliminary simulations of efficient deadlock avoidance policies for shared resources in heterogeneous mobile sensor networks are presented in [30].

A large amount of research has been done in developing various deadlock avoidance algorithms using varied concepts such as circular wait, circular blocking, siphons in Petri nets, critical subsystems, etc. Petri net based deadlock prevention polices [31, 32] deal with detecting siphons and statically introducing control places into the net to eliminate unmarked siphons signifying deadlock. In [27], potential deadlock patterns are acquired from an off-line simulation of the part processing sequence and then, an on-line matching/reordering process is made use of to keep the current system state dissimilar to the acquired deadlock patterns. Mathematical formulations of deadlocks and traps by calculation of s-invariants of marked graphs using linear algebra are thoroughly discussed in [33]. Supervisory control of Petri nets [34] introduces an approach of keeping a Petri net from starvation by using on-line routing functions instead of traditional off-line control places, where the routing

function assigns a non-shared resource to perform the task from within a pool of resources. Detailed mathematical analysis of deadlocks and an efficient dispatching policy for deadlock avoidance based on the generalized kanban scheme using a matrix model for discrete event systems is presented in [24, 25, 26, 35-37].

Due to the heterogeneous nature of mobile sensor networks, resources are capable of performing multiple jobs. These are systems with flexible routing where tasks can choose from a set of resources. In such systems with flexible routing, route enumeration can be of exponential complexity and execution of deadlock avoidance constraints are rendered computationally intractable [38]. In [38, 39], a control model is developed that allows for small, quickly enumerable subset of less-dense routes which allows for several processing alternatives (routes) at each step while still maintaining deadlock free operation. In [40], several novel mathematical formulations are constructed for detecting active circular waits leading to a deadlock in flexible routing systems; however no deadlock avoidance algorithm is arrived at.

In this thesis, we extend the preliminary analysis of deadlock avoidance polices for shared resources in heterogeneous mobile sensor networks to more complicated scenarios. We show through experimental implementation on an actual mobile sensor network test-bed, the feasibility and effectiveness of the proposed deadlock-free supervisory control in performing complex and simultaneous sequencing of interconnected tasks. Further, a general deadlock avoidance policy for systems with flexible routing, where both shared and routing resources are present, is mathematically formulated and various simulations performed to validate deadlock-free operation.

1.3 Simultaneous and Adaptive Localization of a WSN

Location information is imperative for applications in both wireless sensor networks and mobile robotics. Many sensor network applications, such as tracking targets, environmental monitoring, geo-spatial packet routing, require that the sensor nodes know their locations. The large scale of deployment in sensor networks makes careful placement or uniform distribution of sensor nodes impractical. The requirement of the sensors to be small, un-tethered, low energy consuming, cheap, etc., make the sensors resource-constrained [41]. Localization is a challenging problem and yet crucial for many applications.

Approaches to the problem of localization are varied. A detailed introduction to localization in sensor networks is presented in [11]. GPS [42] solves the problem trivially, but equipping the sensors with the required hardware may be impractical. A small section of active beacons can be placed in the sensor network and other sensors can derive their location from these anchor nodes [43, 44]. Cooperative localization methods have been developed for relative localization [10, 45]. Other approaches involve RSSI [46], TOA [47, 48], AOA [49], and Signal Pattern Matching [42].

For localization with no additional hardware on the sensor node, the geometric constraints of radio connectivity are exploited. Some authors suggest using a mobile robot (whose position is known) to localize the sensors. However, the position of the mobile robot may be hard to determine. LaSLAT [50] uses a Bayesian filter to simultaneously localize the sensor network and track the mobile robot. In [51], a particle filter is employed to localize elements of the network based on observation of

other elements of the network. In [52], a mobile robotic sensor localized the network based on simple intersections of bounding boxes. In [53], geometric constraints based on both radio connectivity and sensing of a moving beacon localize the sensor network. The Kalman filter has been used in dead-reckoning for mobile robots but its full potential in localization of WSN has not heretofore been fully explored. In [54], an extended Kalman filter is used for localization and tracking of a target. The Kalman filter was used in [55] for active beacon and mobile AUV localization and in [56] for scheduling of sensors for target tracking. SLAM [12] and CML [13] employ Kalman filters for concurrent mapping and mobile robot localization, which can be considered similar to our work wherein the geometric constraints introduced due to radio connectivity of the static sensors play the role of features. In this paper we use the full capabilities of the Kalman filter in the general WSN localization problem.

The work in this thesis exploits geometric constrains based on radio connectivity such that range information is not needed. A mobile robot initially sweeps the network, and broadcasts from the robot are used to localize the sensor nodes. Computationally inexpensive Kalman filters implemented on the sensors fuse the information. On the other hand, as time passes, the mobile robot gradually loses its own localization information. We present an algorithm that uses updates from the better localized sensors along with GPS updates, when they occur, to correct this problem. A continuous-discrete extended Kalman filter running on the robot estimates the robot state continuously and fuses the discrete measurement updates.

Finally, an adaptive localization algorithm, based on adaptive sampling techniques [22, 23], is presented that navigates the mobile robot to an area of nodes with highest position uncertainty. This ensures that the robot maneuvers to an area where the nodes are least localized, so that it can maximize the usefulness of its positional information in best localizing the overall network. The adaptive localization strategy ensures that, with a minimal robot movement, the largest reduction in aggregated node uncertainty is achieved at every iteration of the adaptive localization algorithm.

1.4 Summary

This chapter introduced the scenarios of adaptive sampling using mobile wireless sensor networks, resource scheduling and deadlock avoidance policies in the presence of shared resources, and routing paths, and simultaneous adaptive localization of wireless sensor networks using geometric constraints of radio connectivity.

1.5 Contributions

This thesis makes the following contributions:

- Closed form information measures in linear regression are used to adaptively estimate spatially distributed static linear and Gaussian fields with linear parameters. Nonlinear optimal estimation techniques, such as the Kalman filter, constrained, and unconstrained nonlinear optimizers are used to adaptively estimate field and field basis parameters. An experimental robotic sensor is designed, developed and used to adaptively estimate a linear color field.

- Deadlock avoidance techniques developed using the discrete event controller is extended and implemented on a mobile wireless sensor network comprising of Cybermotion SR2 patrol robots and Berkley motes, such that smooth, deadlock-free resource scheduling occurs in the presence of shared resources. Further, a general mathematical formulation is developed for deadlock avoidance in systems with flexible-routing, where both shared and routing resources exist. Simulations are done to validate deadlock-free operation.
- A simultaneous localization algorithm is developed and simulated for localization of a sensor network using geometric constraints of radio connectivity. An adaptive localization algorithm is developed to adaptively navigate a mobile robot such that it optimally minimizes the largest localization uncertainty of a sensor network.

1.6 Thesis organization

This thesis presents algorithms for adaptive sampling, resource scheduling and localization using mobile sensor networks. The remainder of it is structured as follows.

Chapter 2 presents an adaptive sampling strategy for field estimation using an extended Kalman filter. Extensive simulation results, experimental results and development of the mobile robotic sensors are discussed.

Chapter 3 presents the simulation and experimental implementation of the deadlock avoidance policy using the discrete-event controller for resource scheduling in the presence of shared resources. Further mathematical formulations are discussed and

a simulation of a deadlock avoidance policy in the general case of routing paths and shared resources is presented.

Chapter 4 provides localization algorithms for the simultaneous and adaptive localization of a wireless sensor network using geometrical constraints of radio connectivity.

Chapter 5 details the robotic platform that was designed and built to experimentally validate the adaptive sampling algorithms.

Chapter 6 summarizes the main contributions of this thesis and provides suggestions for future research.

CHAPTER 2

ADAPTIVE SAMPLING

The capabilities and distributed nature of wireless sensor networks provide an attractive sampling approach for estimation of spatiotemporally distributed environmental phenomena. Adaptive sampling is the scenario where sampling strategies temporally evolve with past measurements for optimality. In the context of mobile sensor networks, the problem of adaptive sampling by selection and repositioning of mobile sensing nodes in order to optimally estimate the parameters of distributed variable field models is considered.

This chapter considers the problem of estimation of a spatially stationary field spread over a region R using mobile robotic sensors. The estimation of the field by the sampling algorithm reduces the region R to a set G of sampling locations. The optimal construction of G is constrained by several factors such as the non-holonomic constraints on vehicle kinematics, the communication connectivity due to mobility of the sensor network, the inherent inaccuracy in positional estimates due to navigational errors of mobile nodes, and the spatial granularity of the field that arises due to the sensors used.

In this chapter, extensive simulations of field estimations using adaptive sampling techniques by simple linear regression, constrained nonlinear optimization and

optimal estimation methods are discussed. Experimental results of 2D deployment scenarios using custom-built, low-cost mobile sensor robots are presented.

This chapter is organized into the following sections. Section 2.1 discusses various field models and mathematical formulations of different adaptive sampling algorithms. Section 2.2 presents several simulations of estimation of spatially distributed static fields. Section 2.3 illustrates the experimental setup and presents experimental results validating the proposed adaptive sampling algorithm.

2.1 Models

Mathematical models are formulated to represent various parameterized fields. These are used for simulating various adaptive sampling algorithms for field estimation. This section presents mathematical models of various parameterized fields and mathematically proposes several adaptive sampling algorithms.

2.1.1 Estimation of a parameterized field using linear regression

Regression is a statistical method of estimating the conditional expected value of a dependent variable given the values of the other independent variables. When the relation between the dependent variable to the independent variable is assumed to be a linear function of some parameters, we have linear regression.

Linear regression has been used for estimating linear fields in [22], and for nonlinear Gaussian fields but linear in the parameters in [23].

The independent variables are the position of the sampling location given as

$$X = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.1}$$

with a general field linear in the parameters represented by

$$F = a_0 + a_1 g_1(X) + \ldots + a_m g_m(X) \tag{2.2}$$

where the parameters $a_0,\ a_1,\ \cdots,\ a_m$ are all linear and the basis function of the field, $g_1(X),\ g_2(X),\ \cdots,\ g_m(X)$ may be nonlinear such as the Gaussian basis given by

$$g(X) = g(x, y) = e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \tag{2.3}$$

The assumption that the field distribution is linear in its parameters allows us to compute a closed form solution for the information measure used by the sampling algorithm. After n measurements at locations $X_1, X_2, \cdots, X_n$, the field measures depend linearly on the coefficients $a_0, a_1, \cdots, a_m$ via position-dependent functions, and we can directly estimate the unknown coefficients from the least-square solution

$$\begin{aligned} Z_1 &= a_0 + a_1 g_1(X_1) + \ldots + a_m g_m(X_1) \\ Z_2 &= a_0 + a_1 g_1(X_2) + \ldots + a_m g_m(X_2) \\ &\vdots \\ Z_n &= a_0 + a_1 g_1(X_n) + \ldots + a_m g_m(X_n) \end{aligned} \tag{2.4}$$

$$\hat{A}_n = \begin{pmatrix} 1 & g_i(X_j) \end{pmatrix}^{+}_{i \le m,\, j \le n} \begin{pmatrix} Z_1 \\ \vdots \\ Z_n \end{pmatrix} = M_n^{+} \begin{pmatrix} Z_1 \\ \vdots \\ Z_n \end{pmatrix} \tag{2.5}$$

Since the pseudo-inverse has a closed form, given by

$$M_n^{+} = \left(M_n^T M_n\right)^{-1} M_n^T \tag{2.6}$$

we obtain

$$\hat{A}_n = \left(\sum_{j=1}^{n} \begin{pmatrix} 1 & \cdots & g_i(X_j) & \cdots & g_m(X_j) \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ g_i(X_j) \\ \vdots \\ g_m(X_j) \end{pmatrix} \right)^{-1} \sum_{j=1}^{n} Z_j \begin{pmatrix} 1 \\ \vdots \\ g_i(X_j) \\ \vdots \\ g_m(X_j) \end{pmatrix} \tag{2.7}$$

The covariance matrix of $\hat{A}_n$ can be related directly to the constant measurement uncertainty as

$$\mathrm{var}\left(\hat{A}_n\right) = \mathrm{var}(Z_i)\left(M_n^T M_n\right)^{-1} \tag{2.8}$$

and the adaptive sampling algorithm will move the vehicle from location X_n to X_{n+1}, such that the following $p-norm$ is minimized over the search space Θ.

$$m(X) = \left\| \begin{pmatrix} & 1 & \\ M_n^T & \vdots & \\ & g_i(X) & \end{pmatrix} \begin{pmatrix} & & M_n & & \\ 1 & \cdots & g_i(X) & \cdots & g_m(X) \end{pmatrix} \right\|_p \tag{2.9}$$

$$m(X_{n+1}) \le m(X), \forall X \in \Theta \tag{2.10}$$

2.1.2 Estimation of a parameterized field using a Kalman Filter

Assuming the field distribution to be linear in its parameters allows us to compute a closed form solution for the information measure used in the sampling algorithm to decide the next sampling location. But such assumptions are not practical for all scenarios. Here we consider estimation of a parameterized field using a Kalman filter such that the need for a closed form information measure is eliminated.

The sampling location is given as

$$X = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.11}$$

with a parameterized field model with linear parameters

$$F = a_0 + a_1 g_1(X) + a_2 g_2(X) \tag{2.12}$$

and nonlinear Gaussian basis

$$g(X) = g(x, y) = e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \tag{2.13}$$

The states to be estimated are the field coefficients

$$A = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}^T \tag{2.14}$$

which get updated in time by the time-update equation of the Kalman filter

$$\begin{aligned} P_{k+1}^- &= P_k + G_k Q_k G_k^T \\ \hat{A}_{k+1}^- &= \hat{A}_k \end{aligned} \tag{2.15}$$

On sampling at a location, the measurement-update equation of the Kalman filter is employed to improve the estimate by combining the information available in the new measurement. The measurement-update equation is

$$\begin{aligned} P_{k+1} &= \left[\left(P_{k+1}^- \right)^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1} \right]^{-1} \\ K_{k+1} &= P_{k+1} H_{k+1}^T R_{k+1}^{-1} \\ \hat{A}_{k+1} &= \hat{A}_{k+1}^- + K_{k+1} \left(z_{k+1} - H_{k+1} \hat{A}_{k+1}^- \right) \end{aligned} \tag{2.16}$$

where the observation matrix is given as

$$H_{k+1} = \begin{bmatrix} 1 & g_1(X_{k+1}) & g_2(X_{k+1}) \end{bmatrix} \tag{2.17}$$

The adaptive sampling algorithm will move the vehicle from location X_k to X_{k+1}, such that the following $2-norm$ of the covariance matrix is minimized over the search space Θ.

$$\begin{aligned} m(X_k) &= \| P_k \|_2 \\ m(X_{k+1}) &\le m(X_k), \forall X \in \Theta \end{aligned} \tag{2.18}$$

2.1.3 Estimation of field parameters using linear regression and nonlinear optimization

Earlier sections have considered fields with Gaussian basis with known mean and variances. Here we approach the problem of estimating a parameterized field with

linear parameters and also estimating the means of the basis that make up the field. This is a more accurate approximation of a true environmental field.

We attempt to arrive at a solution by using linear-regression for estimation of the linear field parameters and using nonlinear optimization for estimation of the basis parameters of the field.

The sampling location, the field model and the field basis are given by

$$X = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.19}$$

$$F = a_0 + a_1 g_1(X) + a_2 g_2(X) \tag{2.20}$$

$$g(X) = g(x, y) = e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \tag{2.21}$$

After n measurements at locations $X_1, X_2, \cdots, X_n$, the field measures depend linearly on the coefficients $a_0, a_1, \cdots, a_m$ via nonlinear position-dependent basis functions. We try estimating the unknown coefficients from the least-square solution

$$\begin{aligned} Z_1 &= a_0 + a_1 g_1(X_1) + \ldots + a_m g_m(X_1) \\ Z_2 &= a_0 + a_1 g_1(X_2) + \ldots + a_m g_m(X_2) \\ &\vdots \\ Z_n &= a_0 + a_1 g_1(X_n) + \ldots + a_m g_m(X_n) \end{aligned} \tag{2.22}$$

and estimating the unknown means of the basis by finding the local minima of the minimization function using nonlinear unconstrained optimization techniques. The minimization function is given as

$$\sum_{\forall i} \left(\hat{a}_0 + \hat{a}_1 g_1\left(\hat{X}_i\right) + \hat{a}_2 g_2\left(\hat{X}_i\right) - Z_i \right)^2 \tag{2.23}$$

2.1.4 Estimation of field parameters using nonlinear optimization

Using linear regression for estimating field parameters and nonlinear optimization for field basis parameters does not incorporate complete knowledge of the history of the estimates. We try to use a nonlinear constrained optimization technique for estimating the field parameters and the basis means together.

The sampling location, the field model and the field basis as earlier are given as

$$X = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.24}$$

$$F = a_0 + a_1 g_1(X) + a_2 g_2(X) \tag{2.25}$$

$$g(X) = g(x, y) = e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \tag{2.26}$$

We use a nonlinear constrained optimizer which minimizes a function

$$\min_x f(x) \tag{2.27}$$

subject to the linear equality and inequality constraints,

$$\begin{aligned} A_{eq} x &= b_{eq} \\ A.x &\le b \end{aligned} \tag{2.28}$$

the nonlinear equality and inequality constraints,

$$\begin{aligned} C_{eq}(x) &= 0 \\ C(x) &\le 0 \end{aligned} \tag{2.29}$$

and bounded by

$$lb \le x \le ub \tag{2.30}$$

To minimize the overall error in our estimation of $a_0, a_1, a_2, x_{c1}, y_{c1}, x_{c2}, y_{c2}$, we use a minimization function

$$\sum_{\forall i}\left(\hat{a}_0^i+\hat{a}_1^i e^{-\frac{\left(x-\hat{x}_{c1}^i\right)^2+\left(y-\hat{y}_{c1}^i\right)^2}{2\sigma^2}}+\hat{a}_2^i e^{-\frac{\left(x-\hat{x}_{c2}^i\right)^2+\left(y-\hat{y}_{c2}^i\right)^2}{2\sigma^2}}-Z_i\right)^2 \tag{2.31}$$

For accurate, repeatable convergence of estimating parameters, we partition our search space Θ into various sub search spaces $(\Theta_{2x2},\Theta_{4x4})$ and carry on sub estimations to arrive at an estimate for a particular configuration. From among the estimates for various configurations, the best estimate is chosen. This is the scenario of Divide-n-Conquer where the subspace is partitioned and from among the partitioned results, the best one is chosen.

To estimate the entire field (both the field parameters and all basis parameters,) we consider a single basis field

$$F=a_0+a_1 g_1(X) \tag{2.32}$$

and to estimate all field and basis parameters, $a_0, a_1, x_c, y_c, \sigma$, we use a minimization function

$$\sum_{\forall i}\left(\hat{a}_0^i+\hat{a}_1^i e^{-\frac{\left(x-\hat{x}_{c1}^i\right)^2+\left(y-\hat{y}_{c1}^i\right)^2}{2\hat{\sigma}^{i^2}}}-Z_i\right)^2 \tag{2.33}$$

2.1.5 Estimation of a parameterized field with localization uncertainty

In approaches described in the previous sections, we assume that there is absolutely no uncertainty about the sampling location. However navigation of a mobile robotic sensor is subjected to various localization errors. We can use location information that is embedded in a field sample (due to the inherent background mathematical model of the field) to better localize our sampling location. A given

known localization uncertainty described by a simple kinematic model is introduced into the system [23].

Along with the field, we also estimate the sampling location. The aggregate state then contains both the positional information X_k, and the field parameter estimates A_k. The state and output equations are written as

$$\begin{bmatrix} X_{k+1} \\ A_{k+1} \end{bmatrix} = \begin{bmatrix} X_k \\ A_k \end{bmatrix} + \begin{bmatrix} I_n \\ 0 \end{bmatrix} U_k + \begin{bmatrix} w_k \\ 0 \end{bmatrix} = \begin{bmatrix} X_k \\ A_k \end{bmatrix} + BU_k + \vartheta_k \tag{2.34}$$

$$\begin{bmatrix} Y_k \\ Z_k \end{bmatrix} = \begin{bmatrix} X_k \\ \begin{bmatrix} 1 & X_k{}^T \end{bmatrix} A_k \end{bmatrix} + \begin{bmatrix} \xi_k \\ v_k \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ 0 & \begin{bmatrix} 1 & X_k{}^T \end{bmatrix} \end{bmatrix} \begin{bmatrix} X_k \\ A_k \end{bmatrix} + \lambda_k \tag{2.35}$$

where the white noise covariances of state and output are

$$\begin{aligned} E\left[\vartheta_k, \vartheta_k{}^T\right] &= Q = \begin{bmatrix} Q_1 & 0 \\ 0 & 0 \end{bmatrix} \\ E\left[\lambda_k, \lambda_k{}^T\right] &= R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \end{aligned} \tag{2.36}$$

The state evolution is governed by the nonlinear Kalman filter time-update equation

$$\begin{aligned} P_{k+1}^- &= P_k + G_k Q_k G_k^T \\ \begin{bmatrix} \hat{X}_{k+1}^- \\ \hat{A}_{k+1}^- \end{bmatrix} &= \begin{bmatrix} \hat{X}_k \\ \hat{A}_k \end{bmatrix} + BU_k \end{aligned} \tag{2.37}$$

On sampling at a location, the measurement-update equation of the Kalman filter is employed to improve the estimate by combining the information available in the new measurement. This information contains both information about the field and the position of the sample. The general nonlinear measurement-update equation is

$$
\begin{aligned}
P_{k+1} &= \left[\left(P_{k+1}^{-}\right)^{-1} + H_{k+1}^{T} R_{k+1}^{-1} H_{k+1}\right]^{-1} \\
K_{k+1} &= P_{k+1} H_{k+1}^{T} R_{k+1}^{-1} \\
\begin{bmatrix} \hat{X}_{k+1} \\ \hat{A}_{k+1} \end{bmatrix} &= \begin{bmatrix} \hat{X}_{k+1}^{-} \\ \hat{A}_{k+1}^{-} \end{bmatrix} + K_{k+1}\left(\begin{bmatrix} Y_{k+1} \\ Z_{k+1} \end{bmatrix} - h\left(\hat{X}_{k}^{-}, \hat{A}_{k}^{-}\right)\begin{bmatrix} \hat{X}_{k+1}^{-} \\ \hat{A}_{k+1}^{-} \end{bmatrix}\right)
\end{aligned}
\tag{2.38}
$$

For a linear field of the form $F = a_0 + a_1 g_1(X) + a_2 g_2(X)$ with $g_1(X) = x, g_2(X) = y$, we have

$$
\begin{aligned}
X &= \begin{bmatrix} x & y \end{bmatrix}^T, \ n = 2 \\
A &= \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}^T
\end{aligned}
\tag{2.39}
$$

$$
h\left(\hat{X}_{k}^{-}, \hat{A}_{k}^{-}\right) = H_k = \begin{bmatrix} I_2 & 0 \\ 0 & \begin{bmatrix} 1 & X_k{}^T \end{bmatrix} \end{bmatrix}
\tag{2.40}
$$

For a Gaussian field of the form $F = a_0 + a_1 g_1(X)$ with $g_1(X) = e^{-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2}}$, we have

$$
\begin{aligned}
X &= \begin{bmatrix} x & y \end{bmatrix}^T, \ n = 2 \\
A &= \begin{bmatrix} a_0 & a_1 & x_c & y_c & \sigma \end{bmatrix}^T
\end{aligned}
\tag{2.41}
$$

with the output equation modified as

$$
\begin{bmatrix} Y_k \\ Z_k \end{bmatrix} = \begin{bmatrix} X_k \\ h\left(X_k, A_k\right) \end{bmatrix} + \begin{bmatrix} \xi_k \\ v_k \end{bmatrix}
\tag{2.42}
$$

$$
h\left(X_k, A_k\right) = a_0 + a_1 e^{-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2}}
\tag{2.43}
$$

and the linearized Jacobian output matrix represented as

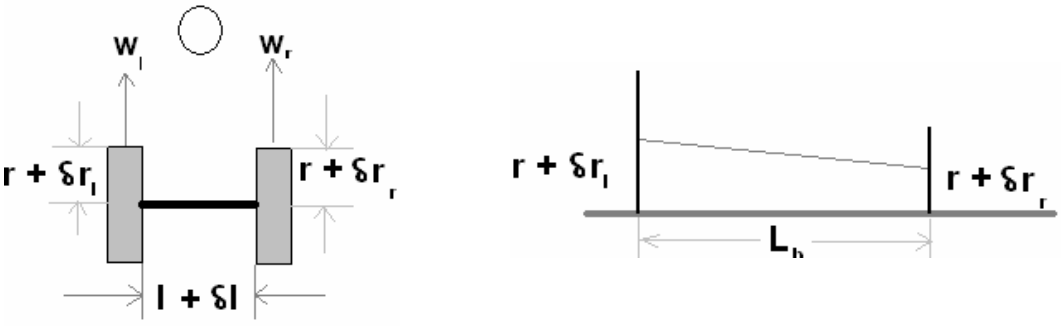
$$
H = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}
\tag{2.44}
$$

$$H_1 = I_2,\ \ H_2 = \begin{bmatrix} \partial h/\partial a_0 \\ \partial h/\partial a_1 \\ \partial h/\partial x_c \\ \partial h/\partial y_c \\ \partial h/\partial \sigma \end{bmatrix} = \begin{bmatrix} 1 \\ e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \\ a_1 \dfrac{(x-x_c)^2+(y-y_c)^2}{\sigma^3} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \\ a_1 \dfrac{x-x_c}{\sigma^2} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \\ a_1 \dfrac{y-y_1}{\sigma^2} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \end{bmatrix} \tag{2.45}$$

For information about the development of the nonlinear Kalman filter equations, interested users are referred to [57, 58]

2.1.6 Differential robot simulation

A differential robot model as illustrated in Figure 2.1 is used to mathematically represent the physical robot kinematics. A systematic error [59] is injected into the system to account for navigational errors that arise due to practical inaccuracies in construction and mechanical assembly.



(a) (b)

Figure 2.1: Differential robot with uncertainty in wheel radii and axle length. (a) Top view, and (b) Front view.

Taking the states of the system to be $X = [x_1 \quad x_2 \quad x_3]^T = [x \quad y \quad \theta]^T$, the system model is given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \frac{(\omega_R r_R + \omega_L r_L)}{2}\cos(x_3) \\ \frac{(\omega_R r_R + \omega_L r_L)}{2}\sin(x_3) \\ \frac{(\omega_R r_R - \omega_L r_L)}{L_b} \end{bmatrix} \tag{2.46}$$

where the effective axle length is

$$L_b = \sqrt{(L^2 - (r_L - r_R)^2)} \tag{2.47}$$

and the inputs being

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T = \begin{bmatrix} \omega_R & \omega_L \end{bmatrix}^T \tag{2.48}$$

The nominal system model can be obtained from the above system by replacing $r_l = r_r = r$, and $L_b = L$. In the dead reckoning scheme [60], $\begin{bmatrix} u_1 & u_2 \end{bmatrix}$ are not control inputs, but are rather the measured wheel velocities which can be measured by taking the difference between encoder counts from successive sample periods. If $\phi_R(k), \phi_L(k)$ represent the encoder counts of the right and left wheels respectively, then using forward difference approximation of the derivative in the kinematic system, equation (2.46), we arrive at the discrete system model given by

$$\begin{aligned} \hat{x}_1(k) &= \hat{x}_1(k-1) + \frac{K_{drv-R}\Delta\phi_R r_R + K_{drv-L}\Delta\phi_L r_L}{2}\cos(\hat{x}_3(k-1)) \\ \hat{x}_2(k) &= \hat{x}_2(k-1) + \frac{K_{drv-R}\Delta\phi_R r_R + K_{drv-L}\Delta\phi_L r_L}{2}\sin(\hat{x}_3(k-1)) \\ \hat{x}_3(k) &= \hat{x}_3(k-1) + \frac{K_{drv-R}\Delta\phi_R r_R - K_{drv-L}\Delta\phi_L r_L}{L_b} \end{aligned} \tag{2.49}$$

where K_{drv-R}, K_{drv-L} are the drive constants of the right and left wheel respectively in terms of distance per drive count and $\Delta\phi_R = \phi_R(k) - \phi_R(k-1), \Delta\phi_L = \phi_L(k) - \phi_L(k-1)$ are the change in drive counts measured between successive sample periods.

The encoder is modeled as a simple extrapolation model and is given by

$$\phi(k) = \phi(k-1) + K_{dc}\left(dc(k) - dc_{zero}\right)\Delta t \tag{2.50}$$

where K_{dc} is a proportionality constant in terms of encoder counts / percent duty cycle, dc_{zero} is the duty cycle percent that keeps the motor at standstill, and Δt is the time interval.

A simple quasi-holonomic control is used to navigate the robot from an initial start location to a destination location. This is achieved by first orienting the robot along the path from the start to the destination, then moving the robot along this path and then orienting the robot to match the required destination orientation. A simple Kalman filter implemented serves to fuse information from the GPS (the overhead camera) to correct navigational errors.

2.2 Field estimation simulations

The proposed adaptive sampling algorithms discussed in section 2.1 are simulated on the various mathematical field models for the purpose of estimation of parameterized fields. The simulation results for the various combinations of field models and adaptive sampling algorithms are presented in this section.

2.2.1 Estimation of a parameterized Gaussian field using linear regression

A 2D nonlinear Gaussian field (though still linear in the parameters) is considered with $m = 2$, such that

$$F = a_0 + a_1 g_1(x, y) + a_2 g_2(x, y) \tag{2.51}$$

where a Gaussian basis as described in eq. (2.3) is chosen. The centers of Gaussians for $g_1(x, y)$, and $g_2(x, y)$ are $(30,30), (65,45)$ respectively for the $2-norm$ case and

$(30,30),(35,80)$ respectively for the $\infty-norm$ case. The standard deviation of the Gaussian distribution is chosen as $\sigma=10$.

Figure 2.2 illustrates the original field, Figure 2.3 depicts the field estimated by least squares and the sampling locations using the $2-norm$, and Figure 2.4 shows convergence graphs of the field parameters.

Figure 2.5, Figure 2.6, and Figure 2.7 illustrate the same information for $\infty-norm$.

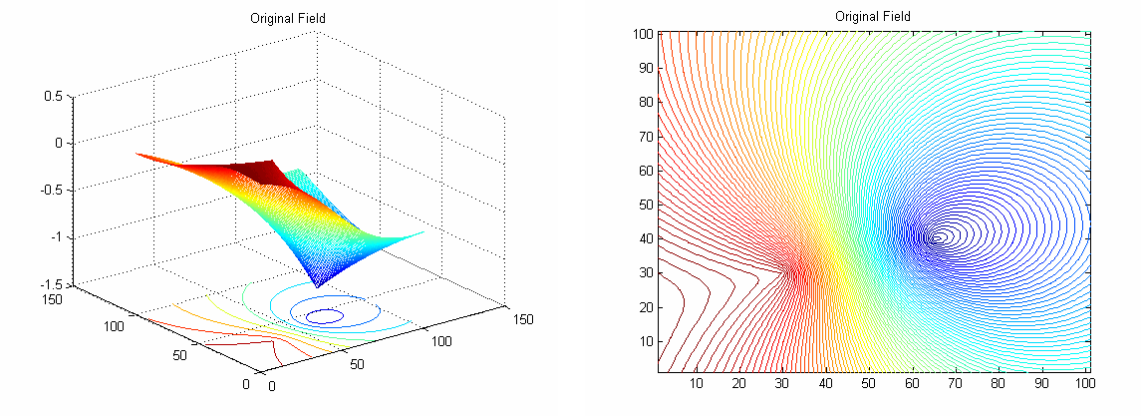


Figure 2.2: Original field (Linear regression with $2-norm$ sampling).

