DISTRIBUTED ALGORITHMS FOR CAMERA NETWORK LOCALIZATION AND MULTIPLE TARGET TRACKING USING MOBILE ROBOTS

By

Seema Kamath

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE
MAJOR SUBJECT: COMPUTER SCIENCE

Approved:

_____

Prof. Volkan Isler
Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

November 2007
(For Graduation December 2007)

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

First, I would like to thank my advisor Professor Volkan Isler whose ideas have helped shape this thesis. The experience of working with him has taught me a lot for which I am thankful.

Thanks to Eric Meisner for helping me at various times during the last two years. His contributions toward the ICRA paper on Target Tracking [1] in form of ideas for experiments, code for simulations and libraries and so on have been immense. Special thanks to Nilanjan Chakraborty for patiently listening to my rants and providing valuable inputs.

Dennis, Celia and Sue, you have been very generous and kind to me over the last two years. I am extremely fortunate to have had the opportunity to work with you. Thank you for all you have done for me.

During the course of my M.S. at RPI, I have been fortunate to have many friends who have supported me through hard times. Arthi, Bhavani, Binay, Rahul and Smriti, thank you for making me feel at home in Troy.

To my parents Uma Kamath and Subramanya Kamath, I will forever be thankful. My sister Suma has provided welcome distractions from the cold dreary winters. This section would be incomplete without mentioning her contributions in keeping me entertained.

Sachin, as hard as I try, it is difficult to sum up all that you have done for me.

ABSTRACT

Camera networks are gaining popularity due to the information rich sensing modality of cameras. Some important problems arise while migrating from traditional sensor networks to networks that use cameras as their primary sensing modality. One such problem is that of localization. Localization is the process of finding the pose (position and orientation) of objects in a common frame of reference. It is studied in two different contexts; localization of the cameras/sensors themselves and then localization of targets which the camera network is tracking.

Distributed algorithms are developed for both the problems in this thesis. A fully automated static camera network calibration strategy is proposed and simulated. The algorithm uses probabilistic methods to account for the errors in state updates and measurements. A distributed multi-target tracking application is studied for a mobile camera network. Two algorithms, based on Loopy Belief Propagation (LBP) and Tree Re-Weighted (TRW) algorithm, are evaluated to solve the problem of tracking targets in a network of cameras that have pairwise dependencies. LBP was found to perform better than TRW in our simulations. A prototype of the tracking algorithm using LBP is implemented to show the feasibility of the algorithm in practice.

CHAPTER 1
Introduction

Smart cameras are cameras that have in-built processing and wireless capabilities. Technological advances in sensor technology have made the deployment of such smart camera networks cost effective. The applications of smart camera networks include target tracking, positioning, surveillance, 3D modeling and environment monitoring.

This thesis focuses on localization; an important problem in smart camera networks. Localization is the process of finding the pose (position and orientation) of objects in a common frame of reference. It is studied in two different contexts; localization of the cameras/sensors themselves and then localization of targets which a (possibly mobile) camera network is tracking.

The first problem addresses localization of nodes in static camera networks. Camera networks are often used to make distance measurements in applications like 3D modeling, hence it is necessary that these measurements are made in a common frame of reference so that the distances make sense. The problem is quite difficult; since it involves not only finding the position information of each static camera, but also their orientation in a 3D frame of reference. Further, cameras in the network can have non-overlapping Fields of View (FOV) which makes the problem even more difficult. This thesis presents a novel solution to this problem which does not rely on cameras having overlapping FOV. It employs a mobile robot carrying a calibration pattern to provide landmarks to the static cameras. The landmarks are used by the static cameras to determine their pose information in a scalable distributed fashion. The thesis uses a probabilistic approach to minimize errors which arise due to inaccurate robot motion and inaccurate landmark measurements. The approach uses a particle filter to efficiently track robot motion and Extended Kalman Filters (EKF) to estimate the states of each of the cameras.