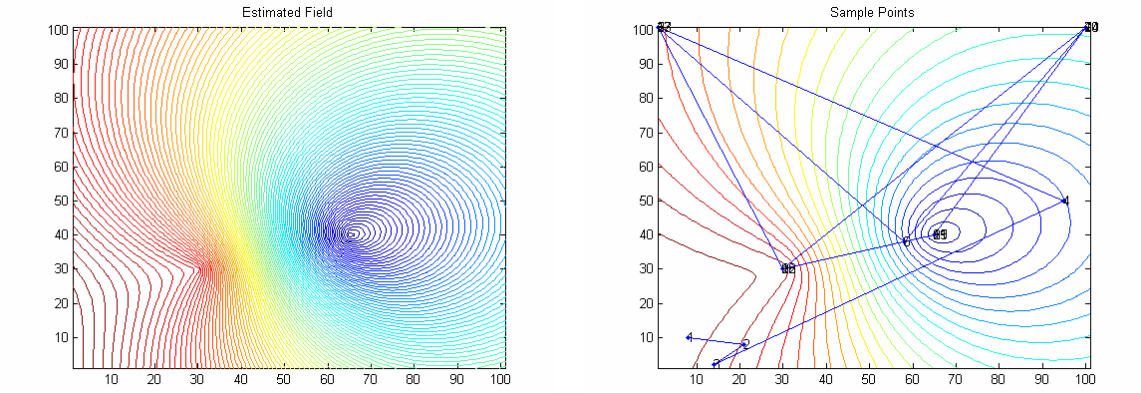


Figure 2.3: Estimated field and sampling locations ($2-norm$ sampling).

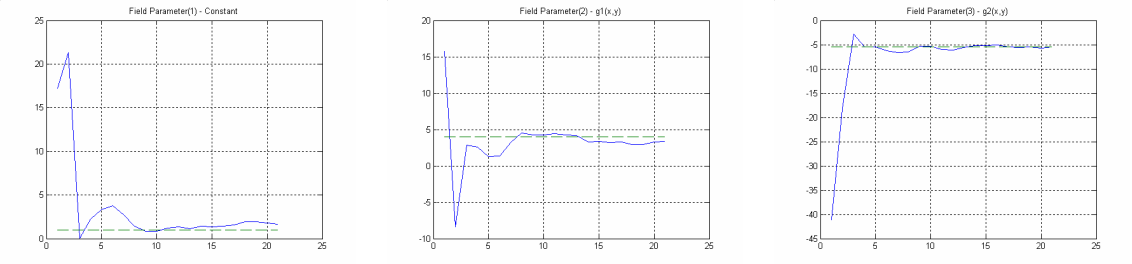


Figure 2.4: Field parameter coefficient convergence ($2-norm$ sampling).

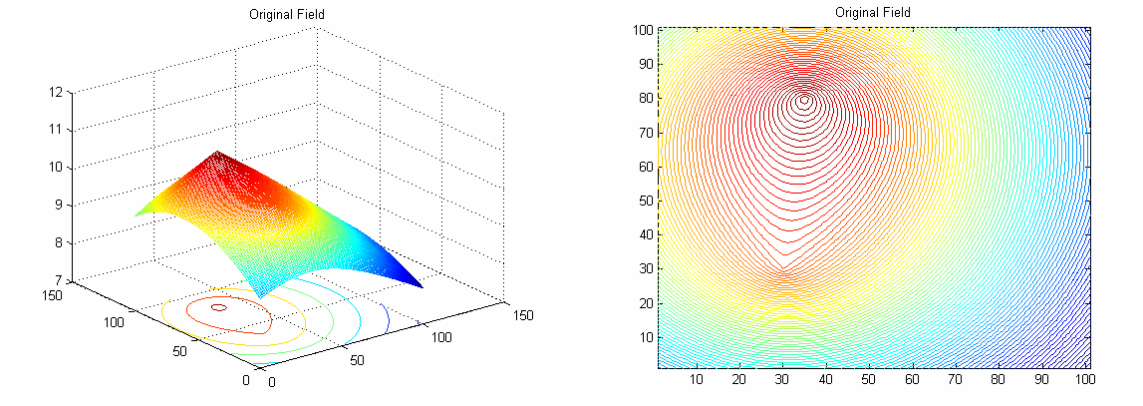


Figure 2.5: Original field (Linear regression with $\infty-norm$ sampling).

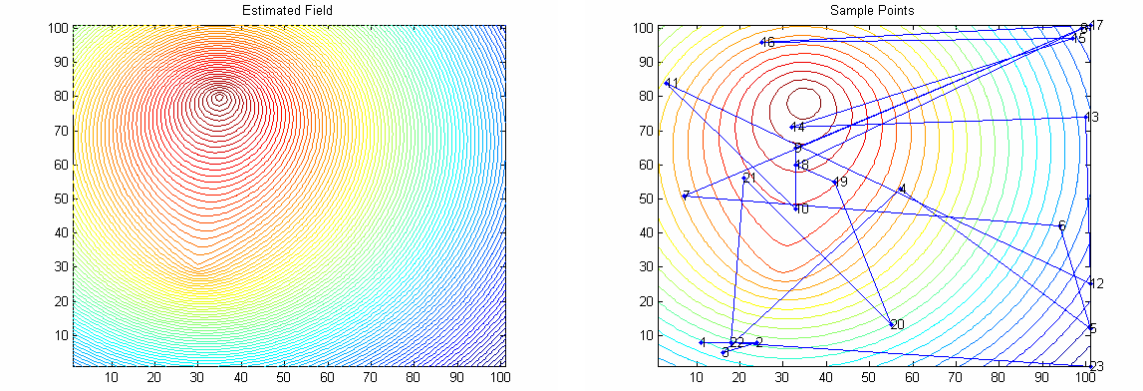


Figure 2.6: Estimated field and sampling locations ($\infty-norm$ sampling).

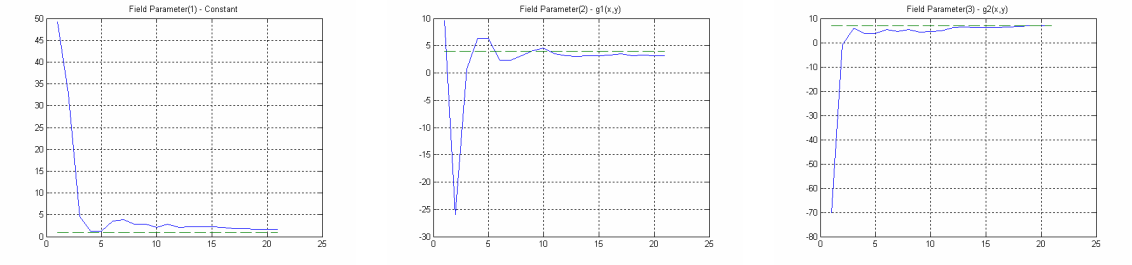


Figure 2.7: Field parameter coefficient convergence ($2-norm$ sampling).

2.2.2 Estimation of a parameterized Gaussian field using a Kalman Filter

The centers of Gaussians for $g_1(x,y)$, and $g_2(x,y)$ are $(30,30),(35,65)$ respectively and the standard deviation of the Gaussian distribution is chosen as $\sigma = 10$.

Figure 2.8 illustrates the original field, Figure 2.9 depicts the field estimated by a Kalman filter along with the sampling locations, and Figure 2.10 shows the convergence graphs of the field parameters. As is evident, even after 45 samples, the field parameters do not converge completely.

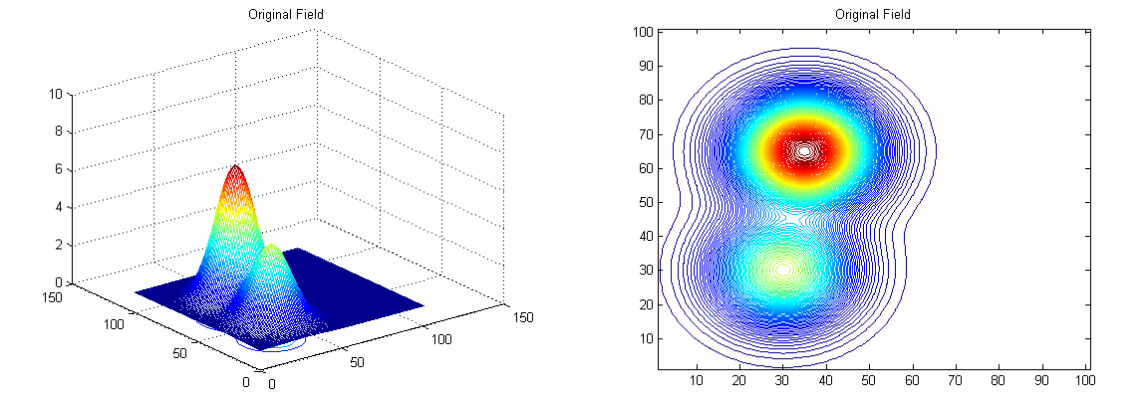


Figure 2.8: Original field (KF sampling).

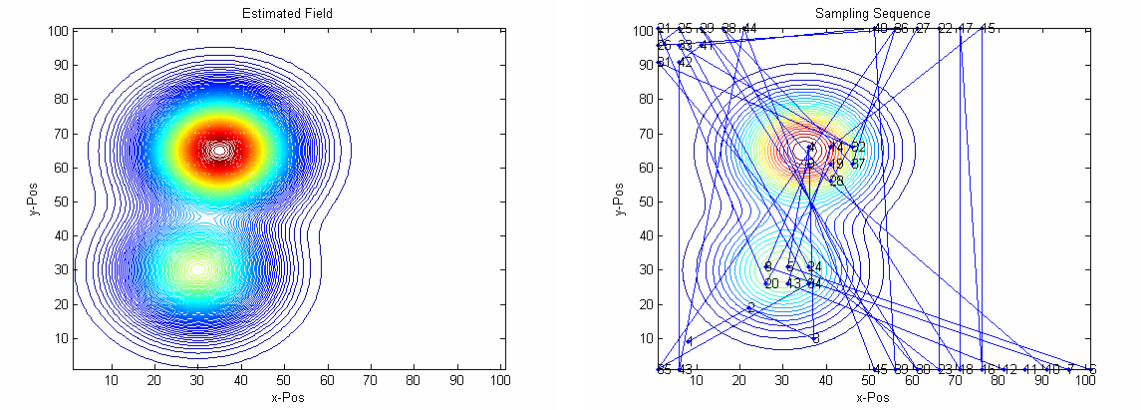


Figure 2.9: Estimated field and sampling locations (KF sampling).

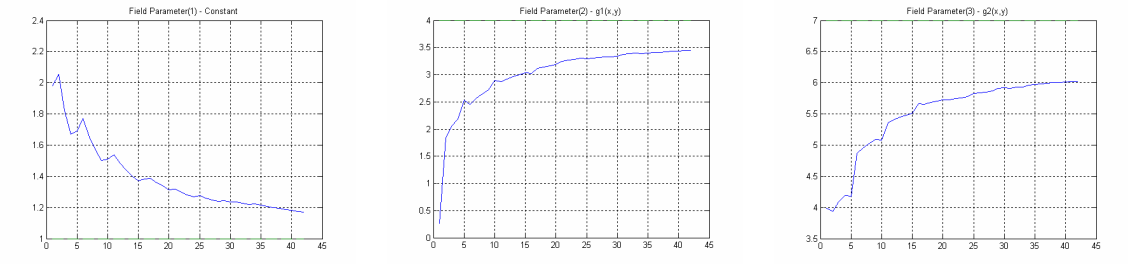


Figure 2.10: Field parameter coefficient convergence (KF sampling).

2.2.3 Estimation of field parameters using linear regression and nonlinear optimization

The centers of Gaussians for $g_1(x,y)$, and $g_2(x,y)$ are $(30,30),(35,65)$ respectively and the standard deviation of the Gaussian distribution is chosen as $\sigma = 10$.

Figure 2.11 illustrates the original field, Figure 2.12 depicts the sampling locations and the field estimated by using a combination of linear regression and nonlinear optimization techniques, Figure 2.13 shows the convergence graphs of the field parameters, and Figure 2.14 depicts the movement of the means of the Gaussian basis functions of the field. This clearly illustrates that the means diverge and we can not completely and accurately estimate both the field and the basis parameters using hybrid linear and nonlinear optimization techniques.

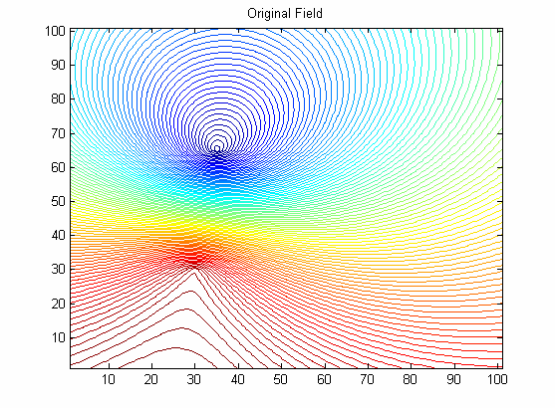


Figure 2.11: Original field (LS / NLS).

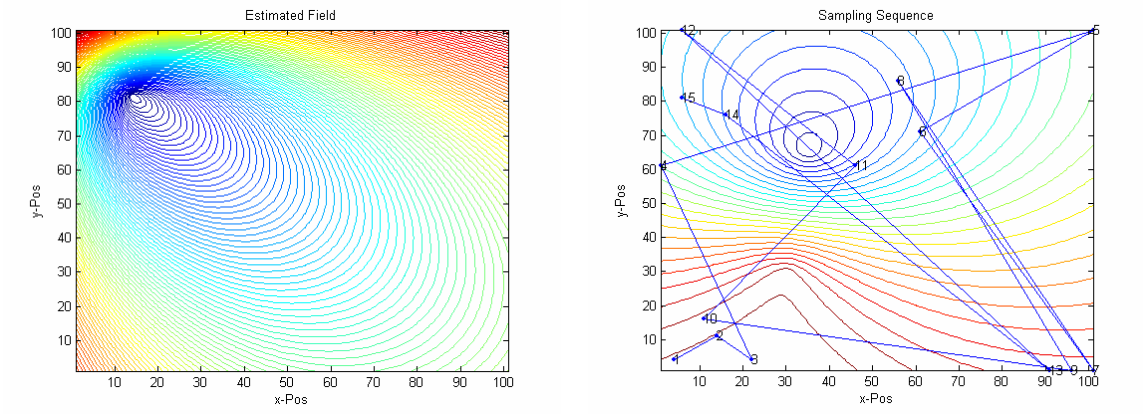


Figure 2.12: Estimated field and sampling locations (LS / NLS).

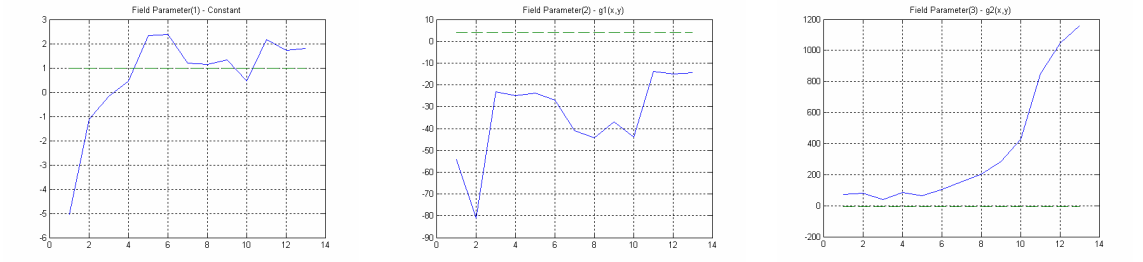


Figure 2.13: Field parameter coefficient convergence (LS / NLS).

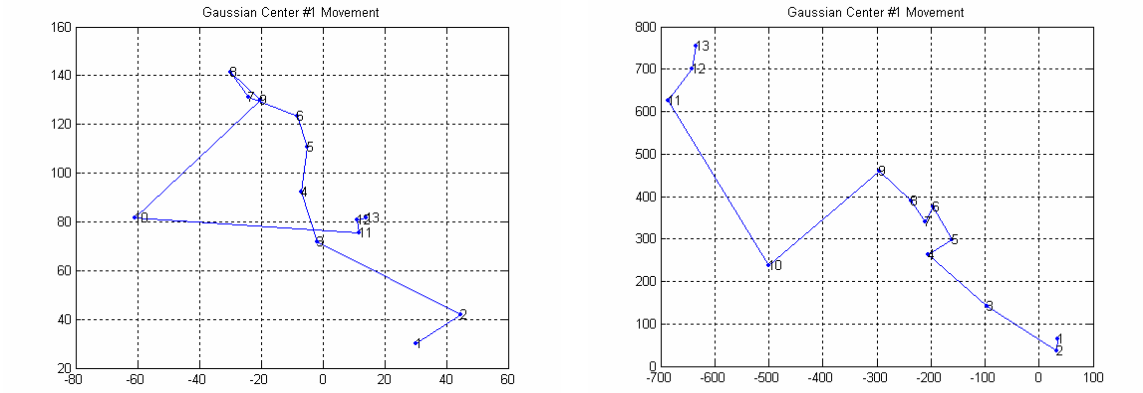


Figure 2.14: Movement of the mean of the Gaussian basis. (LS / NLS).

2.2.4 Estimation of field parameters using nonlinear optimization

A constrained nonlinear optimizer is used to estimate the field parameters a_0, a_1, a_2 and the basis parameters $x_{c1}, y_{c1}, x_{c2}, y_{c2}$ with the constraints on the bounds as

$a_0, a_1, a_2 \in [-10 \quad 10]$, $(x_{c1}, y_{c1}), (x_{c2}, y_{c2}) \in [0 \quad 100]$. The expected values are $a_0 = 1, a_1 = 4, a_2 = -5.5$, and $(x_{c1}, y_{c1}) = (30,30)$, $(x_{c2}, y_{c2}) = (35,65)$.

Figure 2.15 illustrates the original field, Figure 2.16 depicts the sampling locations and the estimated field using a constrained nonlinear optimization technique, Figure 2.17 shows the convergence graphs of the field parameters, and Figure 2.18 depicts the movement of the means of the Gaussian basis functions of the field.

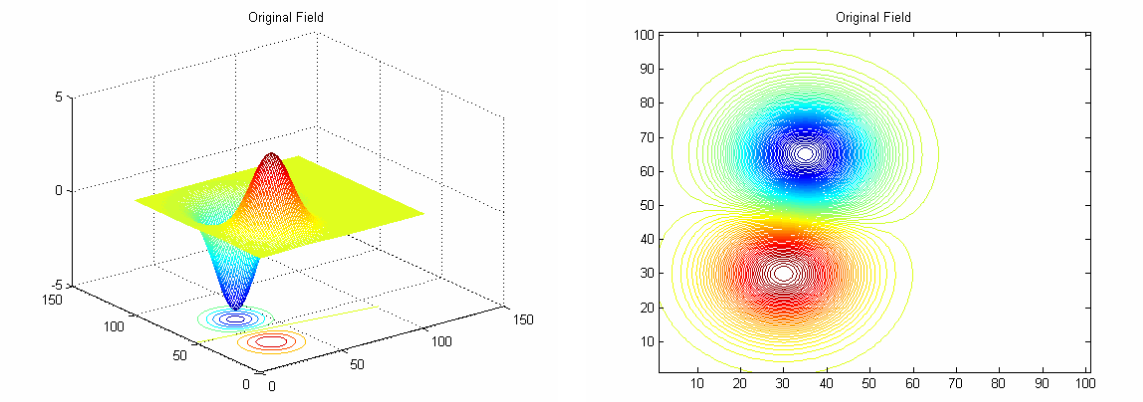


Figure 2.15: Original field (constrained nonlinear optimizer)

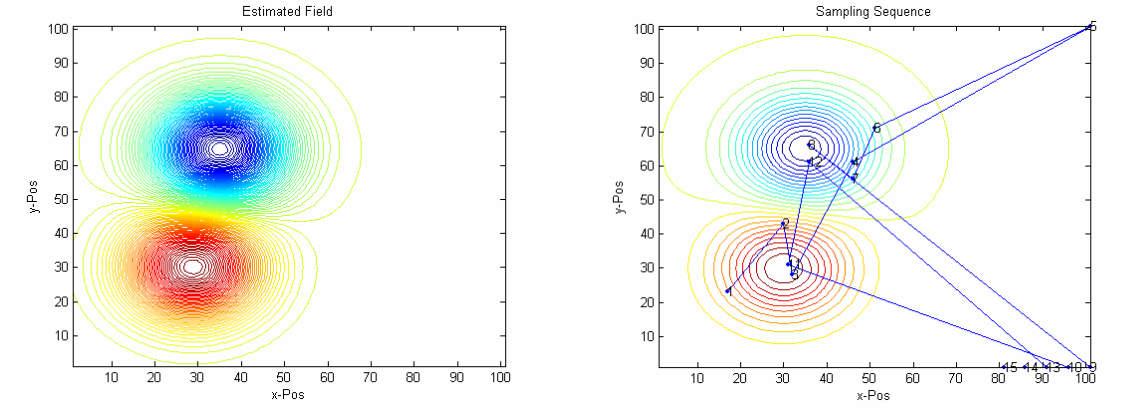


Figure 2.16: Estimated field and sampling locations (constrained nonlinear optimizer).

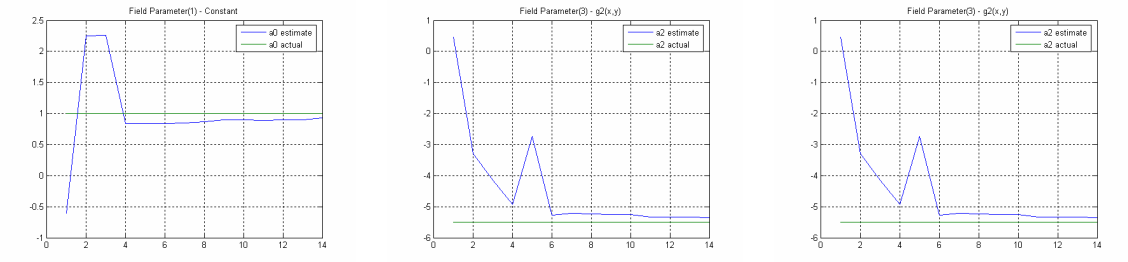


Figure 2.17: Field parameter coefficient convergence (constrained nonlinear optimizer).

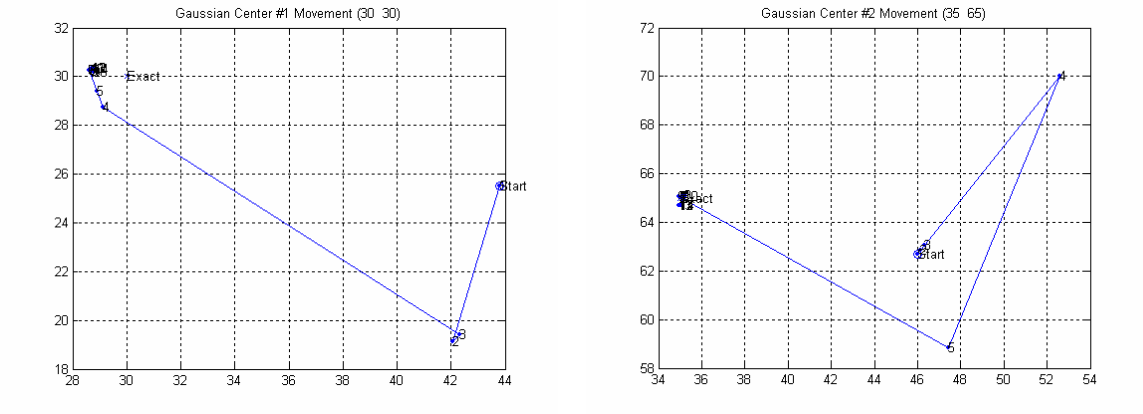


Figure 2.18: Movement of the mean of the Gaussian basis (LS / NLS).

The accuracy in the convergence of the estimation parameters in the previous case is not repeatable. A divide and conquer approach by partition of the search space into various *nxn* partitions is performed as illustrated in Figure 2.19. The constraints of bounds for the nonlinear optimizer are now updated to each sub-space rather than the entire search space.

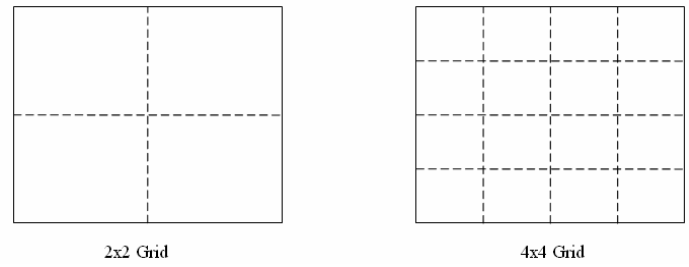


Figure 2.19: Search space partitioning.

For a $2x2$ search partitioning, Figure 2.20 depicts the sampling locations and estimated field, Figure 2.21 shows the convergence graphs of the field parameters, and Figure 2.22 depicts the movement of the basis means. As can be seen the best estimates are available in the search space combination of $(0,0),(0,1)$ where the two Gaussian basis functions are centered. For a $4x4$ search partitioning, Figure 2.23, Figure 2.24, and Figure 2.25 illustrate the same information.

The partitioning gives us faster sub-space convergence and has a much higher possibility of convergence.

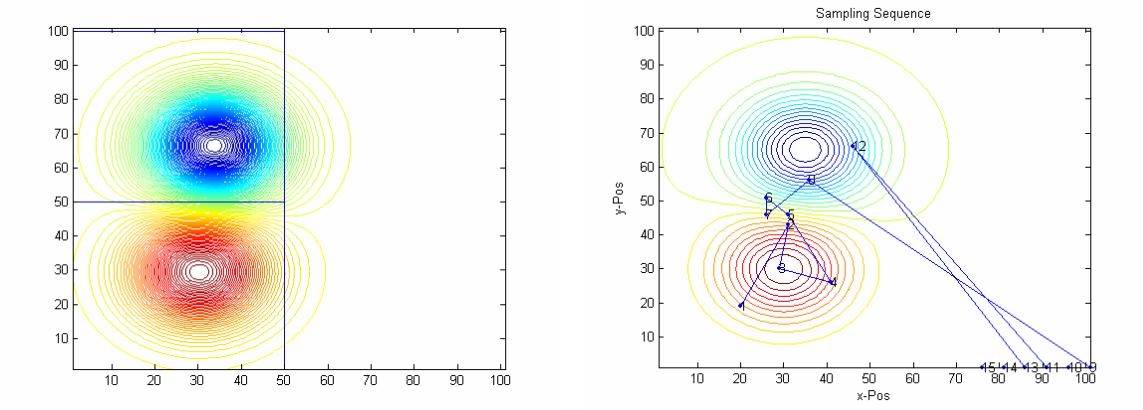


Figure 2.20: Estimated field and sampling locations ($2x2$ partitioning).

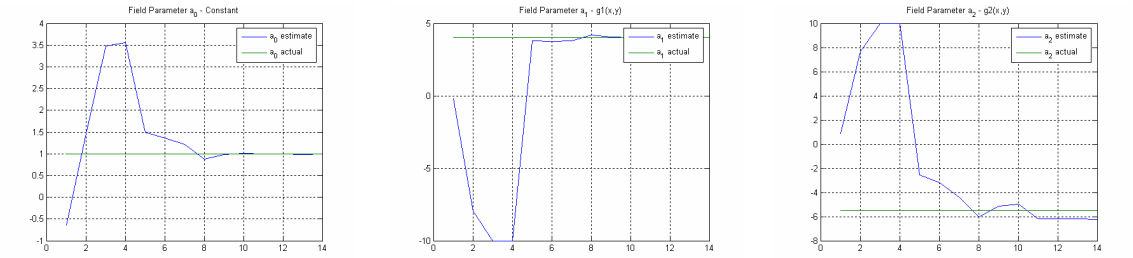


Figure 2.21: Field parameter coefficient convergence ($2x2$ partitioning).

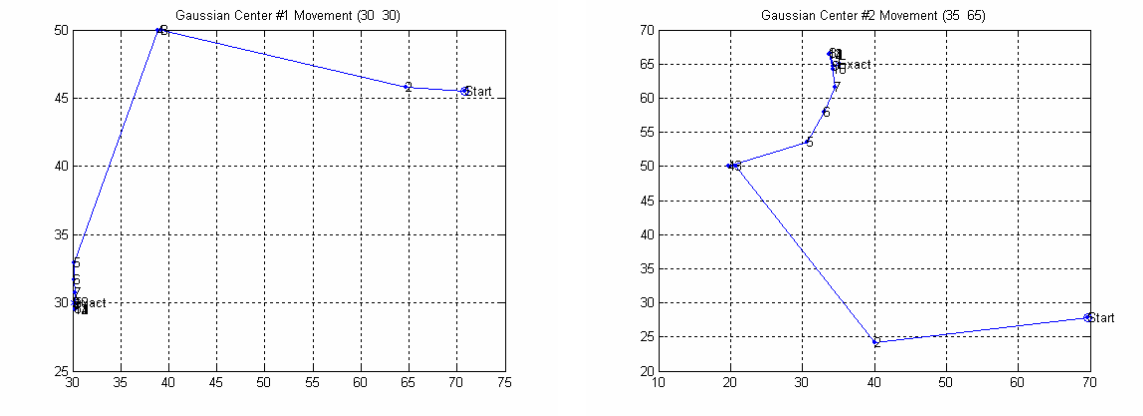


Figure 2.22: Movement of the mean of the Gaussian basis ($2x2$ partitioning).

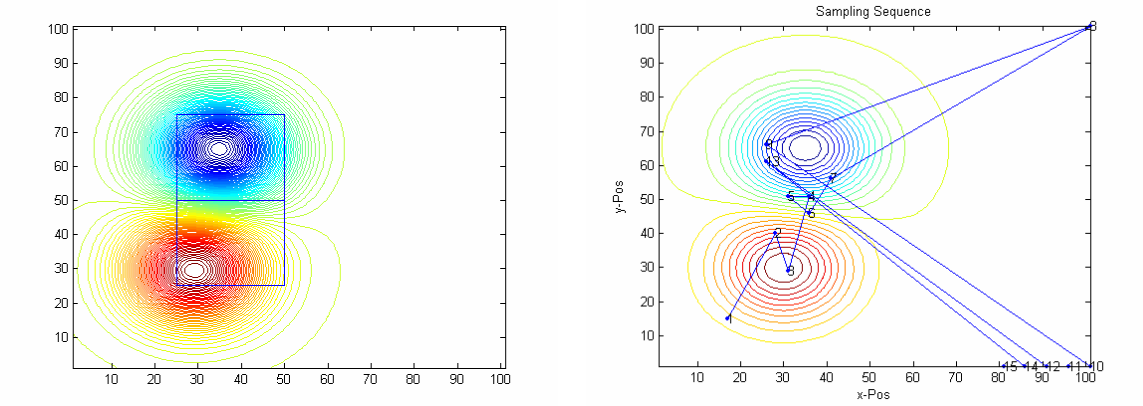


Figure 2.23: Estimated field and sampling locations ($4x4$ partitioning).

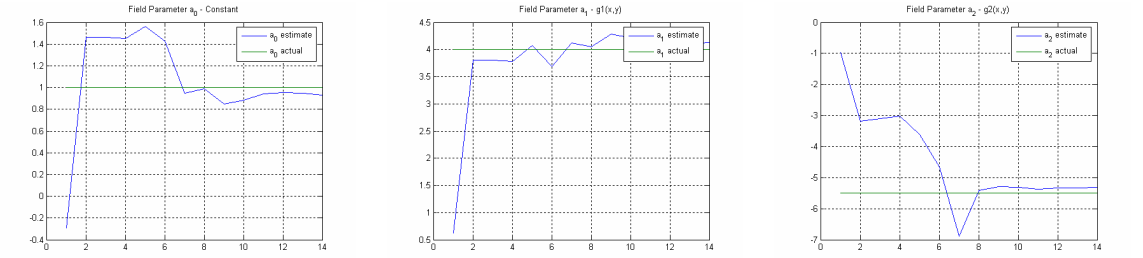


Figure 2.24: Field parameter coefficient convergence ($4x4$ partitioning).

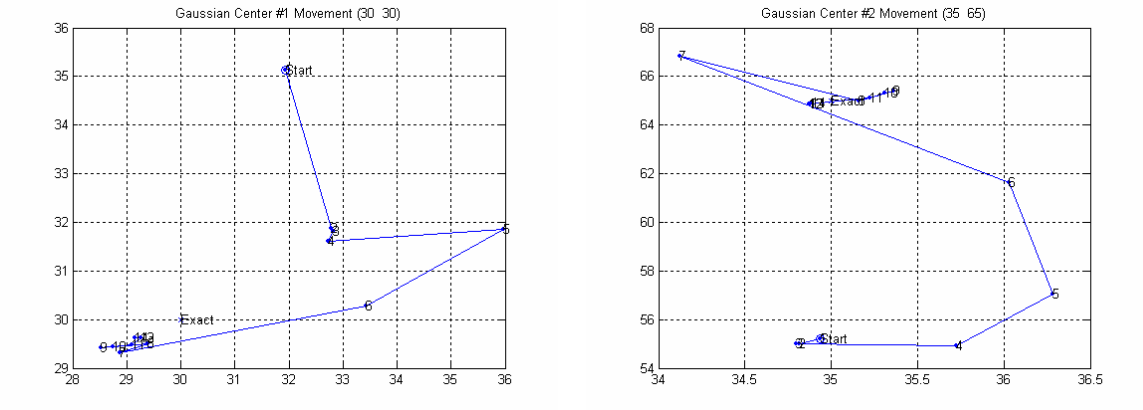


Figure 2.25: Movement of the mean of the Gaussian basis ($4x4$ partitioning).

Further, a single basis function is chosen as the field as in equation (2.32) and all the parameters of the field a_0, a_1, and basis x_{c1}, y_{c1}, σ are estimated. The same constraints as in earlier simulations are used along with the new bound of $\sigma \in [0 \quad 10]$.

Figure 2.26 illustrates the original field, Figure 2.27 depicts the estimated field and the sampling locations where the entire field is estimated, Figure 2.28 shows the convergence graphs for the field parameters, and Figure 2.29 shows the convergence graphs for the basis parameters.

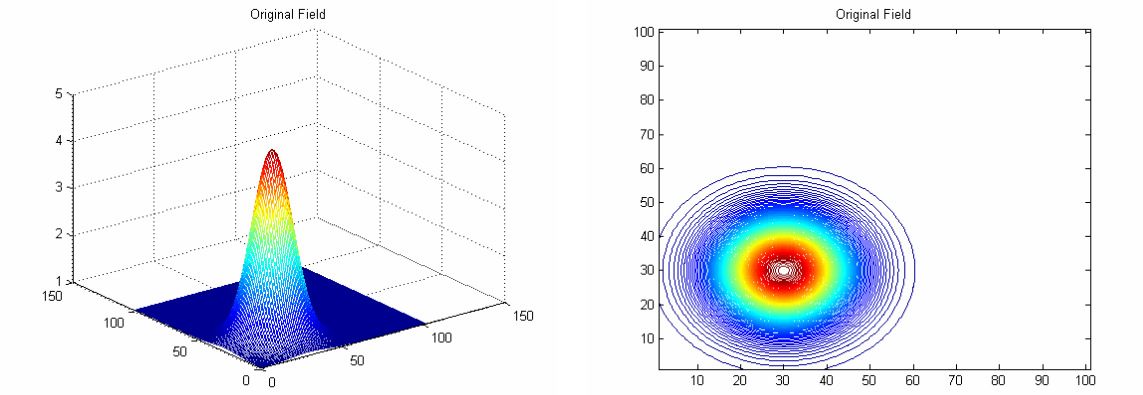


Figure 2.26: Original field (nonlinear optimization for field and basis parameter estimation).

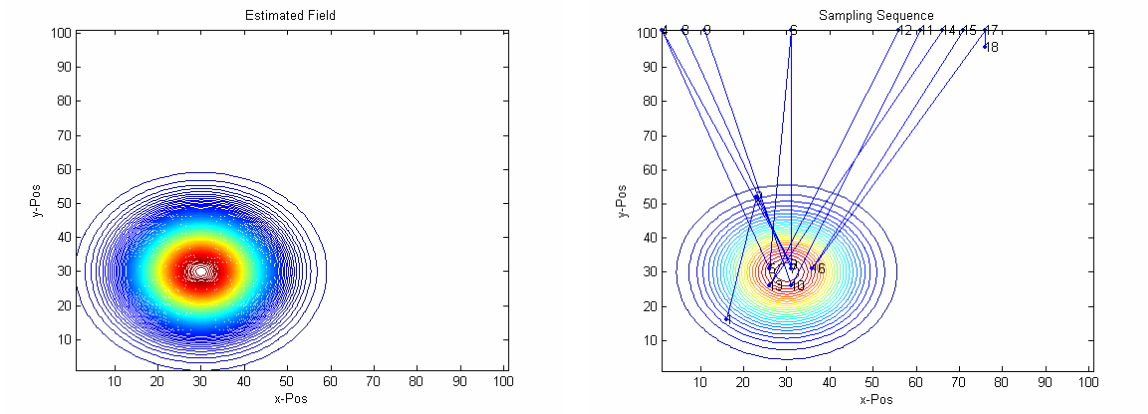


Figure 2.27: Estimated field and sampling locations (nonlinear optimization for field and basis parameter estimation).

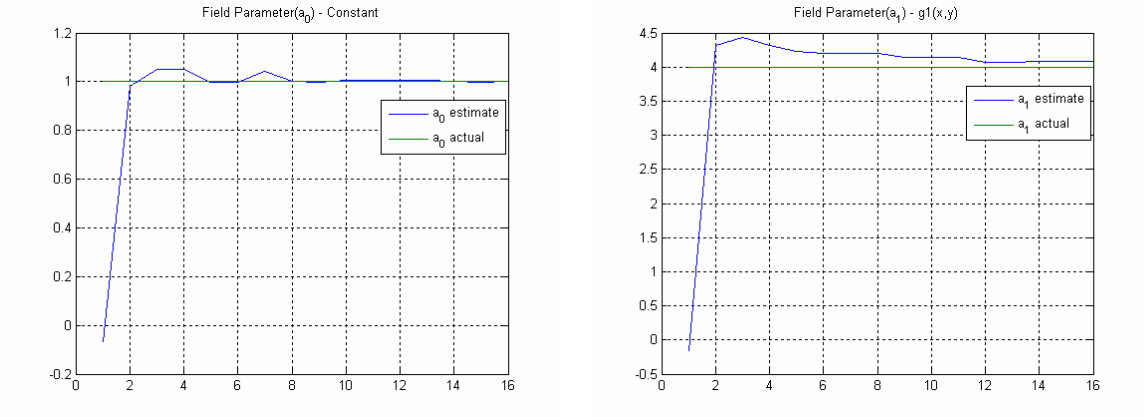


Figure 2.28: Field parameter convergence (nonlinear optimization for field and basis parameter estimation).

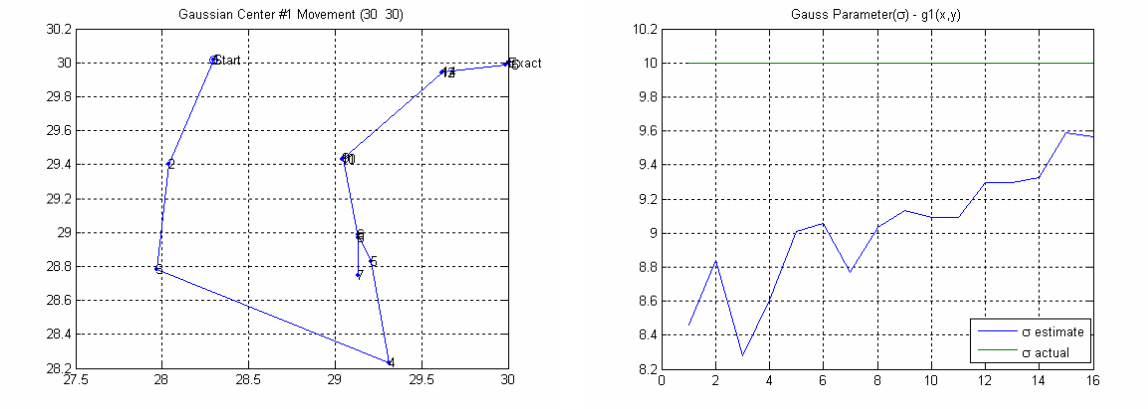


Figure 2.29: Field basis parameter convergence (nonlinear optimization for field and basis parameter estimation).

2.2.5 Estimation of a parameterized field with localization uncertainty

We estimate a linear field with uncertainty in localization as described in section 2.1.5. Initial conditions of state is takes as $X_0 = [0 \quad 0 \quad 0 \quad 0 \quad 0]^T$, with very high uncertainty in the initial estimate $P_0 = 10^{10} I_5$, with state uncertainty

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \tag{2.52}$$

where the localization uncertainty $Q_1 = 0.1 I_2$, and field parameter uncertainty $Q_2 = 0_3$, and the measurement uncertainty as

$$R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \tag{2.53}$$

with $R_1 = 0.1 I_2$, and $R_2 = 0.1$.

Figure 2.30 depicts the original linear field, Figure 2.31 shows the field parameter convergence, and Figure 2.32 shows the sampling locations for simulations with and without localization uncertainty.

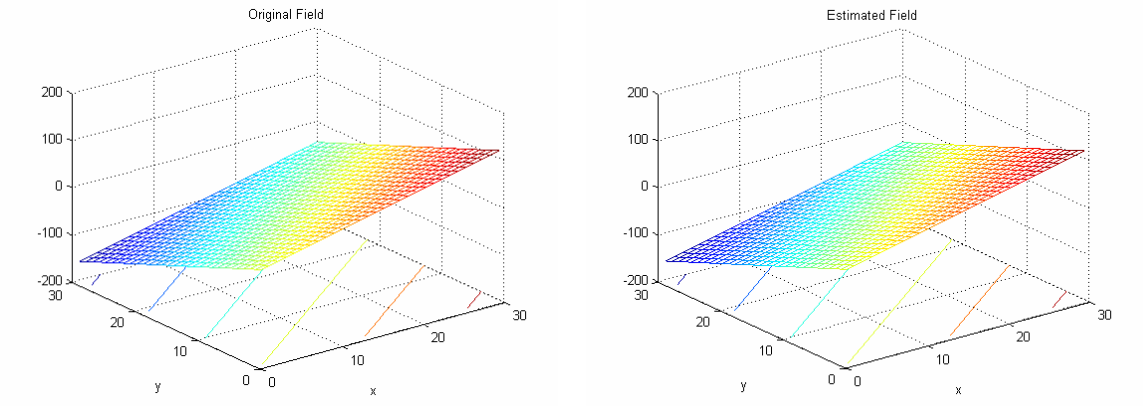


Figure 2.30: Original and estimated linear fields (with localization uncertainty).

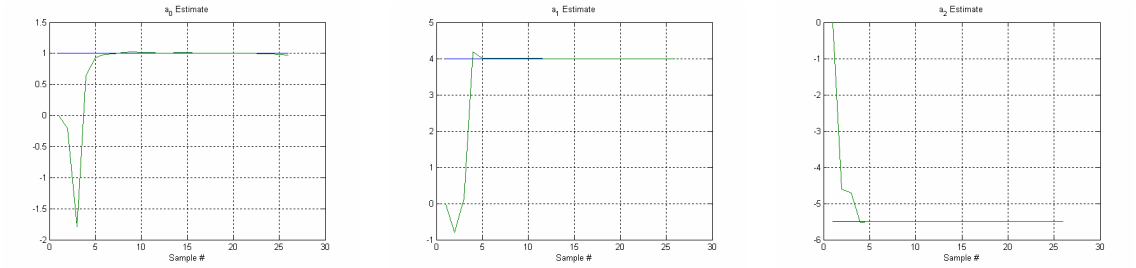
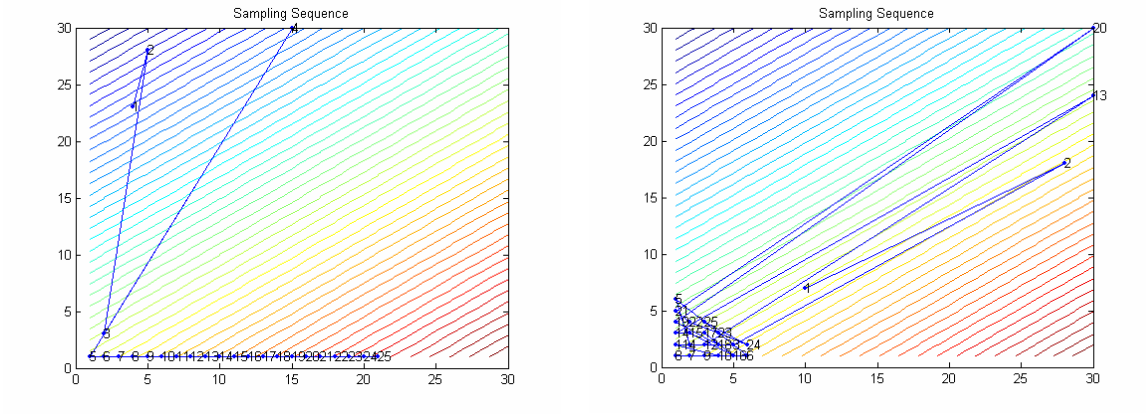


Figure 2.31: Field parameter coefficient convergence for linear field estimation (with localization uncertainty).



(a) (b)

Figure 2.32: Sampling locations for linear field estimation. (a) With localization uncertainty, and (b) Without localization uncertainty.

A Gaussian field is entirely estimated (both field parameters and basis parameters) with localization uncertainty. An initial state

$$X_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sigma + 0.1 * randn \\ x_c + 0.1 * randn \\ y_c + 0.1 * randn \end{bmatrix} \tag{2.54}$$

with very large uncertainty in the initial estimate, $P_0 = 10^{10} I_7$ is chose, with same localization uncertainty, Q_1, and $Q_2 = 0_7$, and with same measurement uncertainty.

Figure 2.33 depicts the original Gaussian field, Figure 2.34 illustrates the estimated

field and the sampling locations where estimation is done with localization uncertainty, Figure 2.35 shows the convergence graphs for field parameters, and Figure 2.36 shows the convergence graphs for basis parameters.

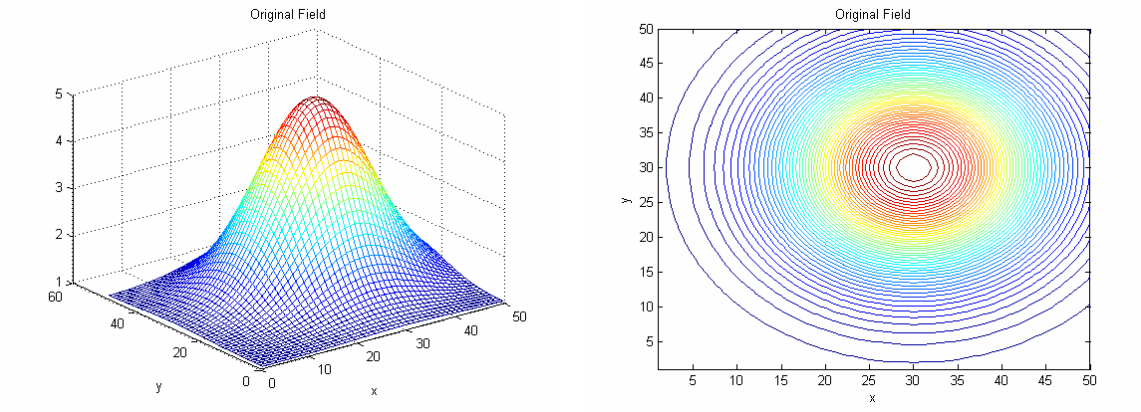


Figure 2.33: Original Gaussian field (with localization uncertainty).

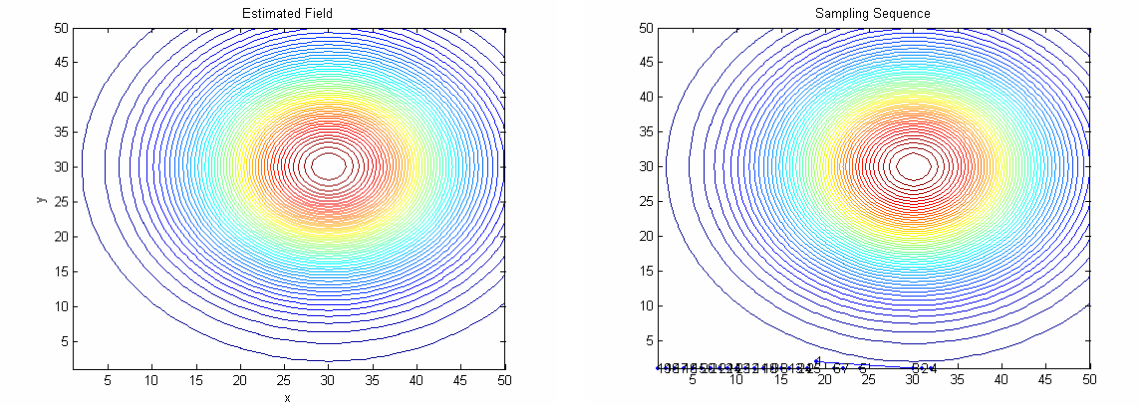


Figure 2.34: Estimated Gaussian field and sampling locations (with localization uncertainty).

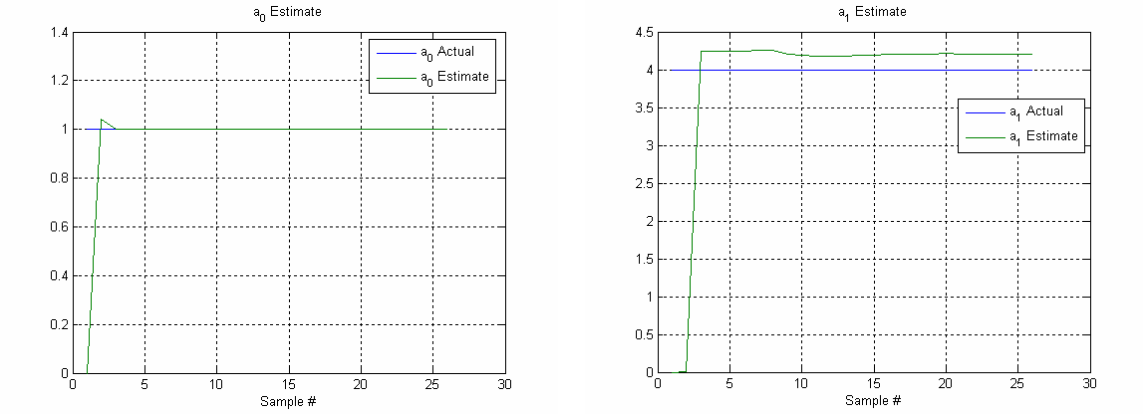


Figure 2.35: Field parameter coefficient convergence for Gaussian field estimation (with localization uncertainty).

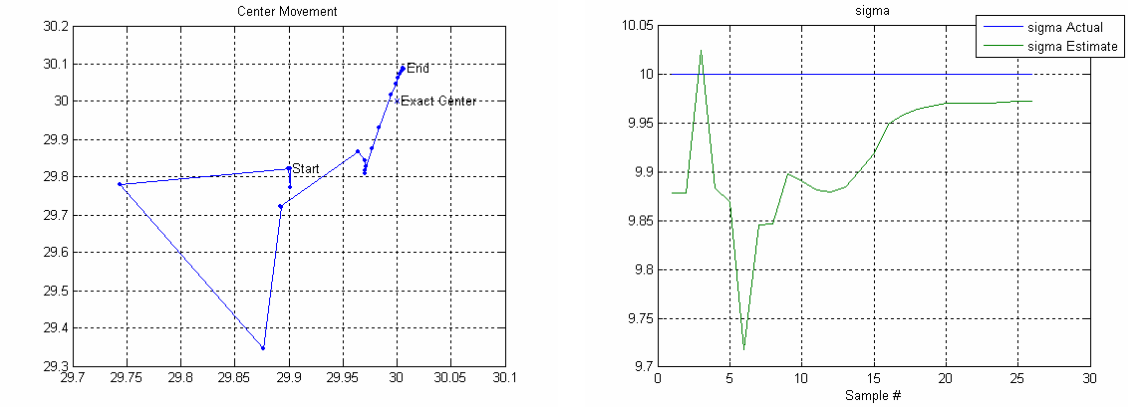


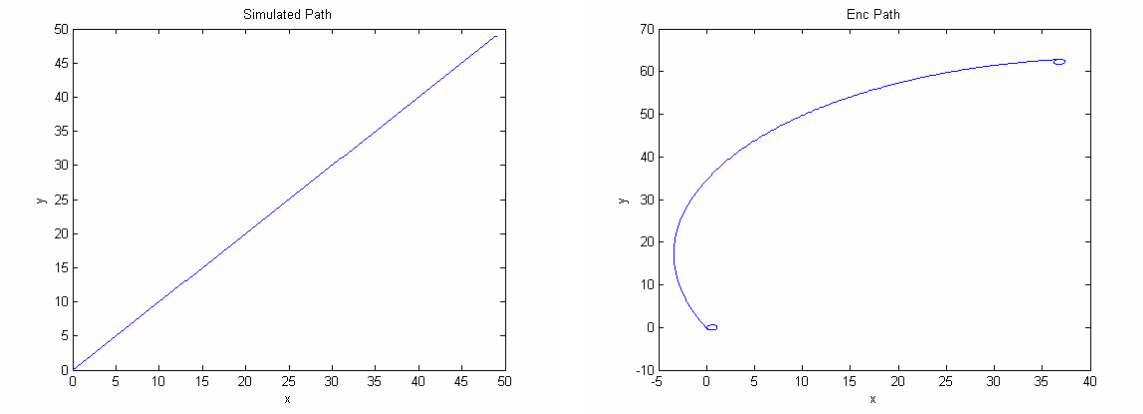
Figure 2.36: Field basis parameter convergence for Gaussian field estimation (with localization uncertainty).

2.2.6 Differential robot simulation

The differential robot model discussed in section 2.1.6 is simulated with a quasi-holonomic controller to navigate from an initial location of $(0,0)$ to a destination of $(50,50)$ with a simple Kalman filter serving to fuse measurement updates from the GPS.

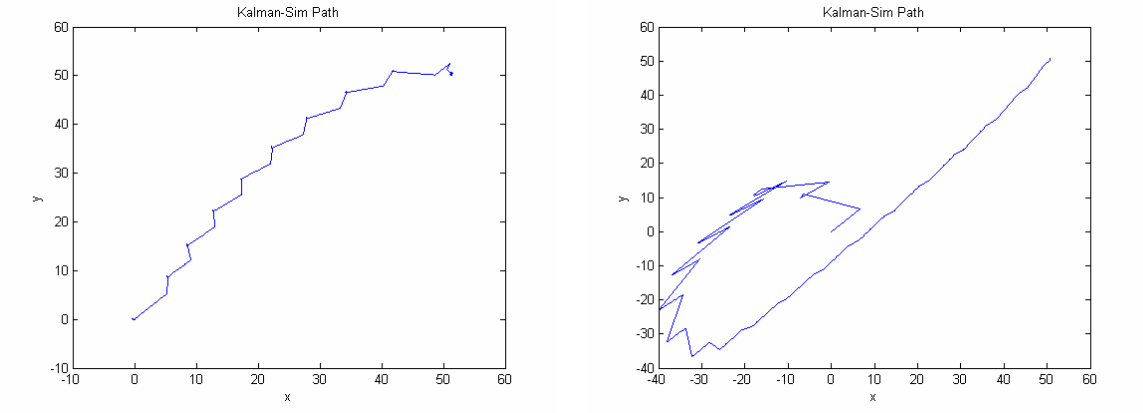
Since the differential robot is injected with systematic errors, the commanded inputs do not drive it along the desired path to the destination. Figure 2.37(a) shows the estimated robot path (where the robot thinks it is), which is different from the actual robot position, Figure 2.37(b). Figure 2.38 shows the improvement in controlling the

position of the robots where information from the GPS (position and orientation information, and position information only) is fused by the Kalman filter to improve the robot positional estimate.



(a) (b)

Figure 2.37: Robot simulation of (a) estimated and (b) actual positions.



(a) (b)

Figure 2.38: Estimated robot position, improved with updates from GPS. (a) Position and orientation updates, and (b) Position updates only.

2.3 Experimental Setup and Results

An experimental setup as illustrated in Figure 2.39 has been setup comprising of a $12' x 8'$ sample space with a color generated field printed on large format paper and assembled on the floor, an inexpensive overhead camera at a height of $13'$

encompassing the entire area in it's field of view, a base station where the primary AS algorithm runs and serves as a central dispatcher of resources, and several mobile robotic sensors. The mobile sensor units (ARRI rovers shown in Figure 2.40) are inexpensive (below USD 500 per unit), are equipped with wheel encoders for localization, a color sensing module for taking color samples, and a RF communication card which serves as the link between the various robotic sensors and the base station in a star topological network.

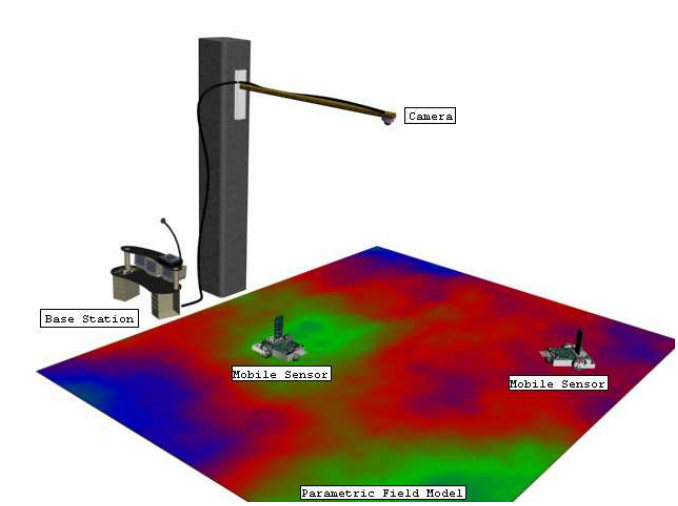
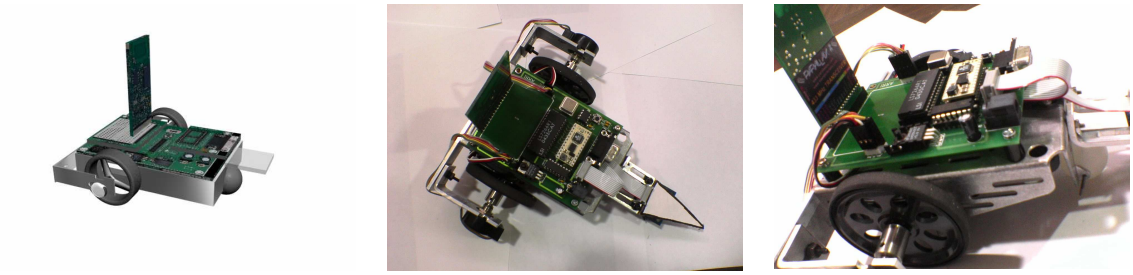


Figure 2.39: Illustration depicting the experimental setup.



(a) (b) (c)

Figure 2.40: ARRI Rover (a) Robot model, (b) Top view, and (c) Perspective view.

2.3.1 Field model and description of uncertainty

A simple linear field, linear in three color components of red, green, and blue, is used for initial algorithm validation. The field is given as

$$\begin{aligned} R &= r_0 + r_1 x + r_2 y \\ G &= g_0 + g_1 x + g_2 y \\ B &= b_0 + b_1 x + b_2 y \end{aligned} \tag{2.55}$$

with nominal field parameters chosen as (Field in Figure 2.41)

$$\begin{aligned} r_0 &= 0.2307, \quad r_1 = 0.0061, \quad r_2 = -0.0026; \\ g_0 &= 0, \quad g_1 = 0.0010, \quad g_2 = 0.0096; \\ b_0 &= 1.0, \quad b_1 = -0.0040, \quad b_2 = -0.0053; \end{aligned} \tag{2.56}$$

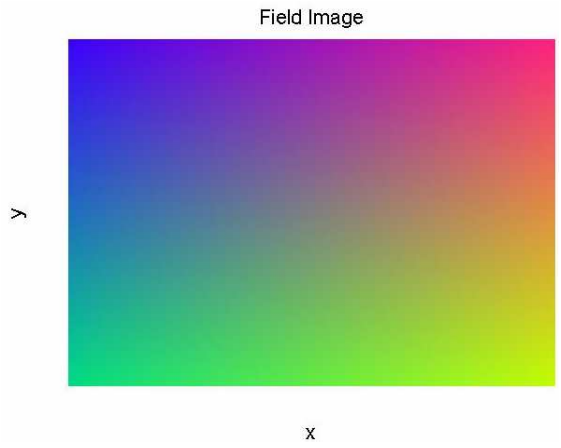


Figure 2.41: True color field generated for printing using field parameters in equation (2.56).

A rectangular grid array is used to correct the wide-angle lens distortion, Figure 2.42, and MATLAB image acquisition and processing toolboxes are used to calculate the pose of the robot using the camera system using image segmentation, Figure 2.43.