The second problem that is studied is that of localizing multiple targets in dynamic smart camera networks. Given a set of targets that are stationary or are

moving, the goal is to track them (i.e provide the position information) using a network of moving monocular cameras. Cameras are bearing sensors and in order to obtain position information of targets, information from more than one camera has to be fused using triangulation. This introduces pairwise dependencies between the mobile cameras. At any given time, a camera might be tracking multiple targets which potentially might be moving in opposite directions. The motion of the camera in order to improve the quality of estimation of the targets depends, not only on the positions of the targets it is tracking but also on the positions of the other cameras that are collaboratively tracking the same targets. The complex dependencies make the general tracking problem NP-Hard. In this thesis, we propose an iterative heuristic algorithm that finds locally optimal positions for each camera to move to in every iteration while improving the overall quality of estimation of the targets.

An important characteristic of the algorithms proposed in this thesis is that are implemented in a distributed fashion. This is a necessary feature due to the information rich sensing modality of cameras. The distributed implementation leads to an order of magnitude savings in communication power.

The main contributions of this thesis are:

1. Developed and simulated a distributed algorithm for localizing a camera network. The localization accuracy was shown to improve with the additional Belief Propagation step.

2. Simulated two different algorithms based on the Loopy Belief Propagation (LBP) algorithm and the Tree Re-Weighted (TRW) algorithm for the Target Tracking problem. The LBP based algorithm outperformed the TRW based algorithm in the simulations.

3. Implemented the LBP based algorithm for tracking targets on an experimental setup. The quality of estimation of the target was shown to improve at every step of the tracking algorithm.

The results in Chapter 3 have appeared in the International Conference on Robotics and Automation (ICRA) 2007 [1].

The rest of the thesis is organized as follows. Chapter 2 presents the solution to the localization problem for static camera networks. Chapter 3 focuses on the target tracking problem in dynamic smart camera networks. The thesis ends with the conclusion in Chapter 4.

CHAPTER 2
Camera Network Localization Using A Mobile Robot

Localization [1] is the process of determining the position and orientation (pose) of an object in a given frame of reference. Localization algorithms that have been developed for traditional sensor networks that use range information as the primary sensing modality are not suited for camera networks. In camera networks, the complete 3-D pose of each of the cameras need to be computed as opposed to traditional sensor networks for which only position parameters are computed. Also, computer vision algorithms used in applications of camera networks are very sensitive to errors in estimated camera pose. Hence, it is necessary to accurately localize camera networks.

A problem with deploying smart camera networks that have GPS capabilities is the prohibitive cost of high-accuracy GPS systems. Another problem with GPS systems are that they provide just position information and need additional sensors in order to be able to provide orientation information. A second solution to the localization problem is to manually calibrate cameras by placing calibration patterns that can be seen by a subset of more than one camera in the network with the additional requirement that subsets of cameras viewing a particular pattern are related to each other by a pair of cameras, one picked from each subset, that have overlapping Fields of View (FOV). In order to achieve this, the network has to be densely deployed which increases the cost.

This chapter presents an algorithm to automatically localize the cameras forming a wireless camera network. The solution uses a robot mounted with a calibration pattern which moves in the FOV of the cameras and provides landmark information for the cameras to localize themselves. An additional Belief Propagation step is implemented to refine the estimated pose of the cameras. The main advantage of this approach is that the cameras need not have overlapping FOV. This allows for efficient placement of the cameras which maximizes the coverage of the network at

[1] Also known as calibration in computer vision literature.

a lesser cost.

The chapter is organized as follows. The related work is discussed in Section 2.1. Section 2.2 presents the formulation of the localization problem. A high-level description of the calibration algorithm along with the details of each part of the algorithm are covered in Section 2.4. Section 2.5 presents the simulation results for our algorithm.

2.1 Related Work

Related work for the localization problem fall into two categories: those that use centralized algorithms and those that use distributed algorithms. The authors in [2] propose a centralized approach in which they use a moving target to provide landmark information. Each camera observes the target for some time instants. Given the relative position of the target at each time instant as seen by the cameras, a Maximum A Posterior (MAP) estimation technique is used to simultaneously estimate the trajectory of the target and the parameters of the camera. Even though the advantage of the approach is that the motion of the target is not constrained, the algorithm does not scale well. With very large camera networks, the size of the non-linear estimation problem to be solved becomes prohibitive.