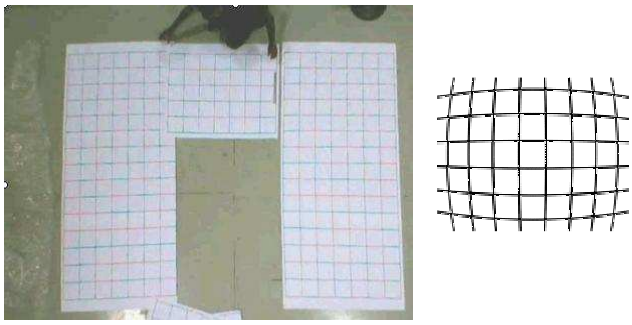


Figure 2.42: The grid used to correct camera lens distortion.

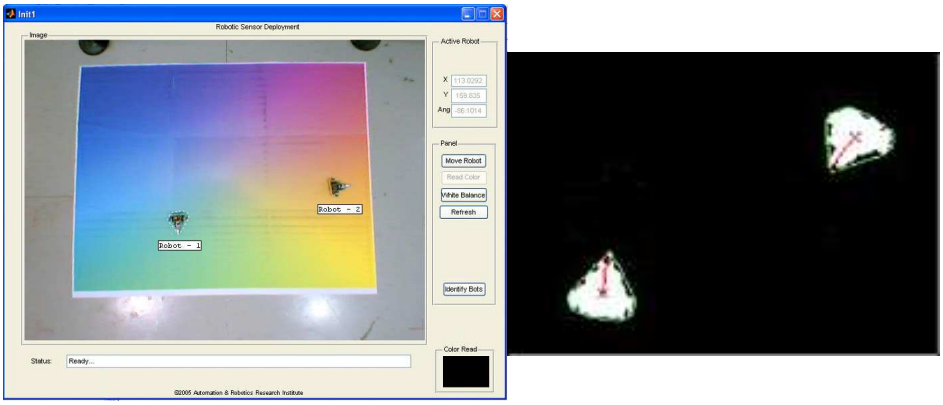


Figure 2.43: The MATLAB GUI showing a RGB field and an example of a segmented image for robot localization.

2.3.2 Experimental adaptive sampling for linear field estimation

Adaptive sampling has been used to estimate a linear field without localization uncertainty of the mobile sensor. The base station runs the adaptive sampling algorithm and commands the mobile sensor to sample at a particular location. The mobile sensor navigates to the location by dead reckoning and thus could end up at a location not exactly matching the commanded one. A camera update at this point ensures that the sampling location used is the current mobile sensor's position. Thus we can ignore localization uncertainty for the moment and iterate the adaptive sampling algorithm to

generate the next sampling location. The experiment used twenty five sampling location to estimate the field.

A simple Kalman filter is setup to estimate the field. The state to be estimated is the compound tri-field representing the three different linear fields of red, green and blue, given as

$$X = \begin{bmatrix} r_0 & r_1 & r_2 & g_0 & g_1 & g_2 & b_0 & b_1 & b_2 \end{bmatrix}^T \tag{2.57}$$

The field is assumed to be stationary and thus does not evolve temporally. The Kalman filter time-update equation illustrates this

$$\begin{aligned} \hat{X}_{k+1}^{-} &= X_k \\ P_{k+1}^{-} &= P_k + Q \end{aligned} \tag{2.58}$$

The measurement-update equation fuses the information obtained after sampling a particular location

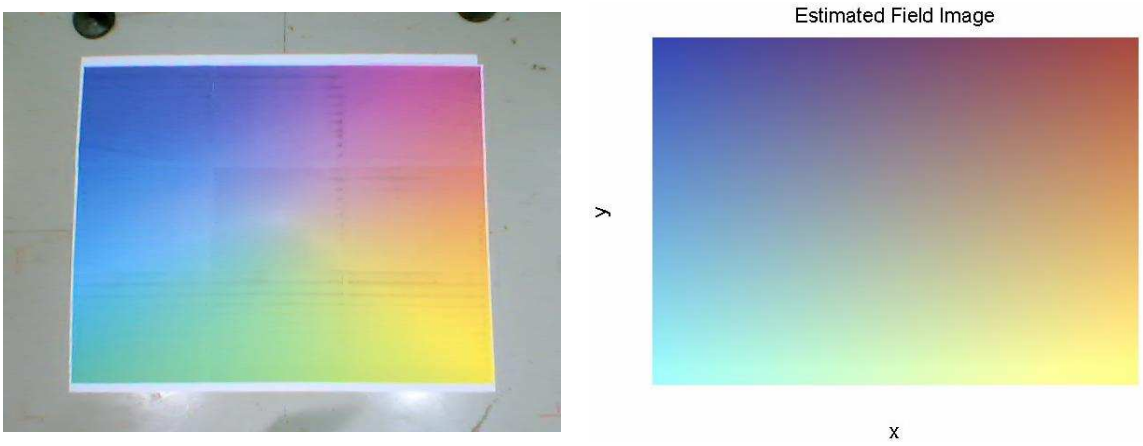
$$\begin{aligned} K_{k+1} &= P_{k+1}^{-} H_{k+1}^T \left(H_{k+1} P_k H_{k+1}^T + R_k \right)^{-1} \\ P_{k+1} &= \left(I - K_{k+1} H_{k+1} \right) P_{k+1}^{-} \\ \hat{X}_{k+1} &= \hat{X}_{k+1}^{-} + K_{k+1} \left(Z_{k+1} - H_{k+1} \hat{X}_{k+1}^{-} \right) \end{aligned} \tag{2.59}$$

where the observation matrix is given as

$$H = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x & y \end{bmatrix} \tag{2.60}$$

Based on the height and resolution of the CCD camera, we estimate a measurement uncertainty of $\pm 2.5\,cm$. The color sensor (TAOS TCS 230) measurement uncertainty, given by the number of discrete RGB values it can measure with, is expressed in RGB units as unity.

Figure 2.44 shows the actual field and the estimated field after 25 samples using the adaptive sampling algorithm. Our eye can not tell the difference; however, in reality there are errors due to color printing, color sensor and localization errors. Figure 2.45 shows the various sampling locations of the mobile robotic sensor. Figure 2.46 illustrates the evolution of the field parameters to values close to nominal as successive samples are taken. The reason for the discrepancy is not color measurement error but rather differences between screen and printer colors.



(a) (b)

Figure 2.44: (a) True printed field and (b) estimated field using adaptive sampling (25 samples).

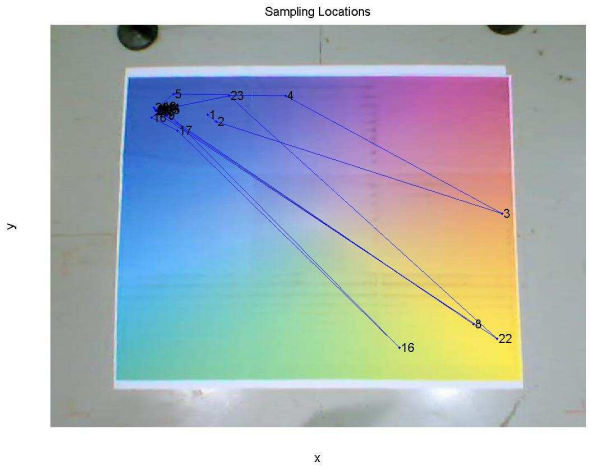


Figure 2.45: Sampling locations superimposed on the field image view from the overhead camera.

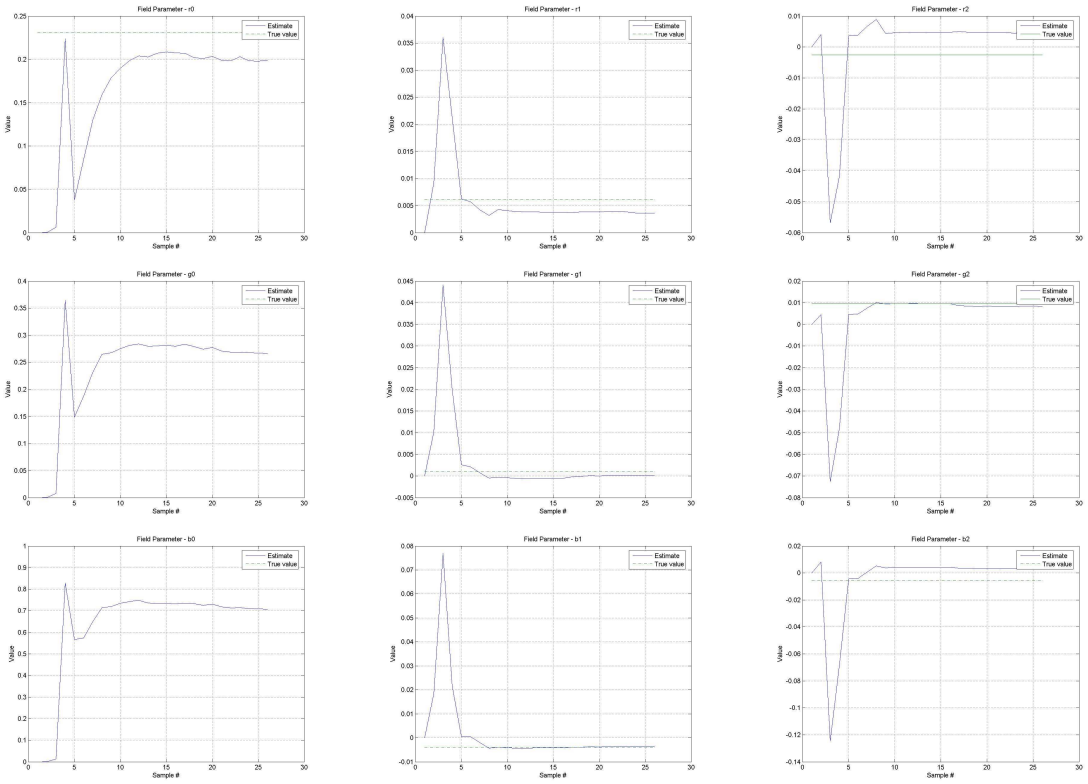


Figure 2.46: Field parameter convergence graphs for the 9 unknown field coefficients.

2.4 Summary

Extensive simulations for field estimation using adaptive sampling algorithms have been discussed involving various approaches for linear and Gaussian fields, with and without localization uncertainty. An experimental setup with mobile robotic sensors, a sample color field, and an overhead camera system as GPS has been constructed. Experimental validation of adaptive sampling approaches for field estimation has been demonstrated.

CHAPTER 3

RESOURCE SCHEDULING

In Manufacturing systems, resources are usually application specific with slight flexibility of resource assignment to tasks, whereas in mobile sensor networks, the resources are heterogeneous and capable of performing diverse tasks. Hence we have shared resources where multiple tasks contend for a single shared resource, or multiple resources contend to perform a single task. In the former case we have shared resources, and in the latter, routing resources. The need then arises to suitably assign, dispatch, schedule resources in such a manner so as to avoid contention, or circular wait of resources leading to deadlock.

This chapter is organized into the following sections. Section 3.1 discusses the matrix-based discrete event controller, section 3.2 introduces deadlocks and presents the deadlock avoidance policy along with implementation on the WSN test bed, section 3.3 discusses the issues of deadlock avoidance in the presence of routing resources, and section 3.4 concludes the chapter.

3.1 Matrix-based Discrete Event Controller

A patented matrix formulation [61] is presented for modeling and analysis of complex interconnected DE systems needing dynamic online resource assignment in the presence of shared resources. The discrete event controller (DEC) is a hybrid system

with logical and algebraic components that allows fast, direct design and reconfiguration of rule-based controllers [62]. The matrix approach provides a rigorous, yet intuitive mathematical framework to represent the dynamic evolution of DE systems through linguistic *if-then* rules:

Rule i: *If* <condition^i> then <consequent^i>

The framework of the Discrete Event Controller is described which provides a rigorous simple representation of these linguistic rules. Let r be the set of resources in the system (e.g., various mobile robots and UGSs), v the set of tasks that the resources can perform (e.g., take a sensor reading, navigate to a commanded location along a desired path, and retrieve/deploy UGS), u the set of inputs that trigger the system (e.g., detection of events such as chemical alert, intruder alert, etc., node failures), y the set of outputs indicating completed missions, and x the logical state vector of rules of the DE controller indicating the activated rules of the supervisory control policy.

The condition and consequent of each rule are segregated by the two sets of logical equations, one for checking the prior conditions leading to the activation of rule *i* (matrix controller state equation), and one for determining the a priori consequent of rule *i* (matrix controller output equation). The logical equations make use of matrix algebra for multiplications and additions with the element multiplications replaced by logical-*and* operations and the element additions replaced by logical-*or* operations. Logical negations are indicated by overbars.

The matrix controller state equation is

$$\bar{x} = F_v \bar{v} + F_r \bar{r} + F_u \bar{u} + F_{uc} \bar{u}_c \tag{3.1}$$

where F_v is the task sequencing matrix, F_r the resource requirements matrix, F_u the input matrix, F_{uc} the conflict resolution matrix, and u_c the conflict resolution vector. u_c along with F_{uc}, is used to inhibit simultaneous activation of conflicting rules. The state of the DE system is maintained in the x, v, r, and u vectors whose active (*true*) entries indicate the activated rules, the completed tasks, the available resources and the occurrence of events respectively.

The task sequencing matrix F_v has element (i, j) set if the completion of task v_j is an immediate prerequisite for the activation of logic state x_i. The resource requirements matrix F_r has element (i, j) set if the availability of resource r_j is an immediate prerequisite for the activation of logic state x_i.

The matrix controller state equation, eq. (3.1), defines the prior conditions required for the activation of for a rule, while the matrix controller output equation, eqs. (3.2-3.4), define the a priori consequents of a rule.

The matrix controller output equations are

$$v_s = S_v x \tag{3.2}$$

$$r_s = S_r x \tag{3.3}$$

$$y = S_y x \tag{3.4}$$

where S_v is the task start matrix having element (i, j) set if logic state x_j determines the activation of task v_i, S_r is the resource release matrix having element (i, j) set if logic state x_j determines the release of resource r_i.

Equations (3.1-3.4) represent the rule-base of the supervisory control of the DE system. All the matrices are composed of sparse boolean entries, so that real time control of large interconnected DE systems with multiple missions and routes is computationally feasible.

The DEC has been used as a supervisory control in mobile sensor networks as detailed in [63].

Figure 3.1 illustrates the DEC architecture.

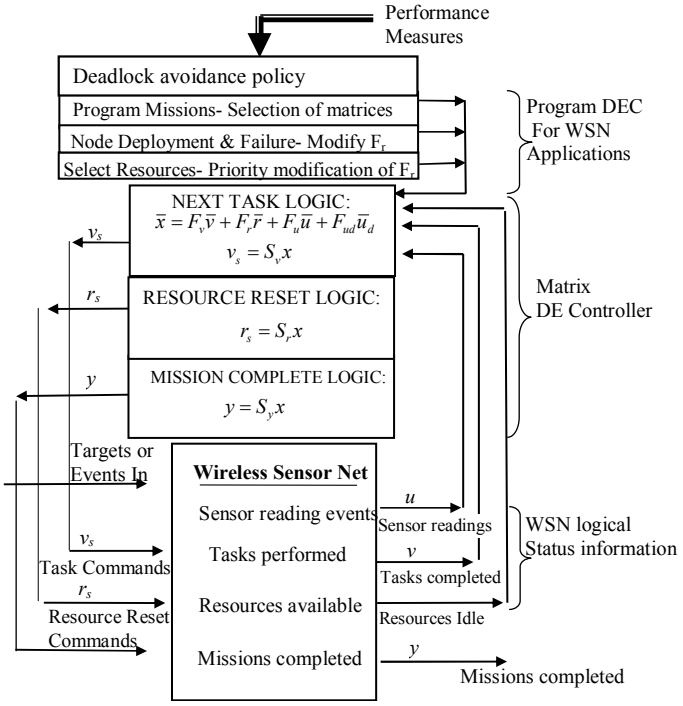


Figure 3.1: Discrete Event Control architecture.

3.2 Deadlock

Deadlock research in computer systems has focused on four main areas. *Deadlock prevention* is involved with removing any possibility of system deadlocks; the result is often over-conservative polices resulting in poor utilization of resources.

Deadlock detection focuses on detecting imminent or current deadlocks, and is required for deadlock recovery and avoidance strategies. *Deadlock recovery* methods are used to clear deadlocks once they occur, often by placing jobs in buffers, or by completely flushing one or more of the deadlocked processes, resulting in lost work. In *deadlock avoidance* the possibility of system deadlock is not totally removed, but whenever a deadlock is imminent, it is sidestepped by a real-time decision-making procedure [24]. In this thesis, we focus on deadlock avoidance.

3.2.1 Deadlock avoidance policy

Deadlock-free dispatching rules are derived by performing circular wait (CW) analysis in matrix form for possible deadlock situations. An analysis of deadlocks in manufacturing systems using the matrix based DEC is presented in [24, 37]. Deadlock avoidance algorithms have been implemented in robotic cells in manufacturing systems using the DEC [25, 26]. Preliminary Analysis of deadlock avoidance policies for shared resources in heterogeneous mobile sensor networks is presented in [30].

For deadlocks, we consider the following assumptions

- No preemption - No resource can be removed from a task until the completion of the task.
- Mutual exclusion - Every resource performs only one task at a given time.
- Hold while waiting – A process holds the resources allocated to it until it has all resources required to perform a job.

Under these assumptions, a necessary condition for deadlock to occur is the presence of a circular wait relation among the resources [24, 26, 35-37].

For any two resources r_i and r_j, r_i is said to wait for r_j, denoted by $r_i \rightarrow r_j$, if the availability of r_j is an immediate requirements for the release of r_i. Circular waits (CW) among resources are a set of resources r_a, r_b,… r_w whose wait relationship among them are $r_a \rightarrow r_b \rightarrow \cdots \rightarrow r_w$ and $r_w \rightarrow r_a$. To identify simple Circular Waits (sCW), a wait relation digraph of resources needs to be constructed. The digraph of resources is easily obtained from the matrix formulation

$$W = \left(S_r F_r\right)^T \tag{3.5}$$

where element W_{ij} is set if $r_i \rightarrow r_j$ holds. Simple circular waits are calculated from the digraph matrix using string algebra [25]. The sCW do not represent all the circular waits and we require the circular wait (CW) matrix that is composed of all sCW along with unions of non-disjoint sCW. The Gurel algorithm [35, 36] is used to efficiently compute all CWs and composed CWs, C_{out}.

In order to implement efficient real-time deadlock avoidance policies, other relevant sets of task and resources from Petri net theory need to be defined. The term *token* is used to indicate a task in progress or an available resource and the term *transition* to indicate a rule of the supervisory controller.

A *siphon* is a set of tasks and resources which if token-free after firing of a certain transition, will remain token-free under all subsequent transition firings. The critical siphon of a CW is the smallest siphon containing the CW. If the critical siphon

ever becomes empty, all its resources are busy and can never become available again. The *Siphon- task* set $J_s(C)$ is the set of tasks of the critical siphon. The *critical subsystem* $J_o(C)$ is set of tasks of the CW, $J(C)$ which do not add a token to the CW.

A deadlock condition occurs if and only if there is an empty CW, which corresponds to an empty critical siphon, or equivalently to a condition where all tasks of the CW belong to the critical subsystem.

Thus, in order to perform deadlock analysis, we need matrix computation tools to determine the siphon-task sets $J_s(C)$, and the critical subsystems $J_o(C)$ of every CW C. Since the deadlock conditions are dependent on the number of tokens in these sets, we need to calculate the set of transitions (input and output transitions) which when fired, add or subtract tokens from the CWs.

The input and output transitions of a CW are calculated as

$$\begin{aligned} {}_d C &= C_{out} . S_r \\ C_d &= C_{out} . F_r^{\,T} \end{aligned} \tag{3.6}$$

The adding and clearing transitions are calculated as

$$\begin{aligned} T_p &= {}_d C - ({}_d C \wedge C_d) \\ T_m &= C_d - (C_d \wedge {}_d C) \end{aligned} \tag{3.7}$$

where the $\wedge$ operator represents logical and.

These set of transitions are important in keeping track, in real-time, of the available resources inside every CW, and hence in determining the status of tasks and resources inside the critical siphon. The task set, siphon-task set, and the critical subsystem of a CW is calculated as

$$
\begin{aligned}
J(C) &= {}_d C.F_v = C_d.S_v{}^T \\
J_s(C) &= T_p F_v \\
J_o(C) &= J(C) \wedge (C_d F_v)
\end{aligned}
\tag{3.8}
$$

The critical subsystem is the set of tasks which require one of the resources from the CW for execution. Therefore the activation of all tasks in the critical subsystem will make the CW empty and lead to a deadlock condition. A simple deadlock avoidance strategy consists of keeping the number of activated tasks of a critical subsystem lesser than the number of resources in the corresponding CW. This is the MAXWIP policy

$$
m(J_0(C_i)) < m_0(C_i) \tag{3.9}
$$

The described deadlock avoidance policy has been implemented on the mobile wireless sensor network test bed and complex interconnected missions executed in a smooth, deadlock-free manner.

3.2.2 Implementation of DEC on WSN test bed

The wireless sensor network test bed at the Automation & Robotics Research Institute comprises of mobile sentry robots, unattended ground sensors, a wireless network, and a centralized control unit, see Figure 3.2. Cybermotion SR2 mobile robots serve as the patrol robots and Berkley motes serve as the UGSs. The base-station PC runs the DE controller, and serves as a central supervisor controlling the various resources through a wireless transceiver.

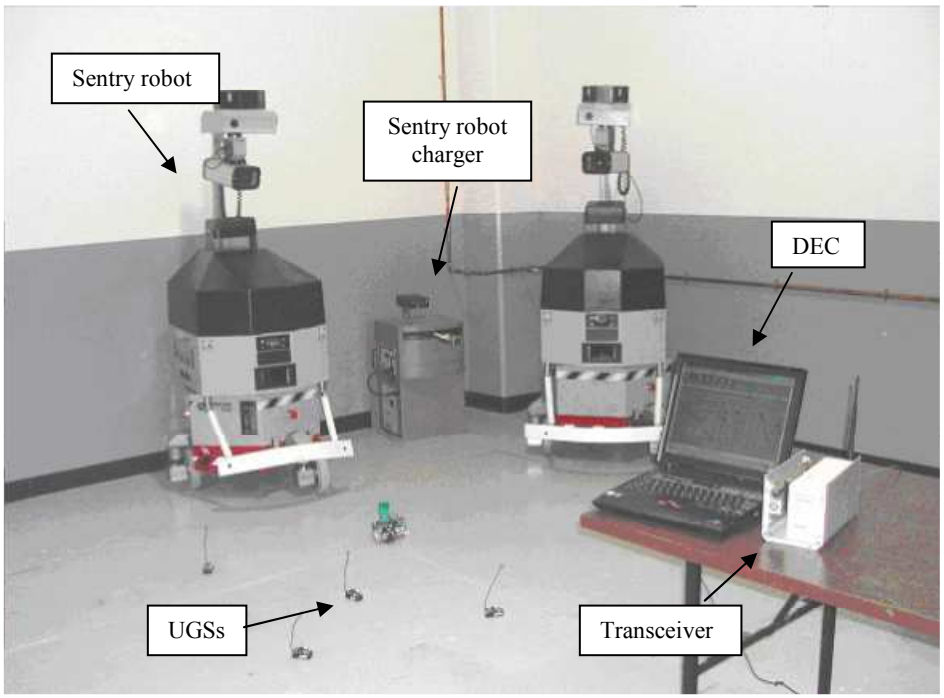


Figure 3.2: The WSN test bed at ARRI.

A virtual WSN test bed has been created to illustrate various mobile robot movements as the WSN topology reconfigures to handle various missions, see Figure 3.3.

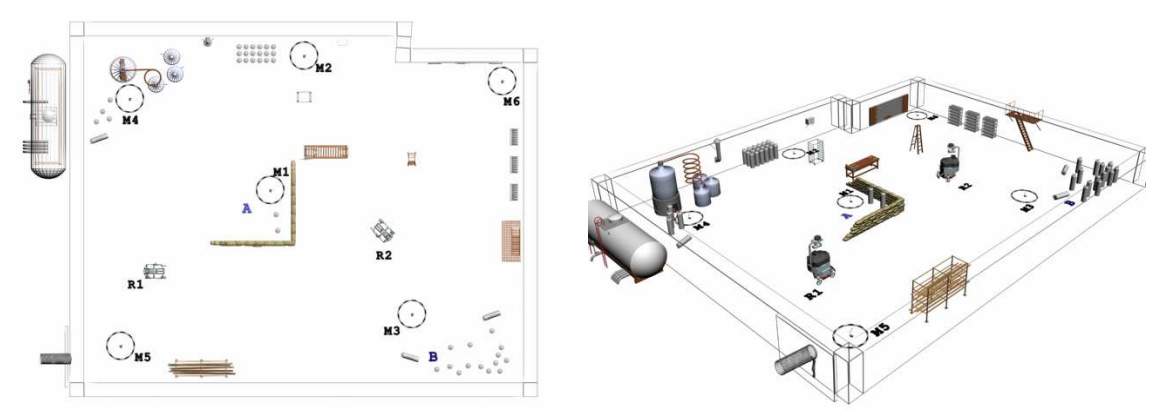


Figure 3.3: Top and perspective views of the virtual WSN test bed in initial network configuration.

3.2.3 Simulation and experimental results

The results presented in this section have been obtained using Matlab and Labview environments. Matlab has been used for initial simulations of the missions, followed by a Labview implementation of the missions with simulated resources. On satisfactory performance of the deadlock avoidance algorithm, the simulated resources have been replaced with actual resources.

Three different missions have been implemented to illustrate the effectiveness of the deadlock avoidance algorithm. All missions use the wireless sensor network comprising of two mobile robots (R_1, R_2), and six Berkley motes $(M_1 - M_6)$ as UGSs. Mission-1 achieves patrolling and sensing of the warehouse, Mission-2 serves to charge the UGSs, and Mission-3 transports dangerous cargo from location 'A' to location 'B'. Missions are triggered by events from sensors, such as the intruder alert, battery low alert, etc. The sensor network reacts to events and could physically reconfigure its topology to adapt to the event.

The Petri net representation of Missions 1-3 is illustrated in Figure 3.4.

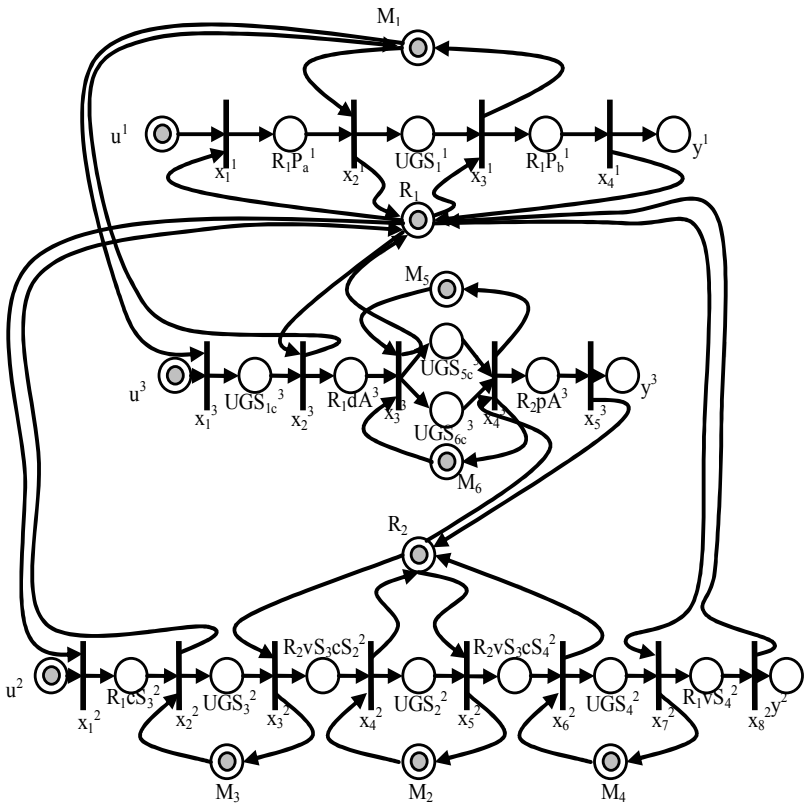


Figure 3.4: Petri net representation of Missions 1-3 (Patrolling, Charging, Transportation).

To implement the supervisory control policy, we define the vector of resources $r = [R_1, R_2, M_1, M_2, M_3, M_4, M_5, M_6]$ of the system consisting of two robots and six stationary sensors. For each mission-i we define the vector of inputs u^i, of output y^i, and of tasks v^i. The task sequence for each mission is defined (Table 3.1, Table 3.3, and Table 3.5 for Missions 1, 2, and 3 respectively) and the *if-then* rules representing the supervisory coordination strategy to sequence the programmed missions are defined (Table 3.2, Table 3.4, and Table 3.6 for Missions 1, 2, and 3 respectively). The linguistic description of the coordination rules is translated into a more convenient

matrix representation suitable for mathematical analysis and computer implementation. As an example, matrices F_v^i, F_r^i (Figure 3.5) relative to mission-1 is illustrated.

Table 3.1: Mission 1 - Task Sequence

Mission-1	Notation	Description
Input	u^1	Intruder Alert from any UGS
Task 1	$R_1P_a^1$	i. R_1 navigates to M_2 ii. R_1 takes measurement at M_2 iii. R_1 navigates from M_2 to M_1 iv. R_1 takes measurement at M_1
Task 2	UGS_1^1	i. M_1 takes measurement
Task 3	$R_1P_b^1$	i. R_1 navigates to M_1 ii. R_1 takes measurement at M_1 iii. R_1 navigates from M_1 to M_3 iv. R_1 takes measurement at M_3
Output	y^1	i. Patrol and sensing of warehouse

Table 3.2: Mission 1 - Rule base.

Mission 1 – Operation Sequence	
Rule 1 x_1^1	If u^1 *occurs* and R_1 *available* then *start* $R_1P_a^1$
Rule 2 x_2^1	If $R_1P_a^1$ *completed* and M_1 *available* then *release* R_1 and *start* UGS_1^1
Rule 3 x_3^1	If UGS_1^1 *completed* and R_1 *available* then *release* M_1 and start $R_1P_b^1$
Rule 4 x_4^1	If $R_1P_b^1$ *completed* then *release* R_1 and *terminate* mission-1 by producing output y^1

Table 3.3: Mission 2 - Task Sequence

Mission-2	Notation	Description
Input	u^2	Low battery warning from UGS
Task 1	$R_1cS_3^2$	i. R_1 navigates to M_3 ii. R_1 charges M_3
Task 2	UGS_3^2	i. M_3 takes measurement
Task 3	$R_2vS_3cS_2^2$	i. R_2 navigates to M_3 ii. R_2 takes measurement and verifies M_3 charge iii. R_2 navigates from M_3 to M_2 iv. R_2 charges M_2
Task 4	UGS_2^2	i. M_2 takes measurement
Task 5	$R_2vS_2cS_4^2$	i. R_2 navigates to M_2 ii. R_2 takes measurement and verifies M_2 charge iii. R_2 navigates from M_2 to M_4 iv. R_2 charges M_4
Task 6	UGS_4^2	i. M_4 takes measurement
Task 7	$R_1vS_4^2$	i. R_1 navigates to M_4 ii. R_1 takes measurement and verifies M_4 charge
Output	y^2	i. Charging of UGSs

Table 3.4: Mission 2 - Rule base

Mission 2 – Operation Sequence	
Rule 1 x_1^2	If u^2 *occurs* and R_1 *available* then *start* $R_1cS_3^2$
Rule 2 x_2^2	If $R_1cS_3^2$ *completed* and M_3 *available* then *release* R_1 and *start* UGS_3^2
Rule 3 x_3^2	If UGS_3^2 *completed* and R_2 *available* then *release* M_3 and *start* $R_2vS_3cS_2^2$
Rule 4 x_4^2	If $R_2vS_3cS_2^2$ *completed* and M_2 *available* then *release* R_2 and *start* UGS_2^2
Rule 5 x_5^2	If UGS_2^2 *completed* and R_2 *available* then *release* M_2 and *start* $R_2vS_2cS_4^2$
Rule 6 x_6^2	If $R_2vS_2cS_4^2$ *completed* and M_4 *available* then *release* R_2 and *start* UGS_4^2
Rule 7 x_7^2	If UGS_4^2 *completed* and R_1 *available* then *release* M_4 and *start* $R_1vS_4^2$
Rule 8 x_8^2	If $R_1vS_4^2$ *completed* then *release* R_1 and *terminate* mission-2 by producing output y^2

Table 3.5: Mission 3 - Task Sequence

Mission-1	**Notation**	**Description**
Input	u^3	Fourty minutes have elapsed
Task 1	UGS_{1c}^3	M_1 takes measurement
Task 2	R_1dA^3	R_1 picks up dangerous cargo and drops off at temporary storage location A
Task 3	UGS_{5c}^3 UGS_{6c}^3	M_5 and M_6 take measurements
Task 4	R_2pA	R_2 picks up cargo from A and transports to location B along path decided by readings from M_5 and M_6
Output	y^3	*Dangerous cargo transported*

Table 3.6: Mission 3 - Rule base

Mission 3 – Operation Sequence	
Rule 1 x_1^3	If u^3 *occurs* and M_1 *available* then *start* UGS_{1c}^3
Rule 2 x_2^3	If UGS_{1c}^3 *completed* and R_1 *available* then *release* M_1 and *start* R_1dA^3
Rule 3 x_3^3	If R_1dA^3 *completed* and M_5 and M_6 *available* then *release* R_1 and *start* UGS_{5c}^3 and UGS_{6c}^3
Rule 4 x_4^3	If UGS_{5c}^3 and UGS_{6c}^3 *completed* and R_2 *available* then *release* M_5 and M_6 and *start* R_2pA
Rule 5 x_5^3	If R_2pA then *release* R_2 and *terminate* mission-3 by producing output y^3

$$F_v^{\,1} = \begin{array}{c} \begin{matrix} R_1P_a & UGS_1 & R_1P_b \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}, \quad F_r^{\,1} = \begin{array}{c} \begin{matrix} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & R_1 & R_2 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 3.5: Mission 1 Job sequencing and Resource requirement matrices.

The circular wait matrix C_{out} and the critical subsystem matrix J_0 are fairly complex for these interconnected missions. Figure 3.6 illustrates these matrices.

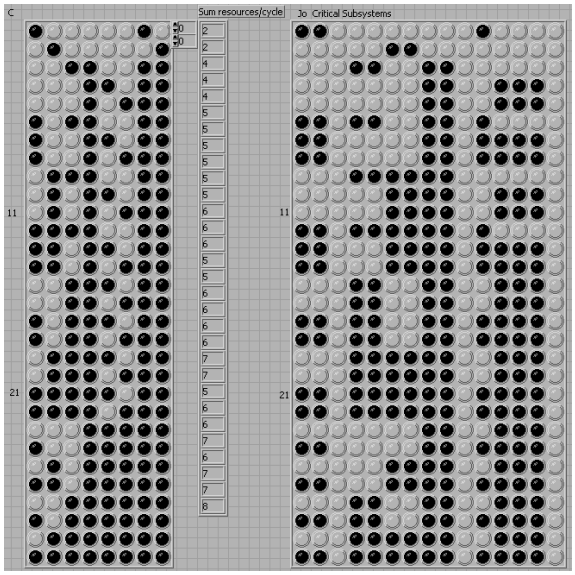


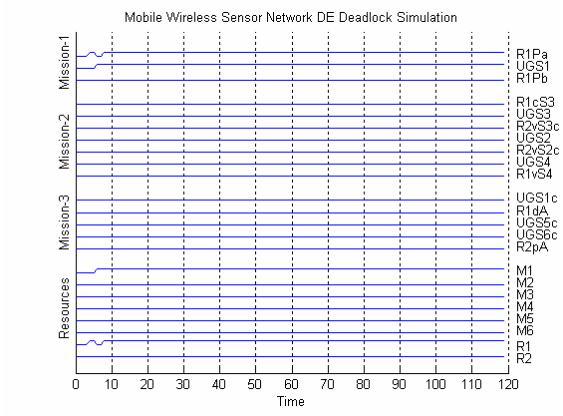
Figure 3.6: Circular wait and Critical subsystem matrices.

Figure 3.7(a,c,e) illustrate the time traces of the discrete event system if no deadlock avoidance policy has been applied. In these time traces, idle resources and tasks not in progress are denoted by low level, whereas busy resources and tasks in progress are denoted by high level.

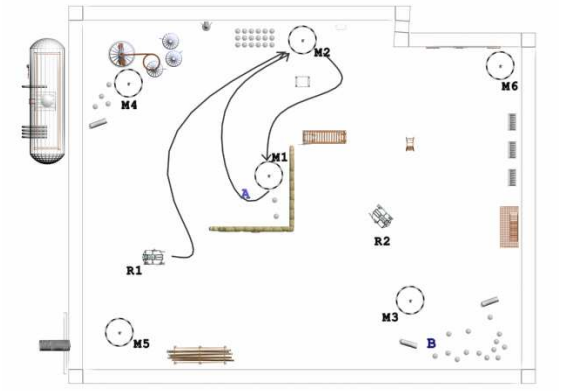
Figure 3.7 shows the simple case of deadlock of mission 1 when it is triggered multiple times. With the initial trigger, task R_1P_a executes to completion and task UGS_1

starts. At this time, a second trigger of the mission causes a second instance of R_1P_a to start simultaneously and both resources R_1 and M_1 are consumed. For UGS_1 to complete and R_1P_b to begin, R_1 is required, but this is being used by R_1P_a. For R_1P_a to complete and UGS_1 to begin, M_1 is required, but this is being used. Thus we have a cyclic wait of resources which lead to a deadlock situation. This cyclic wait of resources can be seen easily in the first row of the circular wait matrix in Figure 3.6 and a deadlock occurs when the corresponding tasks in the critical subsystem matrix are simultaneously in progress. Thus to avoid deadlocks, the dispatching policy has to ensure that all tasks in a particular row of the critical subsystem matrix are not in progress simultaneously. In the case of mission-1, when UGS_1 is in progress, rule-1 has to be inhibited by updating the conflict resolution vector u_d. Figure 3.8 shows the deadlock avoidance policy in effect which smoothly takes mission-1 to completion twice. The dispatching policy is capable of handling deadlocks of higher order which arise when both mission-1, and mission-2 are triggered multiple times, as illustrated in Figure 3.9, with no deadlock avoidance, and in Figure 3.10 with deadlock avoidance.

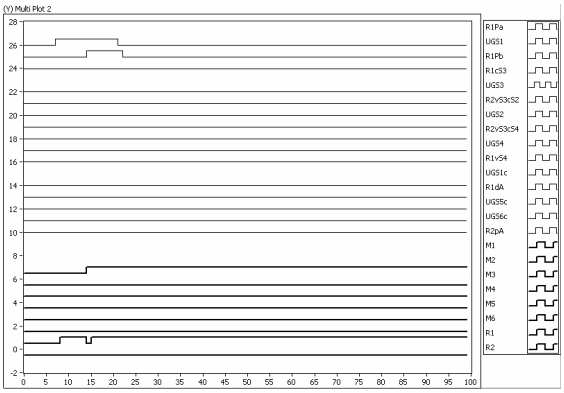
Deadlocks can also arise when two or more missions run in parallel and there exists a circular wait between the missions. This scenario is illustrated in Figure 3.11 where deadlock arises due to a circular wait of resources M_1 in mission-3, and R_1 in mission-1. Figure 3.12 illustrates that the same dispatching policy handles complex deadlocks between missions where task R_1P_a from mission-1 is inhibited until task UGS_{1c} from mission-3 is completed.



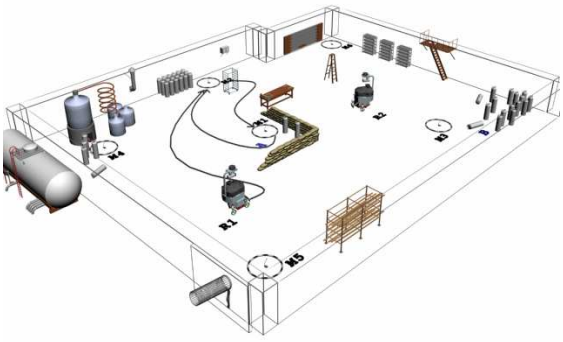
(a)



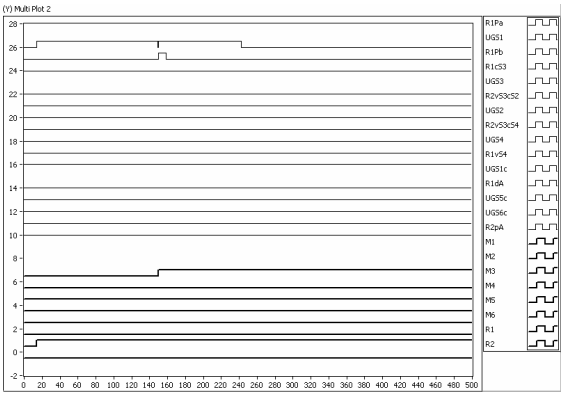
(b)



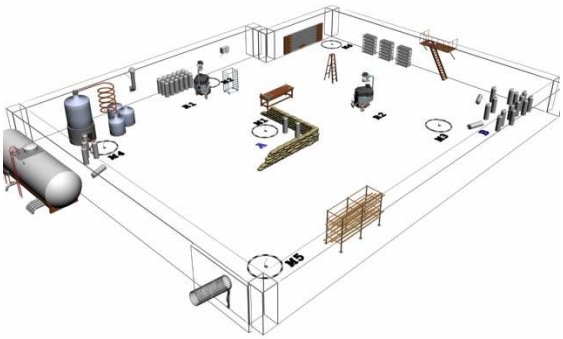
(c)



(d)



(e)



(f)

Figure 3.7: Mission 1 with deadlock. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.

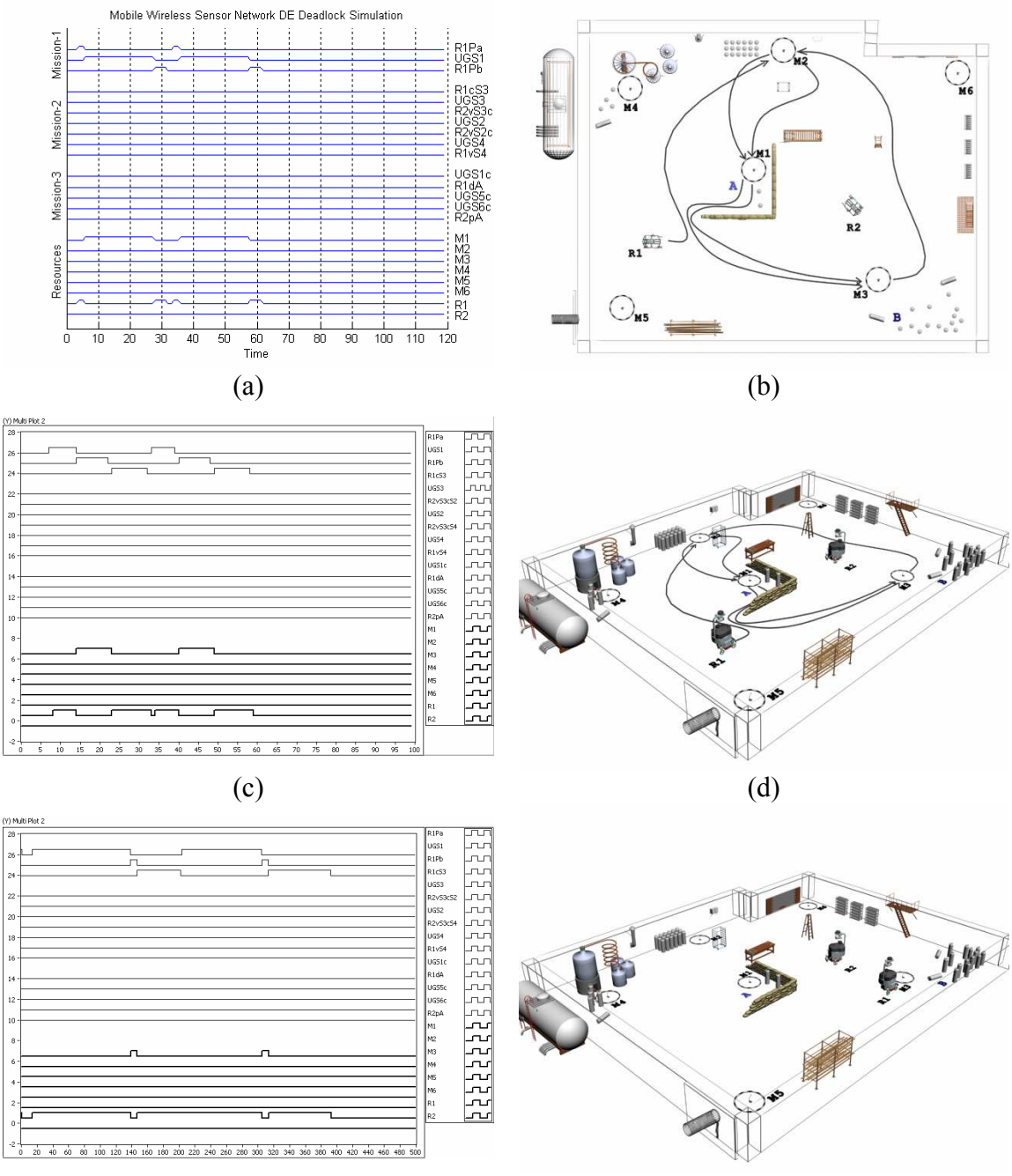
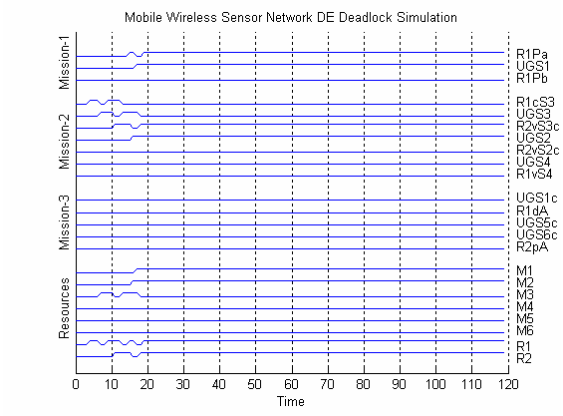
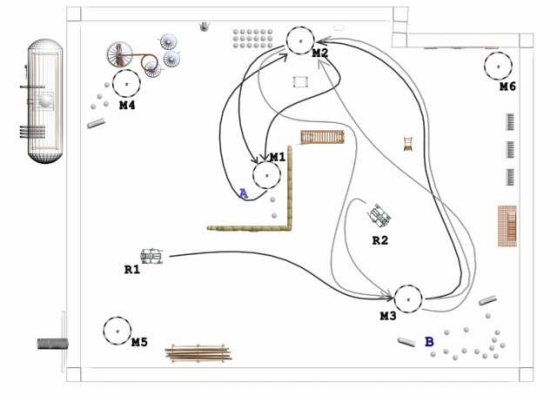


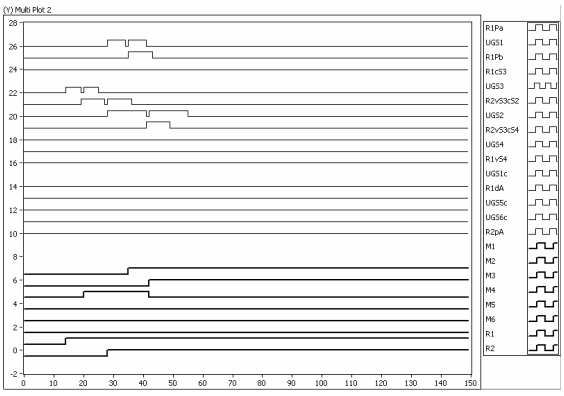
Figure 3.8: Mission 1 with deadlock avoidance. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.



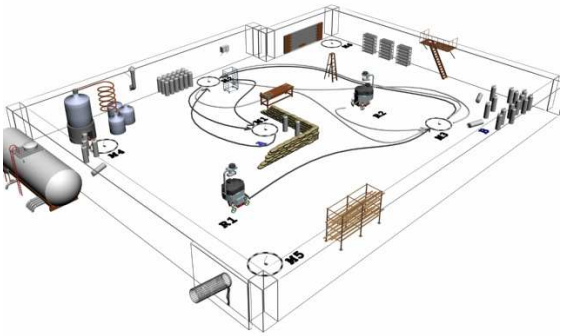
(a)



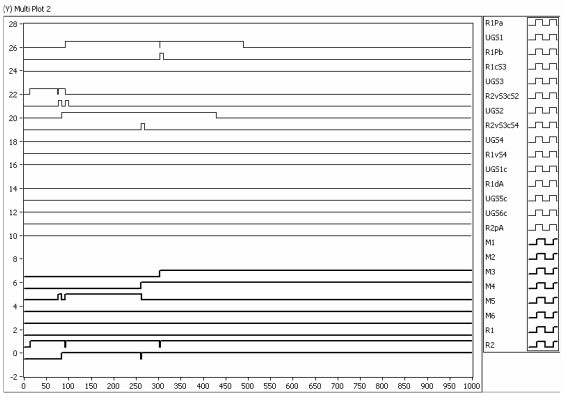
(b)



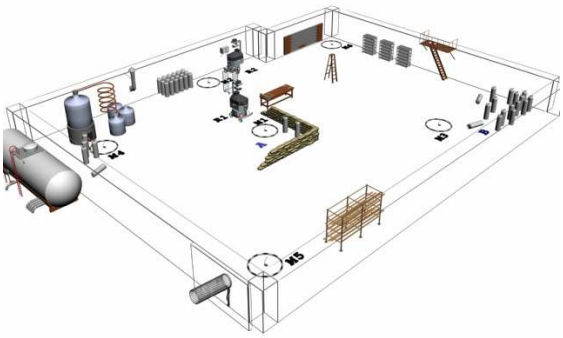
(c)



(d)

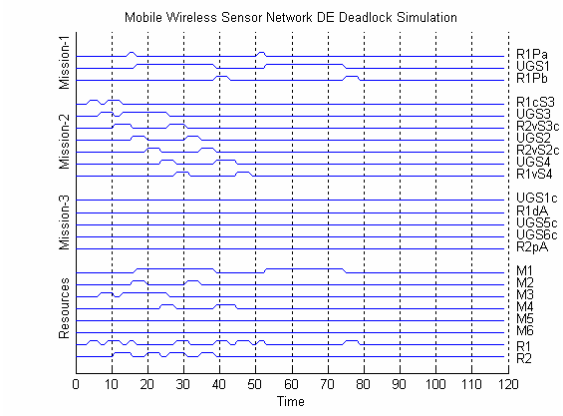


(e)

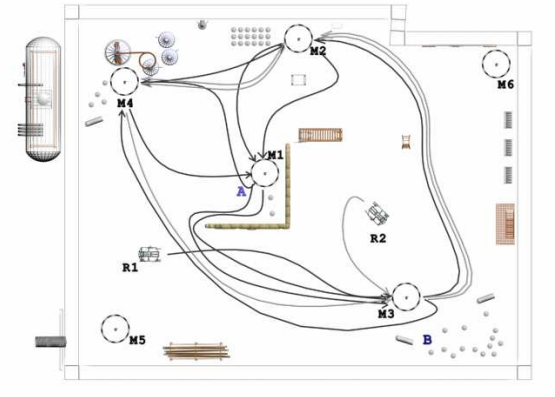


(f)

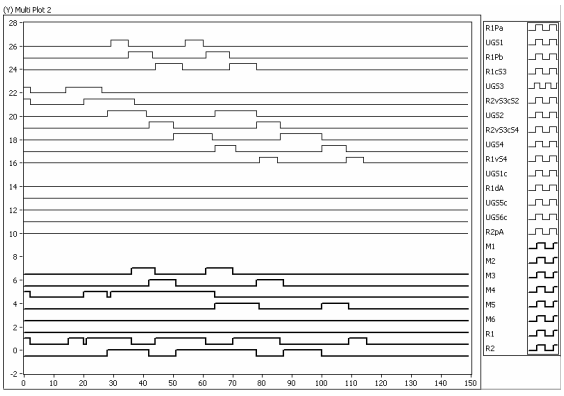
Figure 3.9: Mission 1, 2 with deadlock. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.



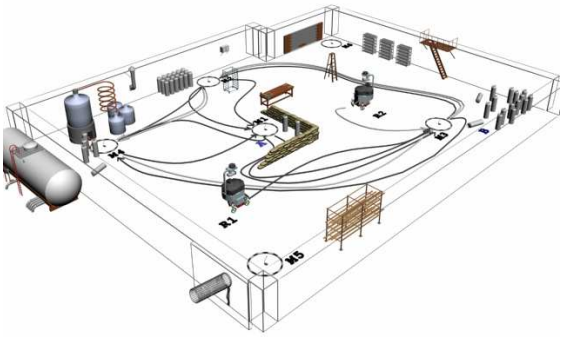
(a)



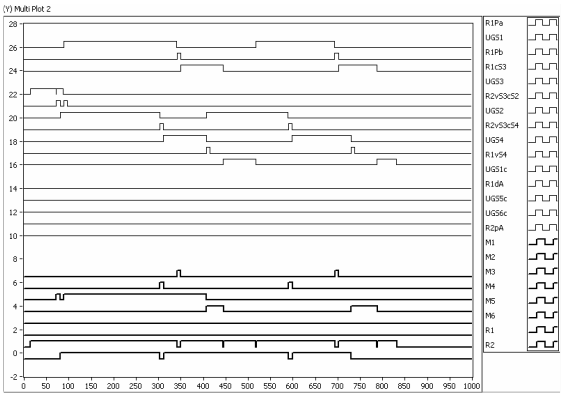
(b)



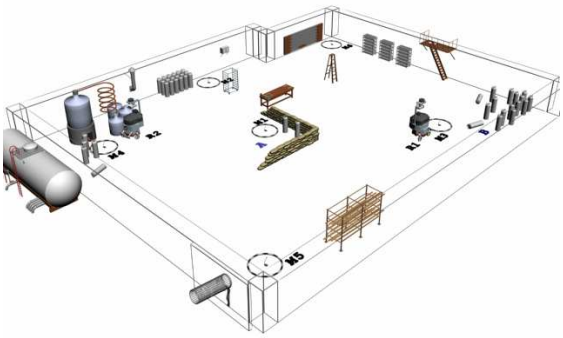
(c)



(d)

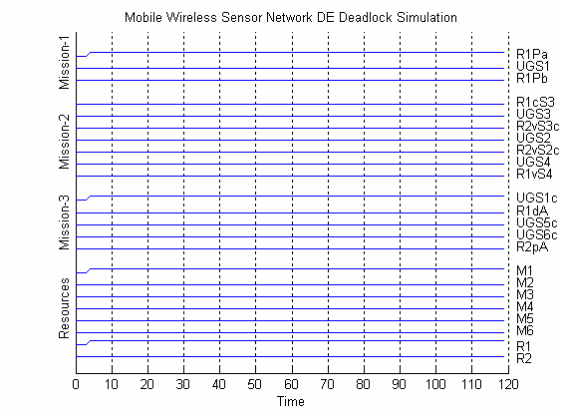


(e)

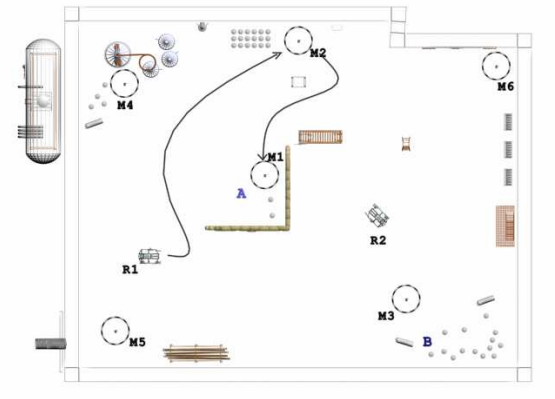


(f)

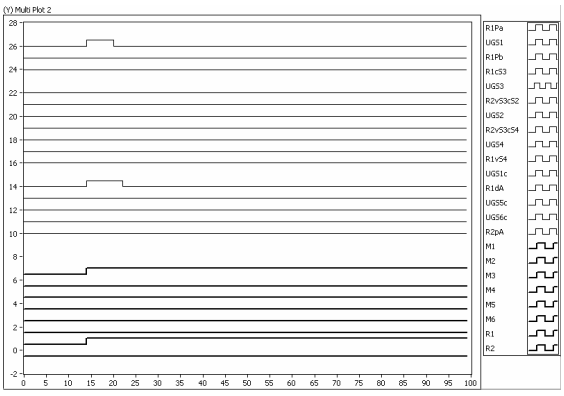
Figure 3.10: Mission 1, 2 with deadlock avoidance. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.



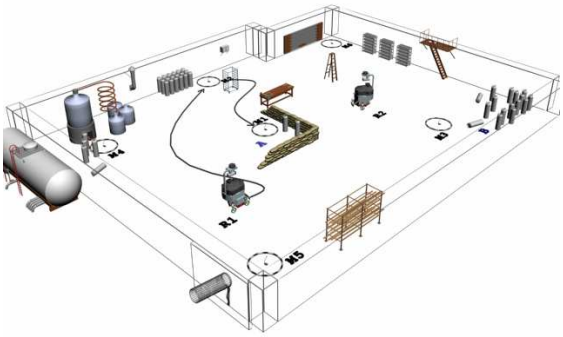
(a)



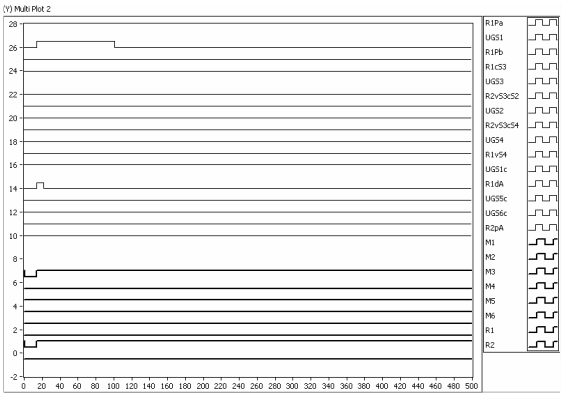
(b)



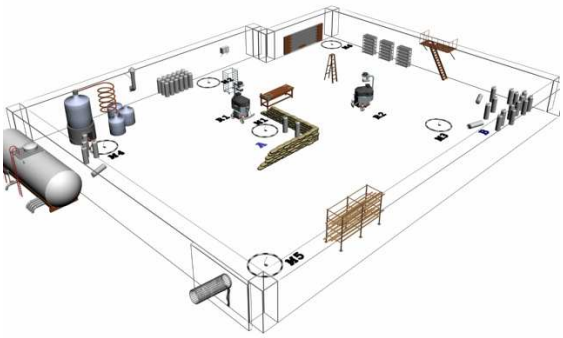
(c)



(d)

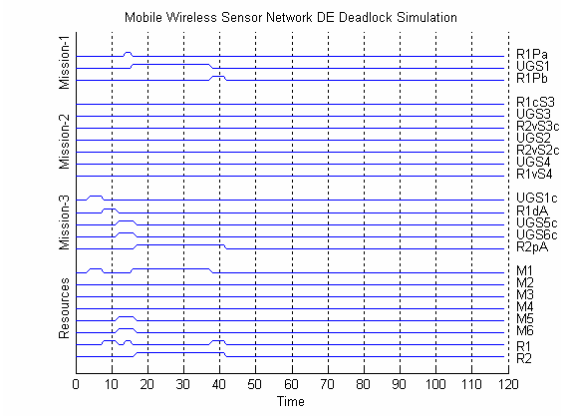


(e)

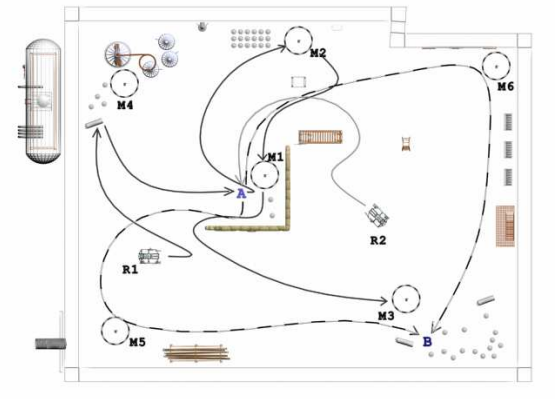


(f)

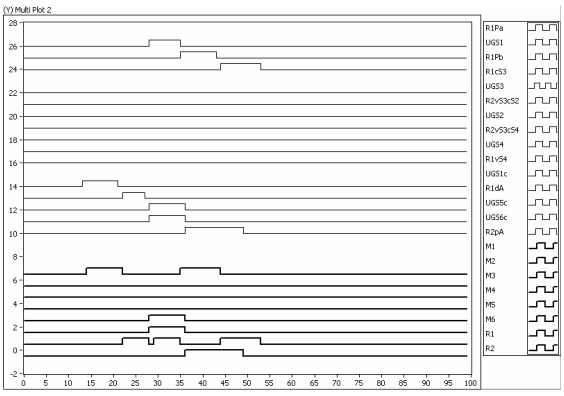
Figure 3.11: Mission 1, 3 with deadlock. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.