Distributed algorithms are proposed in [3–5]. The work in [3] proposes an algorithm that uses a calibration device with positioning sensors to localize a camera network. The calibration device provides position information of the features that are used by cameras to localize themselves. A human has to physically carry the calibration device and initiate a trigger that sends signals to the networked cameras to take pictures which are then used in the calibration process. The use of a human makes the process semi-automatic and is not feasible for large networks. An alternating localization and triangulation algorithm is presented in [4]. A set of feature points with unknown locations are used to localize the camera network. The algorithm requires cameras to have overlapping FOV. Once a camera is localized, the pose of the camera is broadcast to the entire network. Using this localization information, the locations of as many feature points as possible are computed. The features whose positions are now known are used to localize cameras that are able

to observe them. The Levenberg-Marquart algorithm is used to refine the estimated translation and orientation of the localized cameras individually. The algorithm still requires cameras to have overlapping FOV which makes it unattractive. A robot mounted with a calibration pattern which moves in the FOV of the cameras is used in [5]. An Extended Kalman Filter (EKF) is used to compute the pose of the robot and the cameras. This approach is less accurate than the one described in this thesis since it does not use a Belief Propagation stage to refine the estimated pose of the cameras. It also does not relocalize the robot when it rotates in the FOV of a localized camera.

2.2 Problem Statement

Let $L = [l_1, l_2, ..., l_n]$ be the sequence of waypoints on a plane that the robot has to visit to appear in the FOV of the cameras in $C = [C_1, C_2, ..., C_k]$. The pose of the camera i in a world reference frame W is given by $C_i = [x_i, y_i, z_i, \theta_i, \phi_i, \psi_i]$ where x_i, y_i and z_i give the x, y, z coordinates of the reference frame of the camera in W and θ_i, ϕ_i and ψ_i give its orientation with respect to the x, y and z axes respectively. At each time instant t, the robot visits one of the waypoints $l_j \in L$. A subset of the cameras $C' \in C$ observe it and make measurements. The measurements made by the camera C_i at time t is given by $M_{c_i}(t) = (T_i(t), R_i(t))$ which gives the translation and rotation from the robot pose X_t at time t to the pose of the camera C_i. For each pair of corresponding X_t and $M_{c_i}(t)$, the objective is to compute C_i while minimizing the errors in C_i due to the errors in robot motion and camera measurement.

2.3 Background

2.3.1 Basic Pinhole Camera Model

Cameras are completely specified by two sets of parameters, the intrinsic parameters and extrinsic parameters. Figure 2.1 depicts the Pinhole camera which is the most basic camera model. The intrinsic parameters consist of the focal length (f), the principal point (p), the skew coefficients and the distortion coefficients. The skew coefficients define the angle between the x and y axes on the principal plane (the plane containing the principal point).

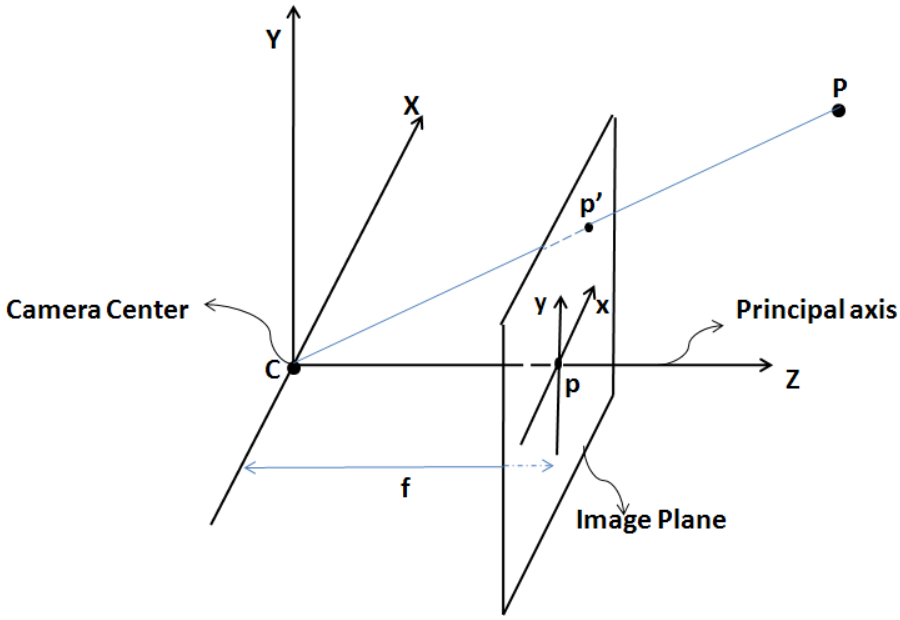

Figure 2.1: Basic pinhole camera

A point $[X \quad Y \quad Z]'$ gets mapped to the point $[x \quad y]'$ in the image plane via a transformation involving the camera matrix which contains the intrinsic parameters of the camera as given in Equation 2.1