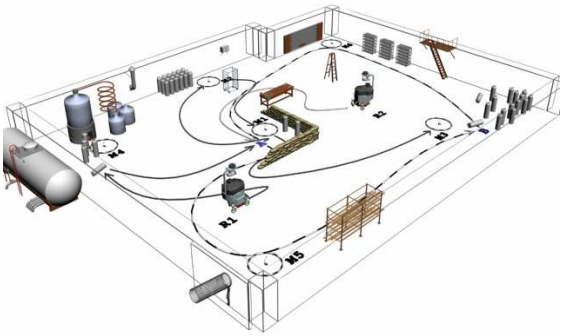
(a)



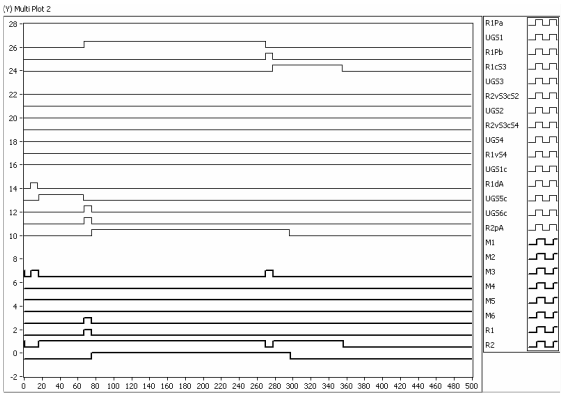
(b)



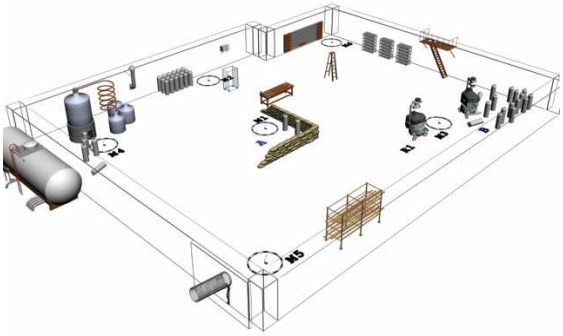
(c)



(d)



(e)



(f)

Figure 3.12: Mission 1, 3 with deadlock avoidance. (a) Matlab simulation, (b) Top view of robot paths, (c) Labview results with simulated resources, (d) Perspective view of robot paths, (e) Labview results with real resources, and (f) Final sensor network topology.

3.3 Routing

Resources in mobile sensor networks are heterogeneous in nature and capable of performing several diverse tasks unlike in manufacturing systems where resources are usually dedicated to a specific task. Thus, for every task awaiting execution, multiple resources are available for being scheduled and the task can request the best (cost / time / energy efficient) resource that can take the task to completion in an optimal manner. Hence we need a formal mathematical model for deadlock-free dynamic resource scheduling where routing as well as dispatching decisions need to be made.

Deadlock analysis in the presence of routing choices is of exponential complexity and deadlock avoidance constraints are rendered computationally intractable [38].

Very little research exists in the field of mobile sensor networks where resources are scheduled dynamically for task execution in a deadlock-free manner. This thesis presents a mathematical formulation for dynamic resource scheduling using the matrix-based discrete event controller. A deadlock avoidance algorithm is developed and simulated for task sequencing in the presence of routing choices.

3.3.1 DEC representation for routing

The matrix-based discrete event controller presented in an earlier section, eqs. (3.1 – 3.4) is flexible and can be easily used to implement the scenario of task sequencing where predetermined resource assignments do not exist and dynamic on-line resource scheduling needs to be performed for allocating a resource to a particular task.

In the presence of shared resources, or in the case of online resource assignment (routing resources), simultaneous activation of conflicting rules arises. The conflict resolution matrix, F_{uc}, in eq. (3.1) is used to resolve conflicts. In earlier work on deadlock avoidance in the presence of shared resources in mobile sensor networks [30], the conflict resolution policy had to handle conflicts of shared resources, i.e., conflicts deriving by the simultaneous activation of rules which start different tasks requiring the same resource. However in the current scenario, with no predetermined resource assignments, conflicts of pseudo-shared resources, and conflicts of routing-resources arise.

Conflicts of pseudo-shared resources arise when simultaneous activation of rules which start the same job using the same resource but having different consequents (such as releasing different resources) occurs. This happens when multiple paths though the sequence join, for instance, logic transitions t_4, t_5 in Figure 3.13 where the resource B_2 acts like a shared resource but starts the same task.

Conflicts of routing resources arise when simultaneous activation of rules which start the same job using different resources (different routes) occurs. This happens when a single route splits, where the same job can be assigned from among a set of resources, for instance, logic transitions t_2, t_3 in Figure 3.13 where resources R_1, R_2 can perform the same job.

A novel augmented conflict resolution matrix is proposed

$$F_{uc} = F_{uc-shared} \,||\, F_{uc-pseudo\_shared} \,||\, F_{uc-routing} \tag{3.10}$$

such that element (i, j) is set if completion of conflict arising task v_j is an immediate prerequisite for the activation of logic state x_i. Then setting the element j in the conflict resolution vector u_c determines the inhibition of logic state x_i (rule *i* cannot be fired.) Thus, depending on the way one selects the conflict-resolution strategy to generate u_c, different dispatching strategies can be selected to avoid resource conflicts due to shared resources, pseudo-shared resources and routing resources. Figure 3.14 depicts the construction of the augmented conflict resolution matrix from elements of the resource requirements, and the resource release matrices.

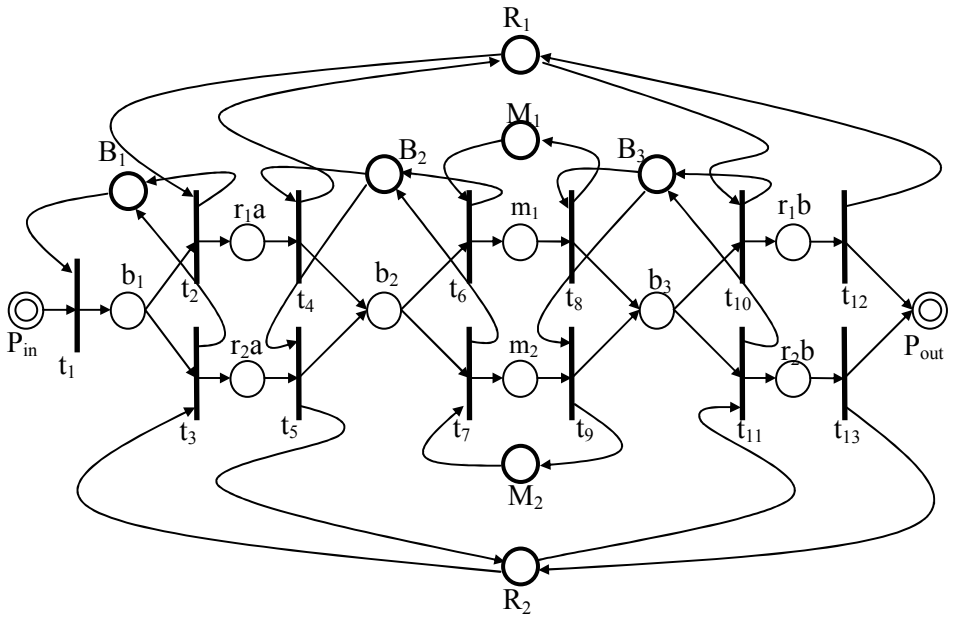
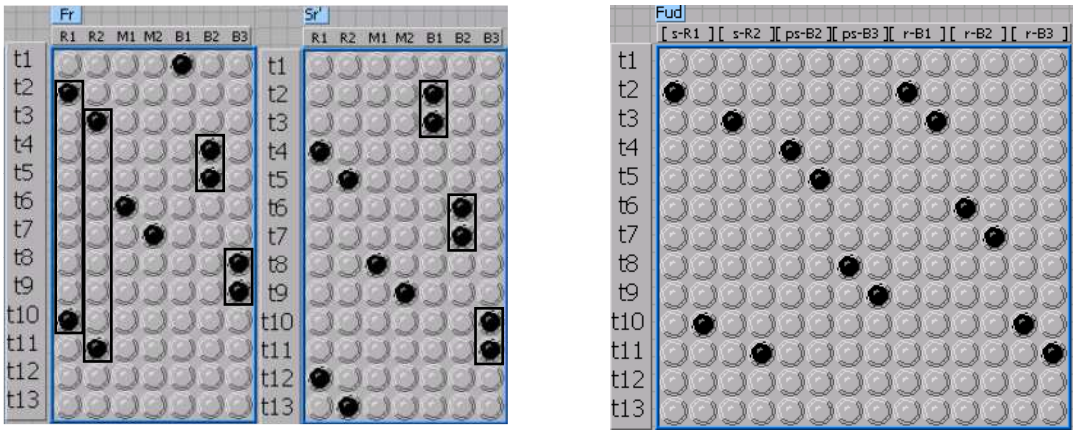


Figure 3.13: Sample Petri net with routing resources.



(a) (b)

Figure 3.14: Augmented conflict resolution matrix formulation. (a) $F_r, S_r{}^T$ matrices for the sample Petri net. Shared, pseudo-shared and routing resources are highlighted, and (b) Augmented conflict resolution matrix.

3.3.2 Deadlock avoidance policy for flexible routing systems

An analysis of deadlock structures using matrices for reentrant flow lines with routing in a flexible manufacturing systems is performed in [40]. However, these mathematical constructions give us a method of detecting active circular waits only. Since, in systems with routing choices, an active circular wait does not always lead to a deadlock. In this section we present mathematical formulations for detecting when an active circular wait could become a potential deadlock and we also present a computationally feasible deadlock avoidance algorithm.

In addition to assumptions for deadlock that were made in section 3.2.1, we have the following non-restrictive capability that

- Some tasks have the option of being executed by a resource from a set of resources (routing resources), and each resource might be used for different tasks (shared resources.)

- Task routings are deterministic and are provided by a dynamic controller.

Mathematical constructions defined in section 3.2.1 are modified as suggested in [40] for adapting the deadlock analysis for the more complex case of routing. Due to the diversity of loop paths that a set of resources contained in a sCW might have [section 3.2.1, equation (3.5)], we need to identify not only the resources that compose the sCW, but also the transitions that link them. This will give us specific information needed to locate siphons and certain critical subsystems needed for construction of the deadlock policy. A general digraph matrix is used

$$W = \begin{bmatrix} 0_r & S_r \\ F_r & 0_t \end{bmatrix} \tag{3.11}$$

where 0_n is a nxn zero matrix, r the number of resources, and t the number of transitions.

Using the general digraph matrix, W with the get both simple circular wait of resources and simple circular wait of transitions

$$C_w = \begin{bmatrix} C_{wr*} & C_{wt*} \end{bmatrix} \tag{3.12}$$

and using the Gurel algorithm, we obtain the the matrix G of composed circular waits to get all the circular wait of resources and transitions

$$\begin{aligned} CW_r &= G^T C_{wr*} \\ CW_t &= G^T C_{wt*} \end{aligned} \tag{3.13}$$

The input and output transitions of a CW are still calculated as eq. (3.6), however the adding and clearing transitions, eq. (3.7) is modified as

$$\begin{aligned} T_p &= {}_dC - \left({}_dC \wedge CW_t\right) \\ T_m &= C_d - \left(C_d \wedge CW_t\right) \end{aligned} \tag{3.14}$$

The task set, siphon-task set, and the critical subsystem of a CW, eq. (3.8) is also modified as

$$\begin{aligned} J_C &= {}_dC.F_v = C_d.S_v{}^T \\ J_s(C) &= J(C) \wedge \left(\overline{C_d.F_v}\right) \\ J_o(C) &= CW_t.F_v \end{aligned} \tag{3.15}$$

With these formulations, equations (3.11 – 3.15), we can detect only active circular waits but not when an active circular wait progresses to a potential deadlock. In a system with routing choices, the system can exist in an active circular wait and still not cause a deadlock. In this thesis, we introduce the concept of exit policies, exit transitions, and exit CW to try to detect when an active circular wait may progress to a potential deadlock.

The exit transition matrix, XT is introduced, which is the set of transitions that when fired would introduce a token into an empty circular wait. Such transitions only exist in systems with routing choices. The exit transition matrix is given as

$$XT = J_0(CW)\bullet - CW_t \tag{3.16}$$

The exit circular wait matrix, CW_X defines the set of circular waits into which an active circular wait could exit to on the firing an exit transition. An exit transition on introducing a token into an active token could cause another circular wait to become empty. The CW_X matrix defines the set of such circular waits for a particular circular wait. The exit circular wait matrix is given as

$$CW_X = \{CW_t : XT \in CW_t\} \tag{3.17}$$

The proposed deadlock avoidance policy allows a circular wait to progress into an active circular wait provided there exists at least one or more exit transitions which when fired would clear the current circular wait without transiting another circular wait into an active circular wait. Hence one or more free exit transitions need to exist, expressed mathematically as

$$n(CW_{active}) \in CW_X > 0 \tag{3.18}$$

Thus our deadlock avoidance policy allows an empty circular wait to form provided a free exit transition exists. This is a computationally feasible solution for deadlock-free dispatching in the presence of routing choices. Figure 3.15 illustrates the various constructs discussed in this section for the four simple circular waits of the sample Petri net in Figure 3.13.

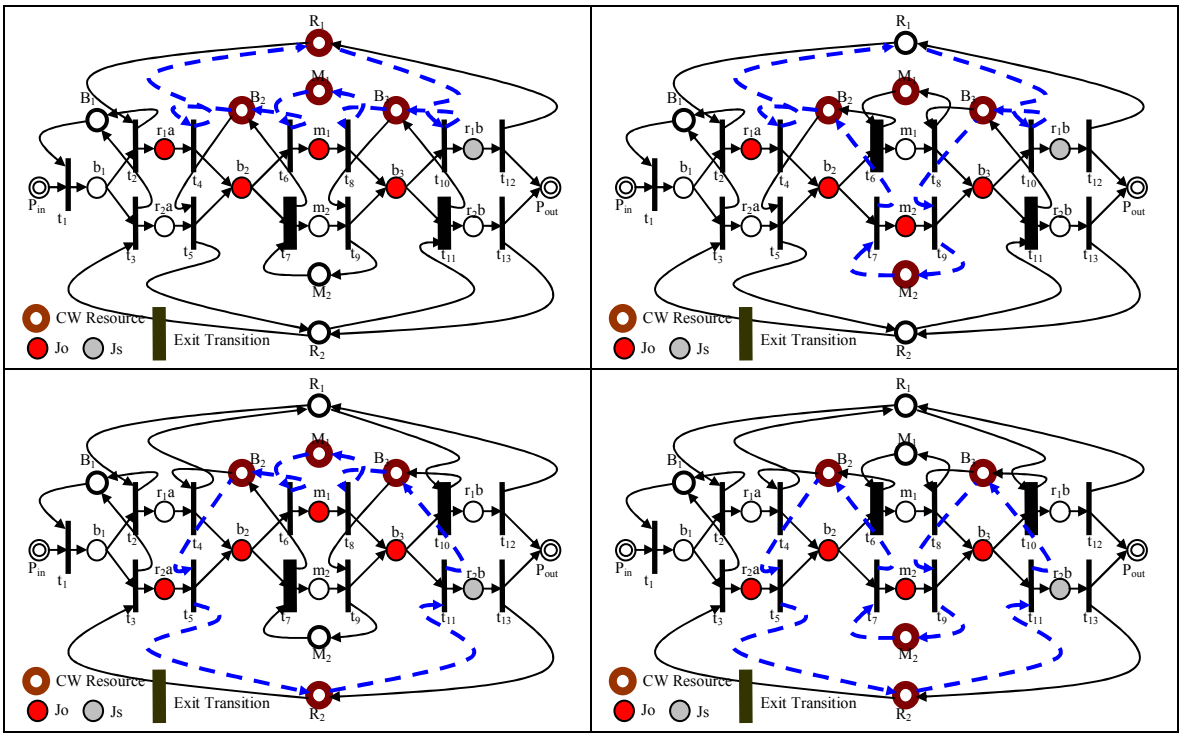


Figure 3.15: Four simple CWs and their corresponding CW_r, CW_t, J_s, J_o, XT .

3.3.3 Simulation results

The sample Petri net considered in Figure 3.13 is simulated. This has multiple routing choices to be made and the proposed augmented F_{ud} matrix ensures that the conflicts that arise due to shared, pseudo-shared, and routing resources are resolved. Initial simulations for deadlock consist of disabling all routing choices and triggering the mission multiple times (ten times), this is the case of simple deadlock and the discrete event transition traces is as seen in Figure 3.16. On enabling both routing and deadlock avoidance, and triggering the mission multiple times (ten times) to cause multiple complex deadlocks, we get the event trace as seen in Figure 3.17. This clearly illustrates that the mission is taken to completion smoothly without any deadlocks multiple times. Comparing Figure 3.16 and Figure 3.17, we see, as expected that with

multiple choices of routing for a particular task, the overall throughput of the entire mission is greatly improved.

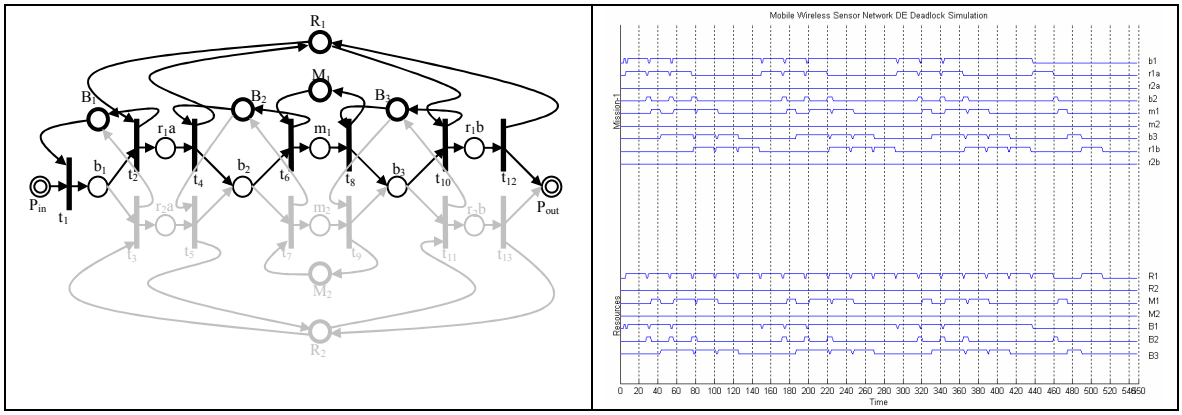


Figure 3.16: Deadlock avoidance simulation with all routing disabled.

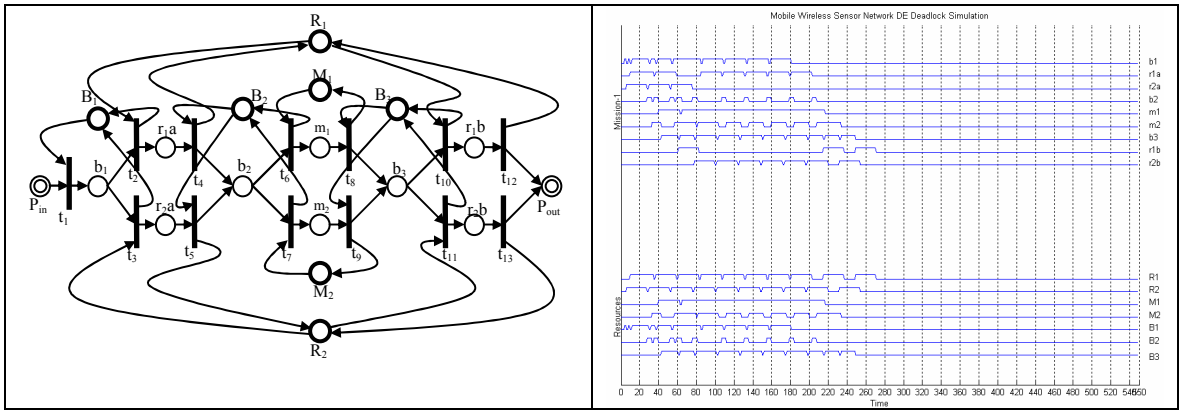


Figure 3.17: Deadlock avoidance in the presence of routing choices.

3.4 Summary

This chapter has extended the preliminary analysis of deadlock avoidance polices for shared resources in heterogeneous mobile sensor networks to more complicated scenarios. We have show through experimental implementation on an actual mobile sensor network test-bed, the feasibility and effectiveness of the proposed deadlock-free supervisory control in performing complex and simultaneous sequencing of interconnected tasks. Further, a general deadlock avoidance policy for system with

flexible routing, where both shared and routing resources are present, has been mathematically formulated and various simulations performed to validate deadlock-free operation in the presence of multiple routing choices.

CHAPTER 4

LOCALIZATION

Location information is imperative for applications in both wireless sensor networks and mobile robotics. Many sensor network applications, such as tracking targets, environmental monitoring, geo-spatial packet routing, require that the sensor nodes know their locations. The large scale of deployment in sensor networks makes careful placement or uniform distribution of sensor nodes impractical. Here we propose a localization algorithm for simultaneous localization of the sensor network and the mobile robot using simple geometric constraints of radio connectivity.

The chapter is organized into the following sections. Section 4.1 presents an algorithm for localization of static sensor nodes using positional updates broadcast from the mobile robot. Section 4.2 presents an algorithm that updates the location information of the mobile robot based on GPS measurements, when they occur, and position information from nodes that are well localized. We illustrate the simultaneous localization of both static sensors and the mobile robot by fusing information from multiple sources. Section 4.3 addresses the problem of where to send the mobile robot next to maximally decrease the localization uncertainty in the sensor network. This is the scenario of Adaptive Localization. Section 4.4 concludes the chapter.

4.1 Sensor Localization using Mobile Robot

In this section we provide an algorithm that runs on each Unattended Ground Sensor (UGS) node that allows it to update its position estimate, and the uncertainty in that estimate, as a mobile robot with known position moves through the network. The algorithm is range-free in that only the communication range need be known, not the range from the node to the mobile robot. It is assumed in this section that the mobile robot's position is exactly known.

4.1.1 Scenario

A deployed wireless sensor network comprised of static unattended ground sensors is to be absolutely localized by a mobile robot. The robot broadcasts consist of its own position and its position uncertainty estimates. Broadcasts can only be heard within the robot's communication range. The static sensors, on receiving these broadcasts, combine the new information to update their current location estimate. A simple discrete-time Kalman filter running on each static sensor node serves to fuse information and update its location and uncertainty estimates.

This is a formalized rigorous approach employing Kalman filters for localization, in contrast to bounding boxes [52, 53], which are harder to update and keep track of. The developed algorithm is simple and can efficiently be implemented on the sensor nodes with a small computing power. The Kalman filter is simply an optimal recursive data processing algorithm [64] and has been subject of extensive research and applications, particularly in the area of autonomous navigation.

4.1.2 Robot Control

A classical three-wheeled tricycle robot model is employed in all simulations. This configuration uses a controlled steering angle and drive speed to navigate to a desired position as illustrated in Figure 4.1.

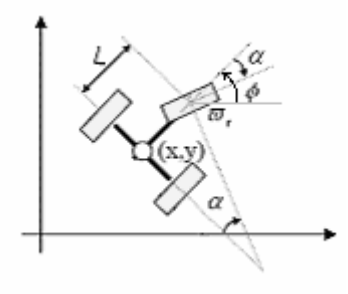


Figure 4.1: Tricycle Robot Configuration.

The states and kinematics of the robot are given by,

$$X = \begin{bmatrix} x & y & \phi & \alpha \end{bmatrix}^T \tag{4.1}$$

$$\dot{X} = a(x,t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} v_t \cos\alpha \cos\phi \\ v_t \cos\alpha \sin\phi \\ v_t / L \sin\alpha \\ \omega_\alpha \end{bmatrix} \tag{4.2}$$

with (x, y) the position of the robot, α the steering angle, and ϕ the heading angle. The control inputs are the speed v_t and the steering rate ω_α.

A simple Proportional-Derivative goal-based controller with a temporally varying goal is implemented to navigate the robot along a desired trajectory. For more details, see [65].

This dynamical setup allows more accurate simulations than the simple moving-point model usually assumed in sensor network localization papers.

4.1.3 Sensor Node Kalman Filter

Each static sensor node maintains its own position and uncertainty estimates. The mobile robot broadcasts contain the robot's position estimate and uncertainty estimate. The broadcasts can only be heard within the robot's communication range. A discrete-time Kalman filter running on each sensor node combines this information to optimally update the node's position estimate and its uncertainty. For more details on the derivation of the Kalman filter equations, interested readers are referred to [58].

The Kalman filter is a set of mathematical equations running in a software algorithm that provide an efficient computational means to estimate the state of a process. The state of sensor i at discrete time instant k is

$$x_k^i = \begin{bmatrix} x^i & y^i \end{bmatrix}^T \tag{4.3}$$

The sensor state is governed by the linear stochastic difference equation

$$x_{k+1}^i = A_k^i x_k^i + B_k^i u_k^i + G_k^i w_k^i \tag{4.4}$$

with measurements given by

$$z_k^i = H_k^i x_k^i + v_k^i \tag{4.5}$$

The random variables w_k^i and v_k^i represent process and measurement noises given by

$$x_0^i = \left(\bar{x}_0^i, P_{x_0}^i\right), w_k^i = \left(0, Q_k^i\right), v_k^i = \left(0, R_k^i\right) \tag{4.6}$$

where (m, P) denotes a Gaussian noise process with mean m and covariance P.

For stationary nodes, the system matrices are given by

$$A_k^i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B_k^i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, G_k^i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, H_k^i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.7}$$

The *a priori* position estimates prior to measurement updates at time $k+1$ are given by the time update equations, which give the effects of time on sensor localization:

$$P_{k+1}^{i\ -} = P_k^i + Q_k^i \tag{4.8}$$

$$\hat{x}_{k+1}^{i\ -} = \hat{x}_k^i \tag{4.9}$$

In these equations, $\hat{x}_k^i$ represents the position estimate of node i at time k, while the covariance matrix P_k^i gives the corresponding uncertainty in the position estimate.

The *a posteriori* estimates given a position measurement z_k are given by the measurement update equations, which gives the effect of the robot broadcast on sensor localization:

$$P_{k+1}^i = \left[P_{k+1}^{i\ -1} + H_{k+1}^{i\ T} R_{k+1}^{-1} H_{k+1}^i \right]^{-1} \tag{4.10}$$

$$\hat{x}_{k+1}^i = \hat{x}_{k+1}^{i\ -} + P_{k+1}^i H_{k+1}^{i\ T} R_{k+1}^{-1} \left(z_{k+1}^i - H_{k+1}^i \hat{x}_{k+1}^{i\ -} \right) \tag{4.11}$$

The covariance matrices Q_k^i and R_k are design parameters chosen by the engineer. With a zero Q_k^i, the uncertainty in location of the sensor i remains constant with time. With an extremely small Q_k^i, the localization uncertainty slowly drifts with time. This means that the current measurements from the mobile robot are given more weight than the current node position estimate, which avoids the node's becoming too certain of a position that may be incorrect.

When the robot is in range and the sensor hears the broadcast position of the robot, the measurement update equation is used to combine the new information to

improve sensor node position and uncertainty estimates. In this section, the robot is assumed to be perfectly localized. Thus when a sensor hears a broadcast, it could only be within the communication range of the robot whose position is broadcast. The measurement uncertainty matrix R_k reflects this, and is chosen as

$$R_k = \sigma^{Bot}, \sigma^{Bot} = \begin{bmatrix} \sigma_x^{Bot} & 0 \\ 0 & \sigma_y^{Bot} \end{bmatrix} \tag{4.12}$$

$$\sigma_x^{Bot} = \frac{Range_x^{Bot}}{\sigma_{const}}, \sigma_y^{Bot} = \frac{Range_y^{Bot}}{\sigma_{const}} \tag{4.13}$$

where σ^{Bot} is the uncertainty introduced due to $Range^{Bot}$, the communication range of the robot. We assume the design parameter $\sigma_{const} = 3$, to include 70% of the communication range, $Range^{Bot}$, of the robot (Gaussian uncertainties are assumed.) Through this selection of R_k the Kalman filter automatically takes care of the range of the robot within which it hears broadcasts.

Algorithm in Table 4.1 shows the position update algorithm that runs on each node, which is very simple and easy to implement. It consists of four equations, two for the time update, and two for the measurement update. This algorithm automatically provides uncertainty estimates through the computation of the error covariance P_k^i, which is equivalent to the bounding box information provided by the algorithm in [52].

Table 4.1: Static sensor node localization algorithm

```
1. At each discrete time instant
2. if robot broadcast received by sensor
3. then
4.   Update sensor state and uncertainty estimates using KF
   measurement Eqs. (4.10), (4.11).
5. else
6.   Propagate estimates using time update Eqs. (4.8), (4.9).
7. end if
```

4.1.4 Simulation Results

Extensive simulations have been performed to verify the effectiveness of the proposed algorithm. We also studied the effects of initial sweep paths and the robot broadcast interval on sensor localization. The mobile robot is navigated along the desired sweep path and periodic location information is broadcast. On receiving a broadcast, sensors update their location and uncertainty estimates. This is a range-free procedure that relies on the limited communication range of the robot, and as such, the sensor locations are updated based on the position of the robot. That is, the updated sensor position estimate is a weighted combination of its current location estimate and the current location of the robot. Thus sensors hearing only one broadcast will have an estimated location that is projected onto the path of the robot.

Figure 4.2 shows the initial sinusoidal sweep path and the position and range of the broadcast with a broadcast interval of 5 discrete time instants. The 'x' represent the actual positions of the static sensors that are to be localized. The sensor nodes do not initially know their actual positions. The nodes all have initial position estimates being the centroid of the deployment area, and an initial uncertainty of infinity, corresponding to complete lack of knowledge of their positions.

Figure 4.3 illustrates the localized sensors after the robot has made its sweep through the network. The '$\bullet$' represent the final position estimates of the nodes. To remain consistent with earlier work involving bounding boxes (e.g. [52]), the uncertainty of the sensors in their position estimates has been depicted as rectangles representing 3σ of the uncertainty distribution, assuming Gaussian uncertainties. Note

that the sensors always outside the communication range of the mobile robot do not become localized (i.e. they have no bounding box, which denotes infinite position uncertainty). The sensors that receive more than one broadcast from the mobile robot end up being better localized, since each position update reduces the position uncertainty.

The effectiveness of the algorithm is demonstrated by the fact that in every case, the actual location (marked by an 'x') is within the uncertainty bound of the estimated position (marked by a '$\bullet$').

The localization error of the sensors, computed as the Euclidean distance between true and estimated positions, is depicted in the vertical axis of Figure 4.4. Sensors near the path of the mobile robot that have received multiple broadcasts have smaller errors.

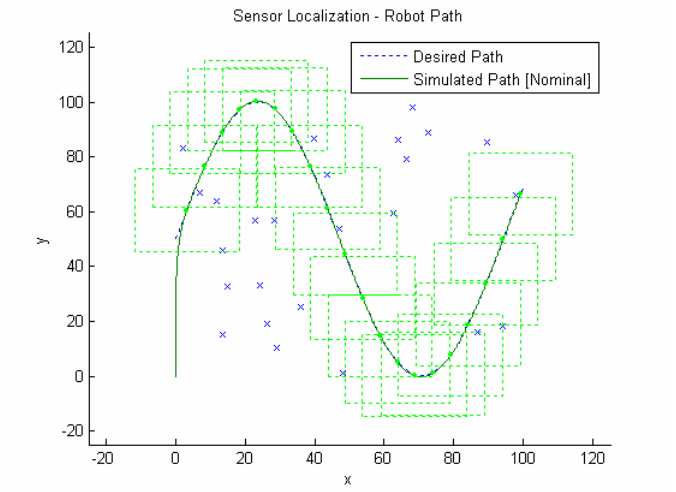


Figure 4.2: Initial sinusoidal sweep path with broadcast locations and range of broadcast.

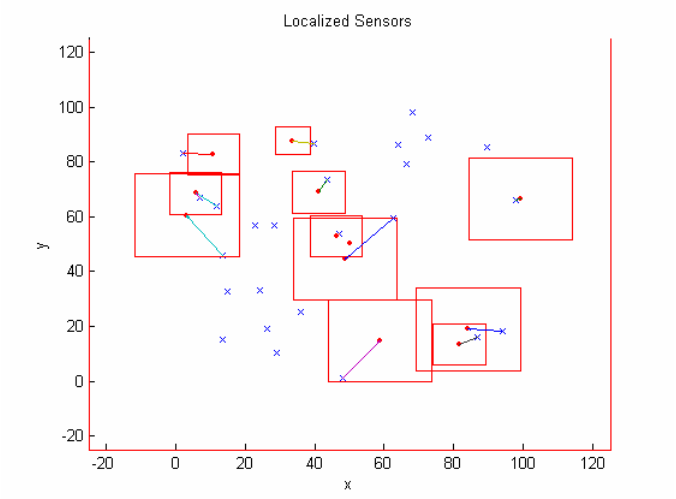


Figure 4.3: Localized sensors, real positions (denoted by 'x') and estimated positions (denoted by '•'), are illustrated after initial mobile robot sweep of the deployment area. Uncertainty rectangles have been illustrated to depict the uncertainty of the sensor in its position estimate.

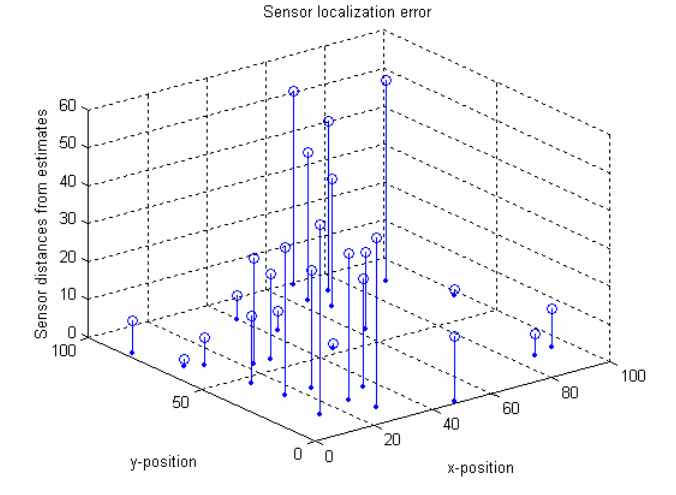


Figure 4.4: Localization error, computed as the Euclidean distance between real and estimated positions, of sensors after initial sweep of the deployment area.

The same simulation was rerun with different mobile robot broadcast intervals, and the effect of broadcast interval on the average localization error of the network is

depicted in Figure 4.5. Generally, as broadcast interval decreases, the average error decreases.

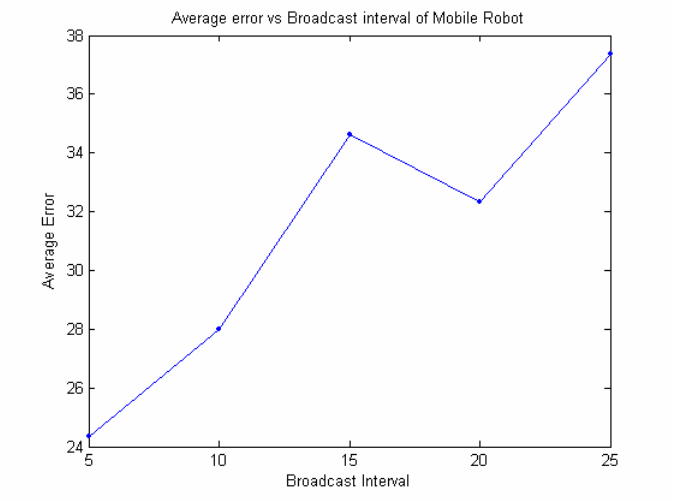


Figure 4.5: Effect of broadcast interval on average localization error.

4.2 Simultaneous Mobile Robot and Sensor Localization

In this section we consider the realistic case where the mobile robot's position is not exactly known. We provide an algorithm which runs on the mobile robot that fuses position information from GPS, when it is available, and from the already-localized sensor nodes. This allows the robot to update its position estimate as well as the uncertainty estimate. When this algorithm is run simultaneously with the algorithm of the previous section running on each sensor node, the result is simultaneous mobile robot and sensor localization. A procedure is given to avoid detrimental recursive feedback between the two algorithms.

4.2.1 Mobile Robot Localization

When localizing the sensor nodes in the previous section, the robot was assumed to know its position exactly at all instants of time. However, as the robot navigates by

dead reckoning, or due to steering inaccuracies, its localization increasingly deteriorates as time passes. Location updates from the GPS, when they occur, and from stationary sensor nodes that have already been localized can be used to improve the localization estimate of the robot.

Some sensor nodes are localized more finely due to more numerous updates they have previously received from the mobile robot. These sensors can be employed to localize the robot when its position information deteriorates. This is accomplished by having each sensor node make a transmission that contains the node's position estimate and uncertainty. This is received by the robot when it is in range. The sensors transmit at fixed intervals, with each sensor having a different random interval. This ensures that the updates between mobile robot and sensor nodes are staggered in time and that no recursive feedback occurs.

A continuous-discrete extended Kalman filter running on the mobile robot is used to simulate the robot and update the states using measurements from the GPS system and the better-localized UGSs. Extended Kalman filters have been used for local and infrequent global senor data fusion [66], for mobile robot localization [9], and in navigation of autonomous vehicles [8]. For information about the Extended Kalman filter see [58].

The continuous-time system model of the robot is given by (4.2) as

$$\dot{X} = a(X, u, t) + G(t)w \tag{4.14}$$

The sampled discrete-time measurement model of the robot is given by

$$\begin{aligned} Z_k^{gps} &= h^{gps}\left[X(t_k),k\right]+v_k^{gps} \\ Z_k^{ugs} &= h^{ugs}\left[X(t_k),k\right]+v_k^{ugs} \end{aligned} \tag{4.15}$$

where

$$X(0)=\left(\overline{X}_0,P_0\right), w(t)=(0,Q), v_k^{gps}=\left(0,R^{gps}\right), v_k^{ugs}=\left(0,R^{ugs}\right) \tag{4.16}$$

$$a(X,t)=\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \dot{\alpha}\end{bmatrix}=\begin{bmatrix}v_t\cos\alpha\cos\phi\\ v_t\cos\alpha\sin\phi\\ v_t/L\sin\alpha\\ \omega_\alpha\end{bmatrix}, G(t)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} \tag{4.17}$$

$$h^{gps}\left[X(t_k),k\right]=\begin{bmatrix}x\\y\end{bmatrix}, h^{ugs}\left[X(t_k),k\right]=\begin{bmatrix}x\\y\end{bmatrix} \tag{4.18}$$

In the extended Kalman filter, the effect of time on the robot states is given by the time update equation

$$\begin{aligned} \dot{\hat{X}} &= a\left(\hat{X},u,t\right) \\ \dot{P} &= A\left(\hat{X},t\right)P+PA^T\left(\hat{X},t\right)+GQG^T \end{aligned} \tag{4.19}$$

In [7], the deleterious effects of time passing are shown in terms of increasing position uncertainty and decreasing belief. These effects are formally captured in a rigorous manner by the time-update equations (4.18), (4.19), which shows how uncertainty increases due to dead reckoning and steering uncertainties.

The effects of the GPS navigation updates, when they are received, are given by the measurement update equation

$$\begin{aligned} K_k &= P^-(t_k)H^{gps^T}\left(\hat{X}_k^-\right)\left[H^{gps}(\hat{X}_k^-)P^-(t_k)H^{gps^T}(\hat{X}_k^-)+R^{gps}\right]^{-1} \\ P(t_k) &= \left[I-K_kH^{gps}\left(\hat{X}_k^-\right)\right]P^-(t_k) \\ \hat{X}_k &= \hat{X}_k^- + K_k\left[Z_k^{gps}-h^{gps}\left(\hat{X}_k^-,k\right)\right] \end{aligned} \tag{4.20}$$

The effects of the updates based on localized sensor nodes, when they are received, are given by the UGS measurement update equation

$$\begin{aligned} K_k &= P^-(t_k)H^{ugs^T}\left(\hat{X}_k^-\right)\left[H^{ugs}(\hat{X}_k^-)P^-(t_k)H^{ugs^T}(\hat{X}_k^-)+R^{ugs}\right]^{-1} \\ P(t_k) &= \left[I-K_kH^{ugs}\left(\hat{X}_k^-\right)\right]P^-(t_k) \\ \hat{X}_k &= \hat{X}_k^- + K_k\left[Z_k^{ugs}-h^{ugs}\left(\hat{X}_k^-,k\right)\right] \end{aligned} \tag{4.21}$$

The measurement uncertainty matrices R^{gps} and R^{ugs} represent the uncertainty in the GPS and the uncertainty in the update offered by UGS i respectively. The uncertainty in the sensor update, R^{ugs}, is a combination of the uncertainty of the sensor position and the uncertainty due to the communication range of the sensor. These uncertainties combine in quadrature as

$$\begin{aligned} R_k^{ugs} &= \left[P^{i\,2}+\sigma^{i\,2}\right], \sigma^i = \begin{bmatrix} \sigma_x^i & 0 \\ 0 & \sigma_y^i \end{bmatrix} \\ \sigma_x^i &= \frac{Range_x^i}{\sigma_{const}}, \sigma_y^i = \frac{Range_y^i}{\sigma_{const}} \end{aligned} \tag{4.22}$$

where σ^i is the uncertainty introduced due to $Range^i$, the communication range of sensor i.

Similarly, the measurement noise covariance of the sensor, eq. (4.12), has to be modified to include the uncertainty in the robot's position. The robot is no longer absolutely localized with zero uncertainty. The uncertainty in robot localization and the uncertainty due to robot communication range combine in quadrature, modifying eq. (4.12) as

$$R_k = \left[P_{XY}^{Bot\,2}+\sigma^{Bot\,2}\right] \tag{4.23}$$

P_{XY}^{Bot} is the partial error covariance of the robot which effects only the position of the robot, and σ^{Bot} is as defined earlier.

The Jacobians of the nonlinear system, determined from (4.2), are given by the following system matrices:

$$
\begin{aligned}
A(X,t) &= \frac{\partial a(X,t)}{\partial X} = \begin{bmatrix} 0 & 0 & -v_t \cos\alpha \sin\phi & -v_t \sin\alpha \cos\phi \\ 0 & 0 & v_t \cos\alpha \cos\phi & -v_t \sin\alpha \sin\phi \\ 0 & 0 & 0 & v_t / L \cos\alpha \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
H^{gps}(X) &= \frac{\partial h^{gps}(X,k)}{\partial X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\
H^{ugs}(X) &= \frac{\partial h^{ugs}(X,k)}{\partial X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{aligned}
\tag{4.24}
$$

With these equations in place and programmed as a software algorithm on the mobile robot, and the sensor nodes running the algorithm presented in the previous section, the mobile robot and the static sensors automatically mutually update their estimates with incoming updates. There is no additional decision-making logic to be implemented as in other range-free work discussed earlier. There is no need to compute bounding boxes, as the error covariance matrices are automatically updated as measurements are received.

The algorithm to be implemented on the mobile robot that updates its position estimate and uncertainty based on GPS measurements and on the localized sensor nodes is given as algorithm in Table 4.2. This algorithm is efficient to implement since the bulk of it is mathematical equations.

When algorithm in Table 4.2 is run on the robot simultaneously along with algorithm in Table 4.1 on each sensor node, simultaneous mobile robot and sensor localization occurs.

Table 4.2: Mobile robot localization algorithm.

```
1. Navigate robot along desired path.
2. Broadcast location information at discrete intervals.
3. if broadcast from GPS received
4.   Update robot state and uncertainty estimates using measurement
   eq. (4.20).
5. end if
6. if broadcast from sensor received
7. Update robot state and uncertainty estimates using measurement
   eq.(4.21).
8. end if
```

4.2.2 Simulation Results

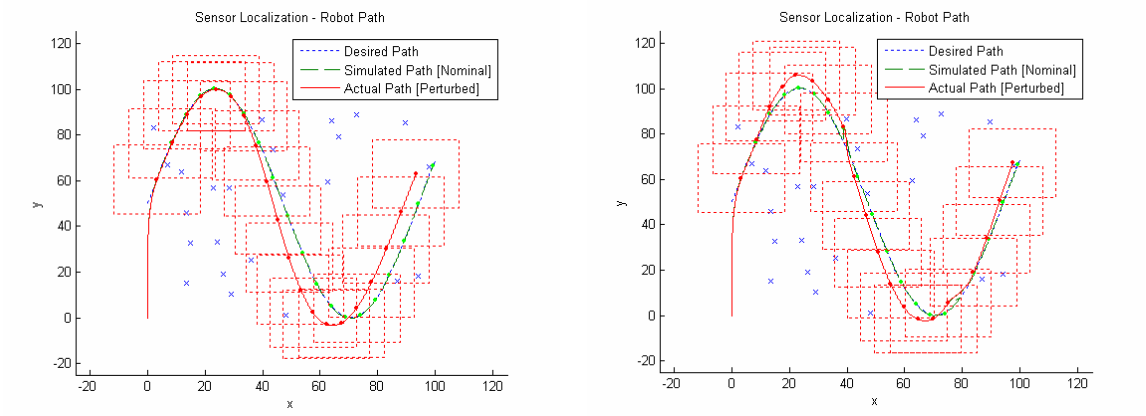
The simulations described in the previous section have been rerun with GPS updates and sensor updates implemented as algorithm in Table 4.2 on the mobile robot. Infrequent GPS updates and temporally staggered sensor updates help localize the robot. Figure 4.6(a) shows the robot's sweep path with GPS and UGS updates disabled. A systematic dead reckoning error [59] has been injected into the mobile robot to give gradually deteriorating position information. The localization of the robot deteriorates with time as can be seen in the deviation of the robot's estimated path (hyphenated green line) from the robot's true path (continuous green line.)

Figure 4.6(b) illustrates the robot's sweep path which is corrected in time by GPS and UGS updates using algorithm in Table 4.2. As is evident, the robot's localization has improved and the positions of where the robot thinks it is (the estimated position), and where the robot actually is (the true position) are much closer, since the

estimates are continuously corrected using algorithm in Table 4.2 as position information arrives, either from GPS or from sensor node broadcasts.

Robot broadcasts occur along the true path of the robot and consist of the robot's estimated position (slightly different from the robot's true position where the broadcast occurs) and uncertainty. Sensors within range receive the broadcast and update their positional information based on the robot's estimates.

Figure 4.7 illustrates the localized sensors after the initial sweep. True sensor positions are indicated by an 'x' and estimated positions by a '•'. Now, some true sensor positions are outside the 3σ boxes due to the added uncertainty in the robot position, though they are generally close to these boxes. Figure 4.8 depicts the final localization error of each sensor.



(a) (b)

Figure 4.6: Initial sweep path of the mobile robot with (a) GPS and UGS updates disabled, and (b) GPS and UGS updates enabled.

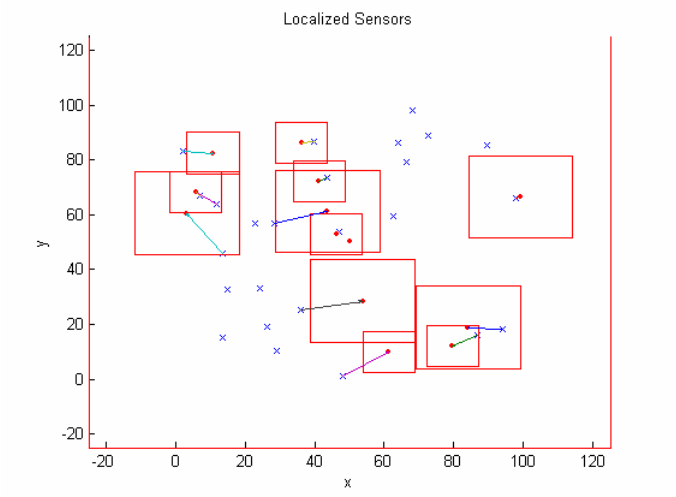


Figure 4.7: Localized sensors after initial sweep of the deployment area.

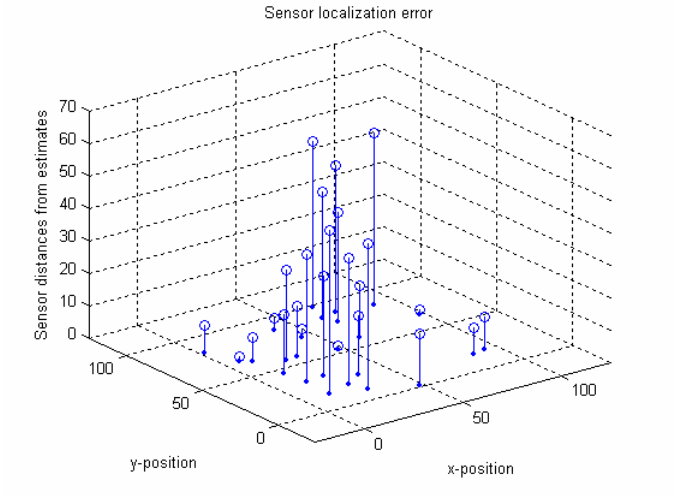


Figure 4.8: Localization error of sensors computed as the distance between true and estimated positions.

4.3 Adaptive Localization

A navigation strategy, to be used subsequent to the initial sweep of the deployment area that was presented in the previous sections, is developed here which further minimizes the localization uncertainty of the sensor network in the most efficient manner. An adaptive localization policy is adopted to navigate the mobile

robot to an area of least localized sensor nodes. This ensures that the robot maneuvers to an area with sensor nodes possessing the largest uncertainty in location.

Accurate position of coarsely localized sensors can not be known (due to inherent coarse localization) so that navigating to these sensors is not possible. The radio connectivity of the network is exploited to address the problem of having the robot navigate to a location which is imprecise. Figure 4.9 depicts the communication connectivity of the network.

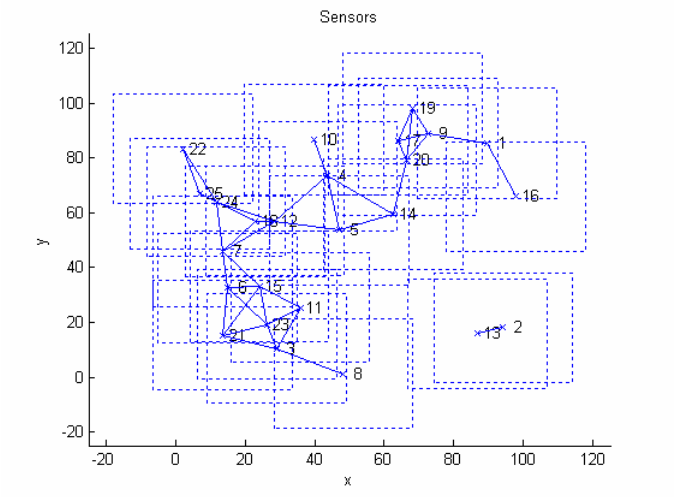
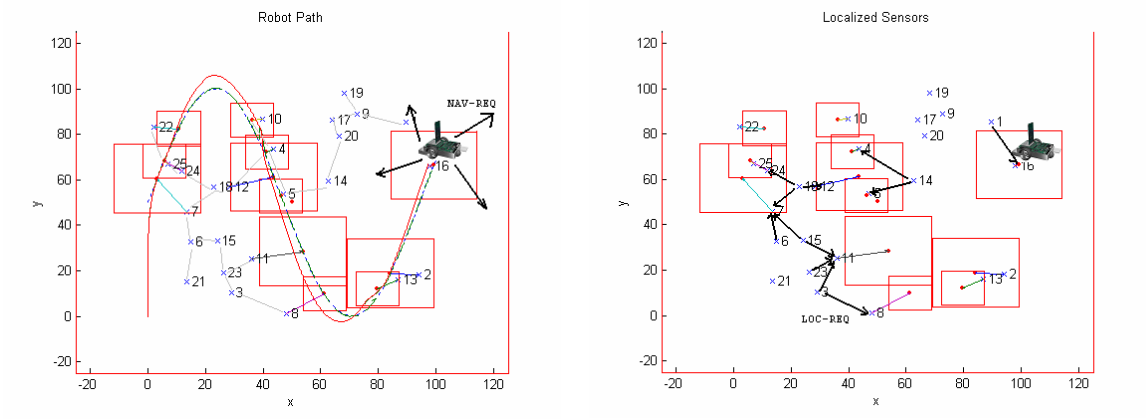


Figure 4.9: Communication connectivity of the network. Communication routes between sensors and range of communication of each sensor are depicted.

A communication protocol is developed wherein, the robot broadcasts a navigation request packet, *NAV-REQ*, when the robot wants to find a new location to navigate to. Sensors which receive the *NAV-REQ* packet, forward it along the network. Sensors having a large uncertainty scalar, the Frobenius Norm [67] of the uncertainty matrix, reply back with a localization request packet, *LOC-REQ*. The *LOC-REQ* packet consists of the uncertainty matrix of the requesting sensor and propagates along the

network until it is received by a friendly localized neighbor. Friendly localized neighboring sensors receiving the *LOC-REQ* packet append it with their position and forward the packet along the sensor network to the robot. Figure 4.10(a) and Figure 4.10(b) show the flow of the *NAV-REQ* and *LOC-REQ* packets.



(a) (b)

Figure 4.10: Flow of the (a) NAV-REQ, navigation request and (b) LOC-REQ, localization request packets through the sensor network.

The robot receives packets from multiple non-unique friendly neighbors each representing a single coarsely localized sensor. The robot needs to choose a friendly neighbor to navigate to. Friendly neighbor arbitration is performed by grouping uncertainties of the same friendly neighbor in quadrature to give its combined uncertainty scalar. The friendly neighbor with the largest combined uncertainty scalar is picked as the location to navigate to. If multiple such neighbors exist, the most localized neighbor is chosen.

Thus regions with a large density of coarsely localized sensors having a common friendly neighbor are adaptively navigated to. However, due to the inherent

imprecise location of the friendly neighbor, the robot actually navigates a circular path around the neighbor.

Algorithm in Table 4.3 summarizes the Adaptive localization algorithm.

Table 4.3: Adaptive localization algorithm.

```
1.  Broadcast Navigation request, NAV-REQ, packet.
2.  Wait to receive Localization request, LOC-REQ, packets.
3.  for all LOC-REQ with the same friendly neighbor
4.    Combine uncertainty scalars of the requesting sensors.
5.  end for
6.  Pick friendly neighbor with maximum combined uncertainty scalar
    of the requesting sensors.
7.  if multiple maximas arise
8.    Among the maxima, pick the most localized friendly neighbor.
9.  end if
10. Navigate around the picked friendly neighbor executing the
    simultaneous localization algorithm, algorithm in Table 4.1 on
    the senors and algorithm in Table 4.2 on the mobile robot.
11. Repeat Steps 1-10 as required.
```

After the initial sinusoidal sweep, see Figure 4.7, Figure 4.12(b), sensors 7 and 11 both receive three Localization request packets each and on combining the uncertainties of the requesting coarsely localized sensors, an equal maximum uncertainty scalar arises for sensors 7 and 11. However sensor 11 is more localized than sensor 7 and robot navigation occurs around sensor 11, see Figure 4.11(b).

Figure 4.11 illustrates four adaptive localization iterations and its navigation paths with corresponding uncertainty scalars of the sensors at the end each adaptive localization iteration as illustrated in (a-e). With each adaptive localization iteration, Figure 4.12 shows the reduction of localization error of each sensor, and Figure 4.13 depicts the reduction of the average localization error of the sensor network. Figure 4.14 illustrates the localized sensors after four iterations of the adaptive localization

algorithm. As can be seen, all sensors are localized and uncertainty in localization fairly small.

At every instant, along with the adaptive localization algorithm, algorithm in Table 4.3, the entire simultaneous localization algorithm with updates from the GPS, and more localized sensor, algorithms in Table 4.1 and Table 4.2, are always running. This demonstrates simultaneous adaptive localization of the sensor network.

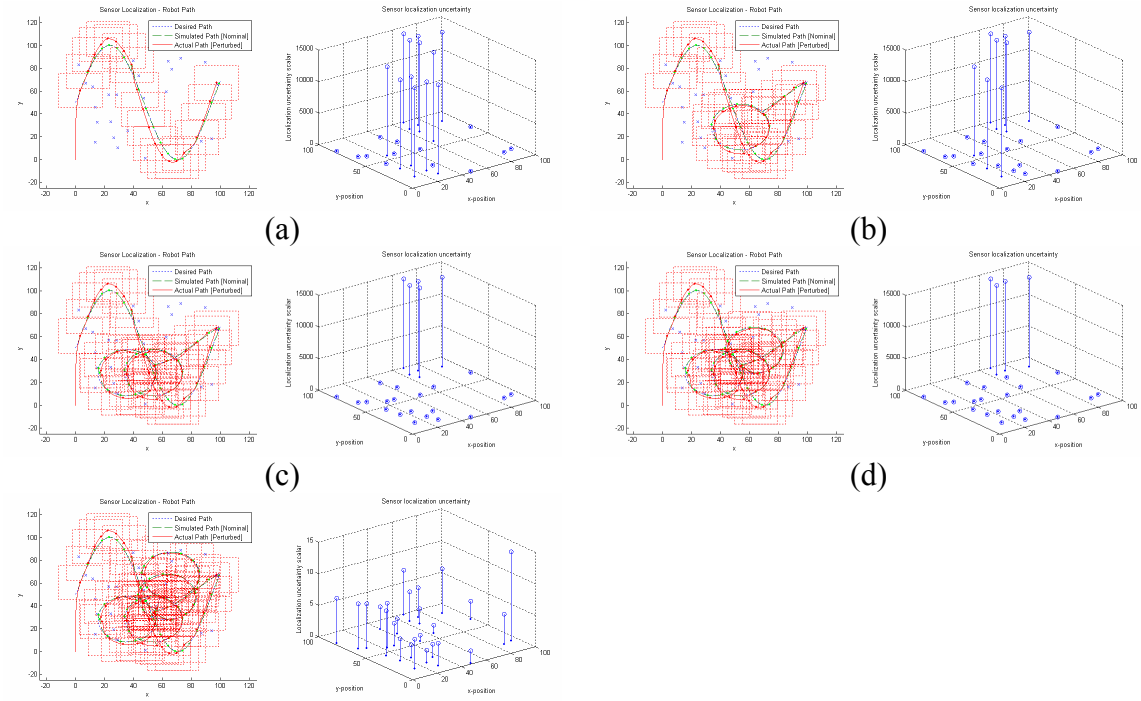
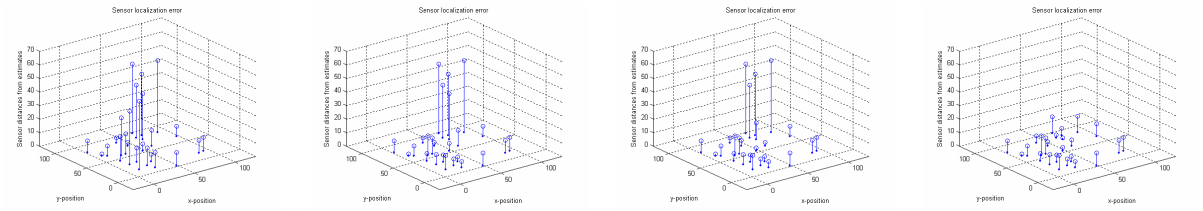


Figure 4.11: Adaptive localization robot paths and corresponding uncertainty scalars for the sensors after (a) Initial sinusoidal sweep, (b) First, (c) Second, (d) Third, and (e) Fourth adaptive navigation steps, respectively.



(a) (b) (c) (d)

Figure 4.12: Reduction of average localization error of sensors with each adaptive localization iteration. (a) Iteration-1, (b) Iteration-2, (c) Iteration-3, and (d) Iteration-4.

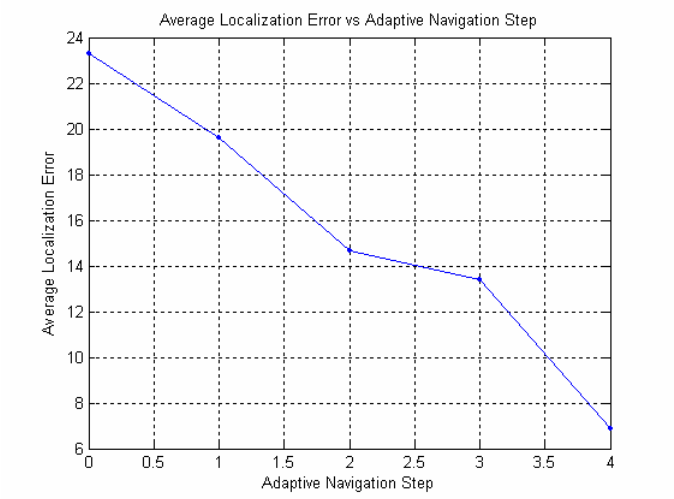


Figure 4.13: Reduction of average localization error with each adaptive localization iteration.

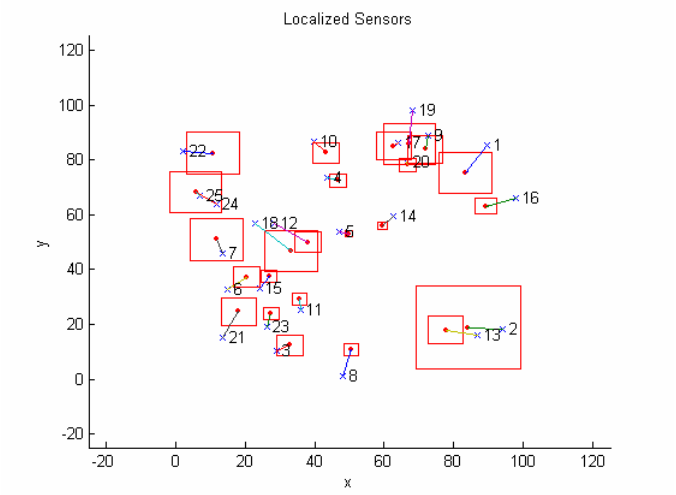


Figure 4.14: The final position estimates of the localized sensors after four iterations of the adaptive localization algorithm.

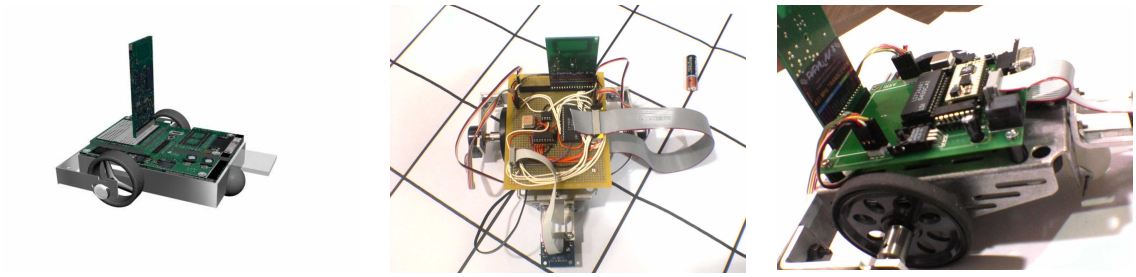
4.4 Summary

Rigorous mathematical algorithms for adaptive simultaneous localization of the static unattended ground sensors and the mobile robot have been demonstrated. The first algorithm localizes the static sensors and the second algorithm localizes the mobile robot. These algorithms together allow simultaneous localization of the static sensor and the mobile robot. A third adaptive localization algorithm ensures that the region of the deployment area with the largest uncertainty is localized with minimal robot movement.