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{where,} \tag{2.1}$$

$$K = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Computing the intrinsic parameters of a camera is a one-time process as long as no physical changes are made to the camera after the calibration process. In this thesis, we assume that the intrinsic parameters for the cameras are computed before deployment and do not change over the lifetime of the sensor network.

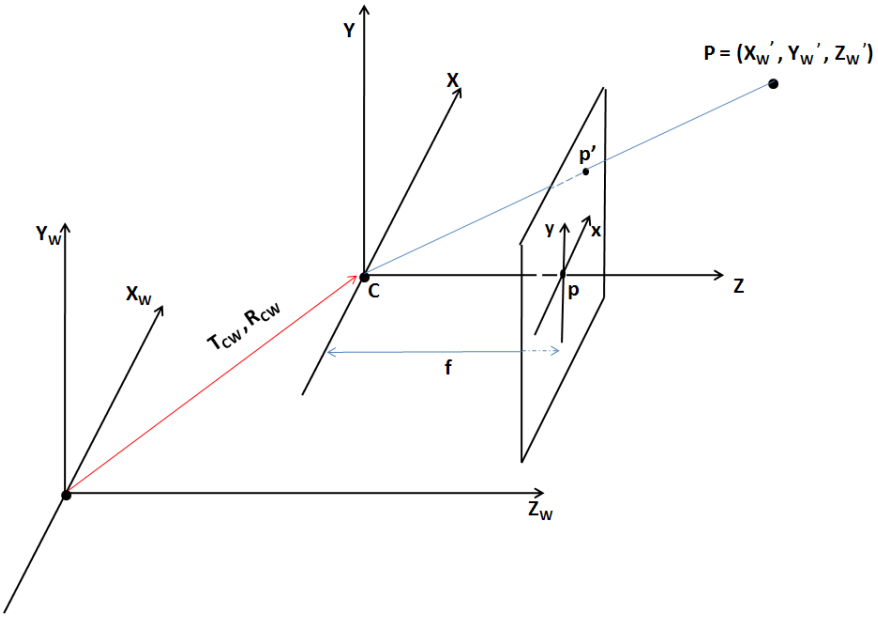

Figure 2.2: Extrinsic Parameters of a Camera

For the Equation 2.1 to hold, the point $P = [X \quad Y \quad Z]'$ must be specified in the frame of reference of the camera. In most cases, the frame of reference of the camera and the World reference frame W in which the P is specified are different. This means that an additional transformation must be applied to P in order to represent it in the frame of reference of the camera. This transformation is given by the extrinsic parameters of the camera. In Figure 2.2, T_{CW} and R_{CW} represent the translation and rotation that must be applied to the point P before it can be projected to the image plane. The transformation is given by Equation 2.3.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{CW} & T_{CW} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} X'_W \\ Y'_W \\ Z'_W \\ 1 \end{bmatrix} \tag{2.3}$$

The final projection of the point $P = [X \quad Y \quad Z]'$ to the image plane of the camera is given by:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K * \begin{bmatrix} R_{CW} & T_{CW} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} X'_W \\ Y'_W \\ Z'_W \\ 1 \end{bmatrix} \tag{2.4}$$

A more detailed description of camera parameters can be found in [6].

Since the extrinsic parameters depend upon the physical environment in which the cameras are deployed, they can only be determined after deployment. This is the focus of the present work.

The simplest method for calibrating a camera for its intrinsic and extrinsic parameters is explained in [7]. This method requires a fixed camera to observe a planar pattern at different orientations. The simplicity arises from the fact that the orientations are random and unknown. The problem of estimating the camera parameters from the images of the planar pattern is formulated as a Maximum Likelihood Estimation problem which is solved using the Levenberg-Marquart Algorithm. In this thesis, we use the Matlab Camera Calibration Toolbox [8] which implements this algorithm.