CHAPTER 5

MOBILE ROBOTIC SENSOR

The mobile robotic sensor used for experimental validation of Adaptive Sampling algorithms presented in Chapter 2 has been entirely designed and developed at the Automation & Robotics Research Institute. Figure 5.1 illustrates the evolution of the robot through different stages of design and development. This chapter discusses the design, development and functionality of the mobile robotic sensor.

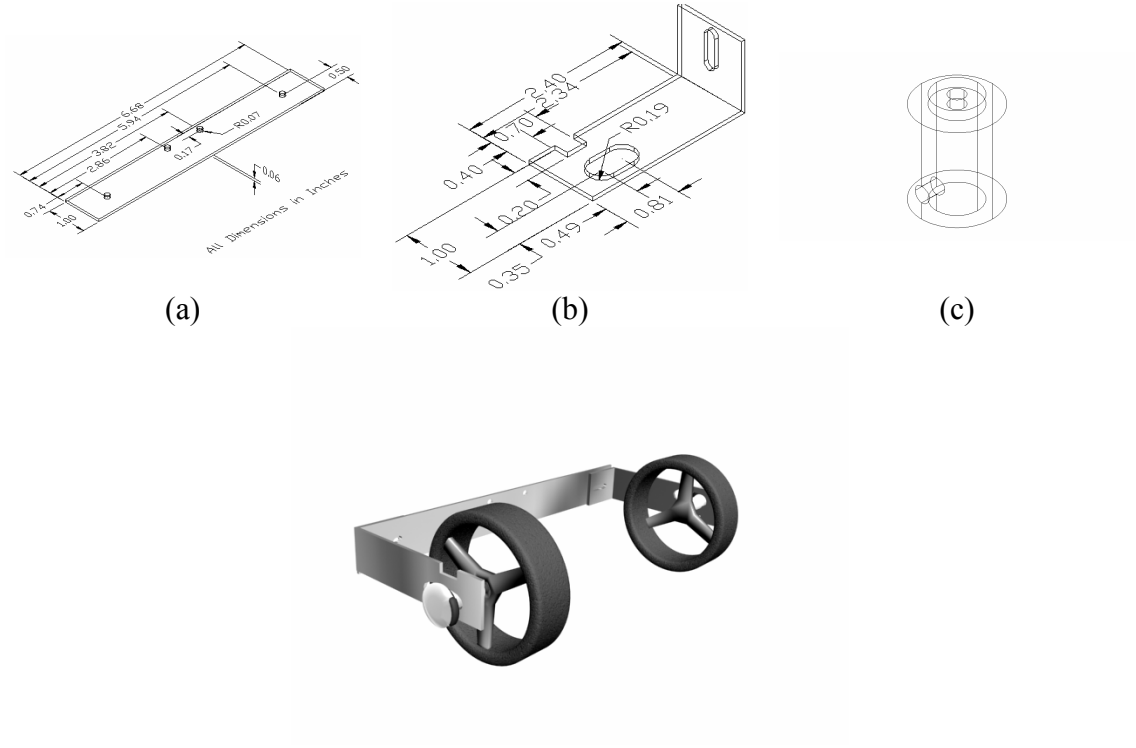


(a) (b) (c)

Figure 5.1: Robot evolution (a) Model, (b) Prototype, and (c) Product.

5.1 Mechanical Design

The mechanical design for machining of custom parts is detailed in this section. The mobile robotic sensor uses a robot chassis from Parallax (Part # 700-00022) as the base and all external components are mounted on to this base. The wheel assembly illustrated in Figure 5.2 has been modeled in AutoCAD and machined to fit on to the chassis.



(a) (b) (c)

(d)

Figure 5.2: Wheel assembly modeled in AutoCAD, and assembled in 3D Studio Max. (a) Base extension, (b) Angle bracket, (c) Shaft coupler, and (d) Wheel assembly.

5.2 Electrical Design

The mobile robotic sensor is a dual microcontroller based system with modules for sensing, radio communication, and motion tracking interfaced onto a common bus. Figure 5.3 shows the design and prototyping of the circuit.

This section details the various electronic components of the mobile robotic sensor.

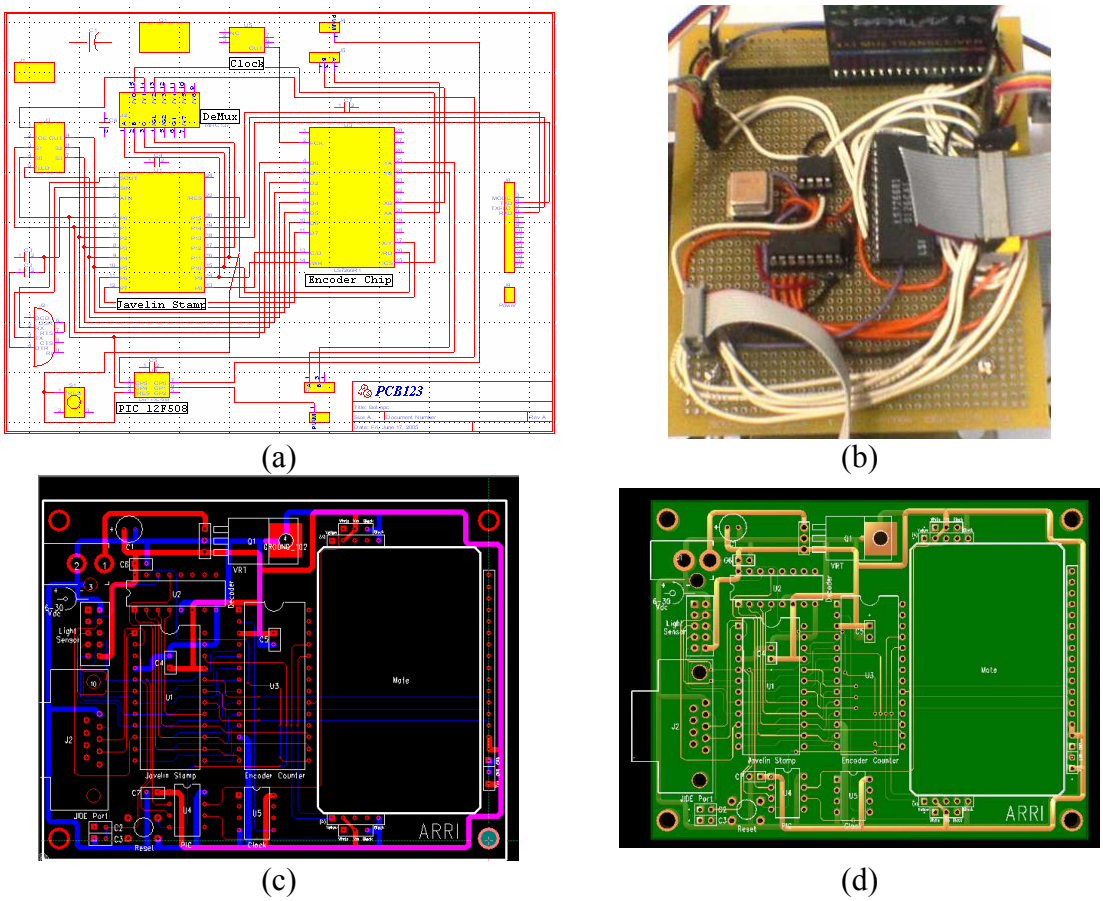


Figure 5.3: Design of the electrical circuit. (a) Schematics, (b) Prototype, (c) PCB layout, and (d) Fabricated PCB.

5.2.1 Microcontrollers

The mobile robot used in this thesis was designed with two separate microcontrollers. A central microcontroller, the Javelinstamp (Part # JS1-IC), deals with all communication and processing for the robot. While a secondary microcontroller, the PIC (Part # 12F508), is solely responsible for the motion of the robot by driving the servos with a pulse width modulated signal. A serial communication protocol for inter-microcontroller communication has been developed for the javelinstamp to send commands to the PIC. By offloading the PWM signal generation to a secondary microcontroller, we guarantee that the central processor is not

bothered for generating periodic pulses and can spend more processing time on other algorithms.

5.2.2 Servos and encoders

The servo motors, Figure 5.4, used on the mobile robots are the continuous rotation servos (#900-00008) from Parallax. These motors operate on 6 Vdc and have an average speed of 60 rpm with no load. The set-point of the servos needs to be calibrated before use to ensure that both left and right wheels revolve at the same speed for the same signal. But achieving this is difficult and we rather command the two wheels with different PWM duty cycle values to ensure that they revolve at the same speed and that the robot moves forward when commanded to do so.

Figure 5.4: Continuous rotation servo.

The encoders used were from Clarostat (Part # 600EN-128-CBL) and produce 128 pulses / revolution. A dedicated 24-bit dual-axis quadrature counter (Part # LS7266R1) has been used dedicated for keeping count of the encoder pulses. A bus architecture connects the encoder counter with the microcontroller for data transmission.

5.2.3 RF Transceiver

The RF transceiver, illustrated in Figure 5.5, is distributed by Parallax Inc. and is manufactured by RF Digital Corp. (Part # 27988). The carrier frequency is 433.92 MHz. The RF transceivers are located on each of the mobile robot platforms and on the base station. The base station is connected to a PC via RS-232 communication link.

Figure 5.5: RF transceiver.

5.2.4 Color Sensor

The robotic sensor is equipped with a color detecting sensor. This facilitates the sampling of a color-coded field. The color sensor is a light to frequency (LTF) sensor modulates the output frequency of a periodic pulse based on the light intensity. The output comprises of the different components of white light - red, green, and blue.

Figure 5.6: TAOS RGB color sensor.

5.3 Software

Extensive coding in multiple languages has been done to achieve a unified functionality of the entire mobile sensor network. In our particular case, the mobile sensor network functions solely to achieve estimation of a linear color field using adaptive sampling. The need for so many different programming environments arises due to the seamless interfacing of multiple native devices which can understand a specific command set only. In our current setup, the following interfaces were programmed.

- Matlab GUI ⇔ Base station.
- Base station ⇔ Multiple mobile robots.
- Mobile robot ⇔ Onboard sensors and devices.

A simple point-n-click command directing a mobile robot to go to a particular location is fairly involved. Firstly, a string command is built up addressing the robot with the "go to" location. (Robot commands are described in section 5.3.1.) Further, this command is serially passed on to the base station where a simplistic network stack is used to package the command into packets to ensure error free wireless transmission with handshaking and acknowledgements. Once the command is received by the wireless module of the robot, a simple parsing algorithm breaks up the command into tokens to be identified. These tokens serve as requests to particular sensors for current readings or for actuators for motion. A low level interfacing module for each specific sensor or device is required here.

The following sections address the various programming requirements at different stages of transmission and processing of a simple command.

5.3.1 Robot commands

Commands to the robot are simple strings terminated with a terminal character such as the semi-colon. These commands achieve motion, sensing, and other miscellaneous system tasks. Motion commands are either open loop where the robot is in motion until a stop command is received, or closed loop where the robot is in motion until the encoders register a motion corresponding to the commanded amount. Some of the commands are described in Table 5.1.

Table 5.1: Robot commands.

Command	***Command description***
F;	Move forward.
B;	Move backward.
R;	Turn right.
L;	Turn left.
S;	Stop.
M 120;	Move forward by 120 encoder counts.
T -85;	Turn clockwise by 85 encoder counts.
T 40;	Turn anti-clockwise by 40 encoder counts.
C;	Take color sample (responds with color read.)

5.3.2 Wireless communication protocol

An error-free communication protocol has been designed such that all communication between the base station and the robot are always acknowledged by each other. A simple checksum inserted computed and inserted into the message ensures that the packet is not corrupted. A packet that is corrupted is not acknowledged by the receiver and this causes the sender to retransmit. Retransmission occurs for a fixed number of times before the sender aborts.

5.3.3 Sensor Interfacing

Low level interfaces for the light sensor and the encoder have been programmed. These involve a physical signal-level commanding of the device for addressing a particular device-specific port or register for obtaining data. The servo actuators are commanded by a dedicated microcontroller (The PIC.) A dedicated microcontroller is required since these actuators need continuous commanding which may take up way too much time by the general purpose microcontroller (the Javelinstamp.) A low-level one signal-line serial interface between the two microcontrollers has been programmed for data communication between the two microcontrollers.

5.3.4 Fixed-point algorithms

Most microcontrollers operate only on signed and unsigned integer numbers. However to implement even the simplest Kalman filter requires mathematical operations on non-integer numbers. The Javelinstamp microcontroller does not support these operations and equipping it with a floating point co-processor would make the system more expensive and also slow down normal operations. A different solution is sought for.

On the current robot system, we implement simple mathematical operations such as addition, subtraction, multiplication and division using a fixed-point binary representation of a non-integer number. This binary representation can easily be stored on a integer register and hardware operations of addition and subtraction suffice for addition and subtraction of non-integer numbers. Separate multiplication and division

algorithms are required however and have been implemented on the robot. Table 5.2 illustrates the binary representation of a non-integer number.

Table 5.2: Fixed-point binary representation of a non-integer number.

integer part							**binary point**	**fractional part**					
...	2^5	2^4	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	...
...	32	16	8	4	2	1	.	$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$	$^1/_{32}$	...

5.3.5 Matlab programming

The adaptive sampling algorithms have been entirely implemented in the Matlab programming environment. A simple graphical user interface developed serves as the interface to the user who can monitor the current topology of the mobile sensor network as mobile robotic sensors navigate adaptively to take samples to estimate the field. The graphical user interface can be use for monitoring the evolution of the estimated field and to compare it with the truth model.

5.4 Experimental Setup

The mobile sensor network system for adaptive sampling is setup as in Figure 5.7. An inexpensive overhead camera serves as the GPS offering infrequent updates of robot poses. The base station and the camera are connected to a PC. All communication with the robots occurs through the base station in a star network topographical manner. Each robot has a unique identifier for communication message arbitration.

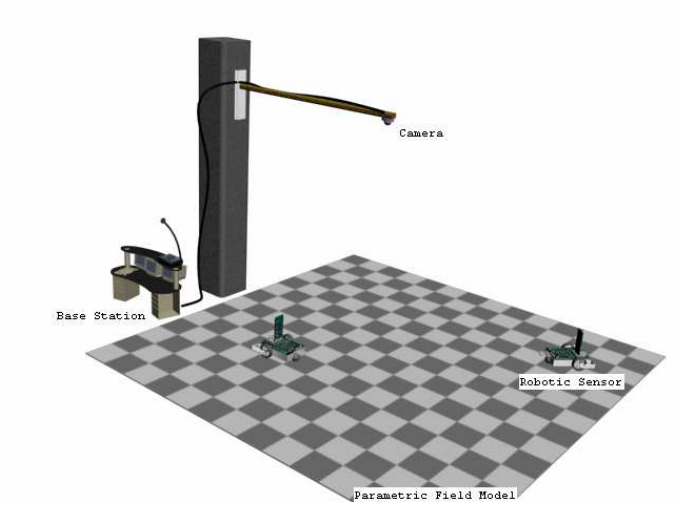


Figure 5.7: Experimental setup.

This simple setup models a real-life scenario where multiple mobile sensors can be deployed in the open, with a global positioning system, when available, localizing the sensors, and the entire sensor network topology changing adaptively to estimate some modeled environmental parameter

5.5 Summary

This chapter has detailed some of the design ideas that went into making the robotic sensor. Various components of the robot and their functionalities have been discussed. The physical setup of the mobile sensor network test bed has been described.

With a physical test bed available, adaptive sampling algorithms developed in simulation can be easily used for experimental feasibility and verification.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This chapter summarizes the contributions of this thesis and presents suggestions for further research in Adaptive sampling, resource scheduling and localization.

6.1 Thesis contributions

This thesis has served in developing Adaptive sampling algorithms for estimating spatially distributed static linear and Gaussian fields which are linear in its parameters. Closed form information measures in linear regression have been used to adaptively estimate linear and Gaussian fields with linear parameters. Nonlinear optimal estimation techniques, such as the Kalman filter, constrained, and unconstrained nonlinear optimizers have been used to adaptively estimate field and field basis parameters. An experimental mobile robotic sensor developed has helped in the validation of the adaptive sampling algorithm by experimentally estimating a linear color field.

This thesis has extended the preliminary analysis of deadlocks using the Discrete Event Controller. A deadlock avoidance algorithm for resource scheduling in the presence of shared heterogeneous resources in mobile sensor networks has been implemented experimentally on the ARRI WSN test bed comprising of Cybermotion

SR2 patrol robots and Berkley motes thereby validating the deadlock avoidance algorithm. Furthermore, a general mathematical formulation has been innovated for deadlock avoidance in systems with flexible-routing.

The thesis also serves in developing a simultaneous and adaptive localization algorithm for the localization of a wireless sensor network using simple geometric constraints of radio connectivity. An adaptive localization algorithm has also been developed for adaptive navigation of a mobile robot such that optimal minimization of the largest uncertainty in the sensor network occurs.

6.2 Future research

Adaptive sampling of complex fields where both the field parameters and the basis parameters of the field need to be estimated still remains unsolved. Further research in nonlinear optimal estimation techniques would serve in approaching this problem.

Several challenges still remain with the mobile robotic sensor where processing power is limited. A Kalman filter implemented directly on the robot would aid in navigation. A logical next step would be to distribute the algorithms to run more locally on the robot itself.

More extensive analysis and simulations of the deadlock avoidance algorithms in the presence of routing choices would help in understanding when such algorithms may fail and deadlock becomes eminent.

Implementing the proposed algorithms in chapter 4 on physical sensors and mobile robots would experimentally validate the proposed algorithms. Future research

should take into consideration, time-varying geometrical constraints of radio connectivity for the sensor network.

BIBILIOGRAPHY

[1] E.M. Petriu, T.E. Whalen, R. Abielmona and A. Stewart, "Robotic sensor agents: a new generation of intelligent agents for complex environment monitoring," *IEEE Instrumentation & Measurement Magazine,* vol. 7, pp. 46-51, September 2004.

[2] F. Zhao, J. Shin and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine,* vol. 19, pp. 61-72, March 2002.

[3] F. Zhao, J. Liu, J. Liu, L. Gibas and J. Reich, "Collaborative signal and information processing: an information-directed approach," *Proceedings of the IEEE,* vol. 91, pp. 1199-1209, August 2003.

[4] D. Fox, "KLD-Sampling: Adaptive particle filters," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic],* T.G. Dietterich, S. Becker and Z. Ghahramani Eds. Cambridge, MA: MIT Press, 2001, pp. 713-720.

[5] T. Arici and Y. Altunbasak, "Adaptive sensing for environment monitoring using wireless sensor networks," in *Wireless Communications and Networking Conference,* 2004, pp. 21-25.

[6] R. Willett, A. Martin and R. Nowak, "Backcasting: Adaptive sampling for sensor networks," in *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks,* 2004, pp. 124-133.

[7] D. Fox, W. Burgard, F. Dellaert and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99),* July, 1999

[8] A. Kelly, "A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles," Carnegie Mellon University., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-94-19, May, 1994.

[9] E. Kiriy and M. Buehler, "Three-state Extended Kalman Filter for Mobile Robot Localization," McGill University., Montreal, Canada, Tech. Rep. TR-CIM 05.06, April, 2002.

[10] C. Savarese, J.M. Rabaey and J. Beutel, "Locationing in distributed ad-hoc wirless sensor networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 2001.

[11] J. Bachrach and C. Taylor, "Localization in sensor networks," in *Handbook of Sensor Networks : Algorithms and Architectures,* 1st ed., vol. 1, I. Stojmenovi Ed. USA: Wiley, 2005,

[12] Dissanayake, Gamini M. W. M., P. Newman, S. Clark and H.F. Durrant-Whyte, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," vol. 17, pp. 229-241, June 2001.

[13] J.W. Fenwick, P.M. Newman and J.J. Leonard, "Cooperative Concurrent Mapping and Localization," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation,* May 2002, pp. 1810-1817.

[14] S. Poduri and G.S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proceedings of IEEE International Conference on Robotics and Automation,* 2004, pp. 165-171.

[15] D.O. Popa, H.E. Stephanou, C. Helm and A.C. Sanderson, "Robotic deployment of sensor networks using potential fields," in *IEEE International Conference on Robotics and Automation,* 2004, pp. 642-647.

[16] C.F. Helm, "Robotic Sensor Deployment using Potential Fields," M.S. Dissertation/Thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, April 2004.

[17] A. Dhariwal, G.S. Sukhatme and Requicha, Aristides A. G., "Bacterium-inspired Robots for Environmental Monitoring," in *IEEE International Conference on Robotics and Automation,* 2004, pp. 1436-1443.

[18] M. Rahimi, R. Pon, W.J. Kaiser, G.S. Sukhatme, D. Estrin and M. Srivastava, "Adaptive sampling for environmental robotics," in *IEEE International Conference on Robotics and Automation,* 2004, pp. 3537-3544.

[19] S.J. Majumdar, C.H. Bishop, B.J. Etherton and Z. Toth, "Adaptive sampling with the ensemble transform Kalman filter. II. Field program implementation," *Mon.Weather Rev.,* vol. 130, pp. 1356-1369, 05// 2002.

[20] A.F. Bennett, *Inverse Modeling of the Ocean and Atmosphere,* Cambridge, UK: Cambridge University Press, 2002.

[21] D.O. Popa, A. Sanderson, V. Hombal, R. Komerska, S. Mupparapu, R. Blidberg and S. Chappel, "Optimal Sampling Using Singular Value Decomposition of the Parameter Variance Space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005,

[22] D.O. Popa, A.C. Sanderson, R.J. Komerska, S.S. Mupparapu, D.R. Blidberg and S.G. Chappel, "Adaptive sampling algorithms for multiple autonomous underwater vehicles," in *Autonomous Underwater Vehicles,* 2004, pp. 108-118.

[23] D.O. Popa, K. Sreenath and F.L. Lewis, "Robotic Deployment for Environmental Sampling Applications," in *International Conference on Control and Automation,* 2005, pp. 197-202.

[24] F.L. Lewis, D. Tacconi, A. Gurel, H.H. Huang and O.C. Pastravanu, "Manufacturing Controller Design and Deadlock Avoidance using a Matrix Model for Discrete Event Systems," in *Methods and Applications of Intelligent Control,* 1st ed., S.G. Tzafestas Ed. Netherlands: Kluwer Academic Publishers, 1997, pp. 485-508.

[25] Mireles, Jose Garcia, Jr., "Matrix-Based Intelligent Discrete Event Control for Flexible Manufacturing Systems," Ph.D. Dissertation/Thesis, The University of Texas at Arlington, Arlington, Texas USA, May 2002.

[26] J. Mireles Jr., F.L. Lewis, A. Gurel and S. Bogdan, "Deadlock Avoidance Algorithms and Implementation, a Matrix Based Approach," in *Deadlock Resolution in Computer-Integrated Systems,* 1st ed., vol. 1, M. Zhou and M.P. Fanti Eds. New York: Marcel Dekker, 2004, pp. 183-232.

[27] T. Cheng, "On-line Deadlock Avoidance for Complex Routing Flexible Manufacturing Cells," *International Journal of Applied Science and Engineering,* vol. 2, pp. 163-176, 2004.

[28] H.J. Yoon and D.Y. Lee, "Deadlock-Free Scheduling of Photolithography Equipment in Semiconductor Fabrication," *IEEE Transactions on Semiconductor Manufacturing,* vol. 17, pp. 42-54, Feb. 2004.

[29] M. Koibuchi, A. Jouraku, K. Watanabe and H. Amano, "Descending layers routing: a deadlock-free deterministic routing using virtual channels in system area networks with irregular topologies," in *Proceedings of 2003 International Conference on Parallel Processing,* 2003, pp. 527-536.

[30] V. Giordano, F.L. Lewis, J. Mireles Jr. and B. Turchiano, "Coordination control policy for mobile sensor networks with shared heterogeneous resources," in *2005 International Conference on Control and Automation,* 2005, pp. 191-196.

[31] J. Ezpeleta, J.M. Colom and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation,* vol. 11, pp. 173-184, April 1995.

[32] Y. Huang, M. Jeng, X. Xie and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *International Journal of Production Research,* vol. 39, pp. 283-305, 2001.

[33] K. Lautenbach, "Linear Algebraic Calculation of Deadlocks and Traps," in *Concurrency and Nets,* 1st ed., vol. 1, K. Voss, H.J. Genrich and G. Rozenberg Eds. Berlin, Germany: Springler-Verlag, 1987, pp. 315-336.

[34] G. Alpan and B. Gaujal, "Supervisory control of Petri nets using routing functions: starvation avoidance issues," *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 30, pp. 684-694, Oct. 2000.

[35] F.L. Lewis, S. Bogdan, A. Gurel and O. Pastravanu, "Analysis of deadlocks and circular waits using a matrix model for discrete event systems," in *Proceedings of the 36th IEEE Conference on Decision and Control,* 1997, pp. 4080-4085.

[36] F.L. Lewis, A. Gurel, S. Bogdan, A. Doganalp and O.C. Pastravanu, "Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems," *Automatica,* vol. 34, pp. 1083-1100, 1998.

[37] J. Mireles Jr., F.L. Lewis and A. Gurel, "Deadlock avoidance for manufacturing multipart re-entrant flow lines using a matrix-based discrete event controller," *International Journal of Production Research,* vol. 40, pp. 3139-3166, 2002.

[38] M. Lawley, "Integrating flexible routing and algebraic deadlock avoidance policies in automated manufacturing systems," *International Journal of Production Research,* vol. 38, pp. 2931-2950, 2000.

[39] M. Lawley, "Flexible Routing and Deadlock Avoidance in Automated Manufacturing Systems," in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation,* 1998, pp. 591-596.

[40] J. Mireles Jr. and F.L. Lewis, "Deadlock Analysis and Routing on Free-Choice Multipart Reentrant Flow Lines Using a Matrix-Based Discrete Event Controller," in *Proceedings of the 41st IEEE Conference on Decision and Control,* 2002, pp. 793-798.

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

[42] B. Hofmann-Wellenhof, H. Lichtenegger and J. Collins, *Global Positioning System : Theory and Practice,* New York: Springer, 2004.

[43] R.L. Moses, D. Krishnamurthy and R.M. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing,* 2002.

[44] N. Bulusu, J. Heidemann and D. Estrin, "Adaptive Beacon Placement," *ICDCS '01: Proceedings of the the 21st International Conference on Distributed Computing Systems,* pp. 489, 2001.

[45] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero III, R.L. Moses and N.S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *Signal Processing Magazine, IEEE,* vol. 22, pp. 54-69, July 2005.

[46] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," in *Proceedings of the IEEE Infocom 2000,* pp. 775-784.

[47] N.B. Priyantha, A. Chakraborty and H. Balakrishnan, "The cricket location-support system," in *Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM),*

[48] A. Savvides, C. Han and M.B. Strivastava, "Dynamic fine-grained localization in Ad-Hoc networks of sensors," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking,* pp. 166-179.

[49] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) using AoA," in *Proceedings of INFOCOM,*

[50] C. Taylor, A. Rahimi, J. Bachrach and H. Shrobe, "Simultaneous Localization and Tracking in an Ad Hoc Sensor Network," *Information Procession in Sensor Networks,* 2005.

[51] A.M. Brooks, S. Williams and A. Makarenko, "Automatic Online Localization of Nodes in an Active Sensor Network," *International Conference on Robotics and Automation,* vol. 5, pp. 4821-4826, 2004.

[52] S. Shenoy and J. Tan, "Simultaneous Localization and Mobile Robot Navigation in a Hybrid Sensor Network," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* August 2005

[53] A. Galstyan, B. Krishnamachari, K. Lerman and S. Pattem, "Distributed online localization in sensor networks using a moving target," *IPSN'04: Proceedings of the Third International Symposium on Information Processing in Sensor Networks,* pp. 61-70, 2004.

[54] V. Cevher and J.H. McClellan, "Sensor array calibration via tracking with the extended Kalman filter," *IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 5, pp. 2817-2820, 2001.

[55] E. Olson, J. Leonard and S. Teller, "Robust Range-Only Beacon Localization," in *IEEE/OES Autonomous Underwater Vehicles,* 2004, pp. 66-75.

[56] W. Xiao, J.K. Wu and L. Xie, "Sensor scheduling for target tracking in networks of active sensors," in *IEEE International Workshop on Sensor Networks and Applications,* 2005,

[57] F.L. Lewis, *Optimal Estimation with an Introduction to Stochastic Control Theory,* New York: John Wiley and Sons, 1986.

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

[59] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on Robotics and Automation,* vol. 12, pp. 869-880, 1996.

[60] A. Divelbiss, "Nonholonomic motion planning in the presence of obstacles," Ph.D. Dissertation/Thesis, Rensselaer Polytechnic Institute, Troy, New York, December 1993.

[61] F.L. Lewis, D.A. Tacconi, O.C. Pastravanu and A. Gurel, "Method and apparatus for testing and controlling a flexible manufacturing system," U.S. Patent 6,185,469, February 6, 2001.

[62] D.A. Tacconi and F.L. Lewis, "A new matrix model for discrete event systems: application to simulation," *IEEE Control Systems Magazine,* vol. 17, pp. 62-71, Oct. 1997.

[63] V. Giordano, F.L. Lewis, B. Turchiano, P. Ballal and V. Yeshala, "Matrix computational framework for discrete event control of wireless sensor networks with some mobile agents," in *Proc. Mediterranean Conference on Control & Automation,* 2005,

[64] P.S. Maybeck, *Stochastic models, estimation, and control,* USA: Academic Press, 1979.

[65] W.S. Levine Ed., *The Control Handbook,* New York: CRC Press, 1996.

[66] S.I. Roumeliotis and G.A. Bekey, "An extended Kalman filter for frequent local and infrequent global sensor data fusion," in *SPIE International Symposium on Intelligent Systems and Advanced Manufacturing,* pp. 11-22.

[67] E.W. Weisstein, Frobenius Norm. *From MathWorld--A Wolfram Web Resource.* Available: http://mathworld.wolfram.com/FrobeniusNorm.html

BIOGRAPHICAL INFORMATION

Koushil Sreenath received his Bachelor of Engineering in Electronics and Communications from The Visveshwaraiah Technological University in 2002 where he developed a bipedal full-body exoskeleton based motion capture system as his graduating project. He then worked as an Asst. Systems Engineer in Tata Consultancy Services. His work on writing simulators for car chases introduced him to the broad world of control systems. He began his Masters in Electrical Engineering at the University of Texas at Arlington in January 2004, specializing in the area of systems, control and robotics. He has been a Graduate Teaching Assistant for both undergraduate and graduate level courses. He began working on various research projects as a Graduate Research Assistant at the Automation & Robotics Research Institute for professors Dr. Dan Popa and Dr. Frank Lewis. His current research interests include robotics, bipedal motion, nonlinear control systems, and control theory in computer game systems.