2.4 Calibration Algorithm

Algorithm 1 presents the high-level description of our approach. The algorithm uses a Particle Filter to model the pose of the robot and an Extended Kalman Filter (EKF) to model the pose of each camera. The particles representing the robots' pose are picked from the proposal distribution obtained empirically. The calibration algorithm consists of two stages.

A path is chosen such that every waypoint in L is visited at least once and successive waypoints are connected by straight line trajectories. If the robot appears in the field of view (FOV) of a camera that has not been previously localized, the normal EKF prediction equations are used to find the camera pose for each particle representing the robots' pose. The robot is relocalized after performing a rotation which makes it face the direction of the next waypoint to visit and the actual rotation is recorded. On visiting a previously localized camera, the robot is

Data: (Initial Position of the robot, List of waypoints to visit)
Result: Pose of the cameras in the World reference frame
Initialization;
while *not visited every waypoint* **do**
    Rotate towards the next waypoint;
    Relocalize the robot and record the rotation;
    Translate to the next waypoint;
    if *in FOV of a camera which has not been previously visited* **then**
        Measure the relative pose of the camera w.r.t the robot ;
        Predict the camera pose;
        Mark the camera as visited;
    else
        Relocalize the robot;
        Update the camera particles;
        BeliefPropogation($G(t)$);
    end
end

Algorithm 1: Calibration Algorithm

relocalized by the camera since the errors in camera measurement are assumed to be very less as compared to the motion errors. The camera parameters are then updated based on the measurement and the new particle set representing the robot. Since revisiting a camera introduces loops in the underlying graph specifying the dependencies between cameras, a message passing algorithm like Loopy Belief Propagation is used to refine the position estimates of previously visited cameras. The proposed algorithm is similar to the FastSlam 1.0 algorithm [9] with some differences as follows. The landmarks in this case correspond to the cameras and they are capable of recalibrating the robot when it moves in its FOV. Another modification to the algorithm arises from the fact that the cameras used are smart cameras that have wireless networking capability. This fact enables the use of message passing algorithms that can refine the estimates of camera poses in a distributed fashion.

Each of the steps in the Algorithm 1 as well as the reasoning behind specific representations is further explored in the sections that follow. The underlying Particle Filter in Algorithm 1 is first described in Section 2.4.1. The motion model for the Robot is discussed in detail in Section 2.4.2. The EKF prediction and update stages are discussed in Section 2.4.3. The Belief Propagation step is discussed in Section 2.4.4

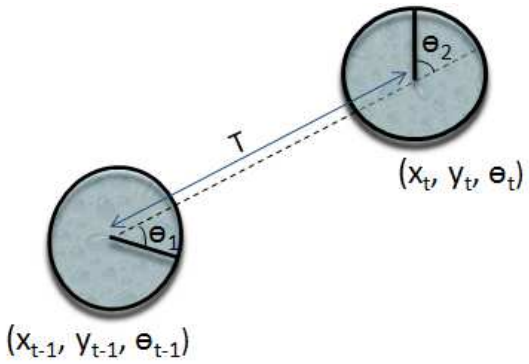

Figure 2.3: Sequence of motions that are executed to move from $(x_{t-1}, y_{t-1}, \theta_{t-1})$ to (x_t, y_t, θ_t), i.e. rotate by θ_1 degrees followed by a translation to ρ meters and a final rotation of θ_2 degrees.

2.4.1 Particle Filter

A Belief represents the knowledge that a system has of its internal state. Bayes Filters have been extensively used to compute beliefs in probabilistic networks. Particle filters are one such implementation of the Bayes Filter that is nonparametric in nature. They represent the posterior by a set of samples drawn from the underlying distribution. Algorithm 1 uses a Particle Filter to represent the evolution of the pose of the robot as it moves. The choice of a Particle Filter for our algorithm can be explained as follows. The mapping from the pose of the robot to that of the camera involves nonlinear transformation. On relocalizing the robot using information from a camera, the errors in the camera pose get mapped to the robot errors which, due to the nature of the nonlinear transformation, cause the error distribution to be possibly non-Gaussian. We use a Particle Filter to represent this non-Gaussian and non-linear system. In order to implement the Particle Filter, the motion model and measurement model need to be specified. The motion model is described in Section 2.4.2. In order to improve the quality of the particles, a resampling algorithm has to be used when the particle set is depleted. In this thesis, we use the basic Select with Replacement Algorithm [10] which selects particles based on their importance weight, i.e, particles with higher weight are duplicated while those with

near-zero weights are eliminated.

2.4.2 Motion Model for the Robot

The pose of the robot at time t is represented by (x_t, y_t, θ_t) where x_t and y_t specify its location on the ground plane and θ_t specifies its orientation. The robot kinematics is described by the state transition probability $p(x_t|u_t, x_{t-1})$ where u_t denotes the control information and x_{t-1} provides the pose at time $(t-1)$. A motion model similar to the Odometry Motion Model described in [9] is used. In the odometry motion model of [9], the control information u_t for the robot is obtained from the encoders in its wheels. In this thesis, we use motion commands represented by the tuple (T_t, θ_t) as control where T_t specifies the distance the robot has at time t after a rotation of θ_t degrees. The motion command is internally converted to wheel velocities in order to perform the motion. Figure 2.3 represents the sequence of motions that are executed to move from $(x_{t-1}, y_{t-1}, \theta_{t-1})$ to (x_t, y_t, θ_t). For our algorithm, the final rotation is not performed since the goal is to just move from one waypoint to the next. Given a starting pose $(x_{t-1}, y_{t-1}, \theta_{t-1})$ and motion command $u_t = (T_t, R_t)$, the final pose of the robot is given by

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + T_t \cos(\theta_t) \\ y_{t-1} + T_t \sin(\theta_t) \\ \theta_{t-1} + R_t \end{bmatrix} \tag{2.5}$$

The translational and rotational errors for robot motion are empirically determined. The setup for determining both the errors are the same. An external camera is used to obtain pictures of the robot mounted with a calibration pattern before and after a known motion command is executed. The extrinsic parameters of the camera with respect to the calibration pattern are calculated each time. The experimental setup for finding the actual motion is depicted in Figure 2.4.

From Figure 2.4, we can obtain the actual motion of the robot from the pose X_{R1} to the pose X_{R2} as follows.

$$X_C = R1 * X_{R1} + T1 \tag{2.6}$$

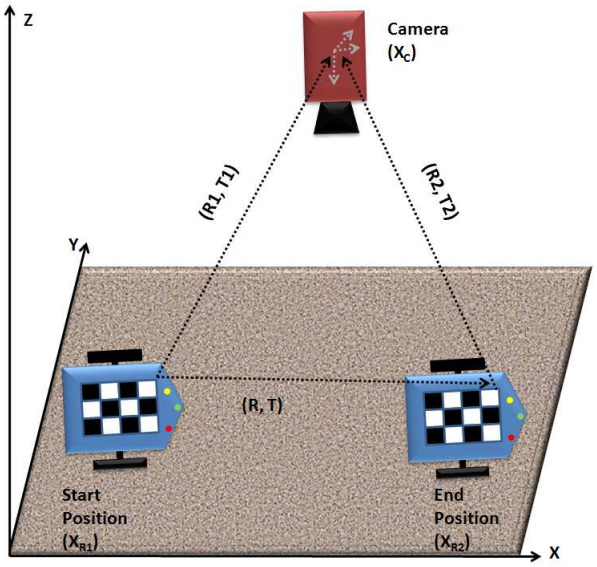

Figure 2.4: The experimental setup for determining the errors in robot motion. $(R1, T1)$, $(R2, T2)$ and (R, T) represent the transformations from X_{R1} to X_C, X_{R2} to X_C and from X_{R1} to X_{R2} respectively.

$$X_C = R2 * X_{R2} + T2 \tag{2.7}$$

$$X_{R2} = R * X_{R1} + T \tag{2.8}$$

$$R1 * X_{R1} + T1 = R2 * X_{R2} + T2 \tag{2.9}$$

$$X_{R2} = R2^{-1} * (R1 * X_{R1} + T1 - T2) \tag{2.10}$$

From equations 2.8 and 2.10,

$$R = R2^{-1} * R1 \tag{2.11}$$

$$T = R2^{-1} * (T1 - T2) \tag{2.12}$$

The experiment to measure the errors in robot motion was conducted on different motion commands (translation-only and rotation-only commands). The errors in the motion command given and the actual motion executed is calculated and plot-

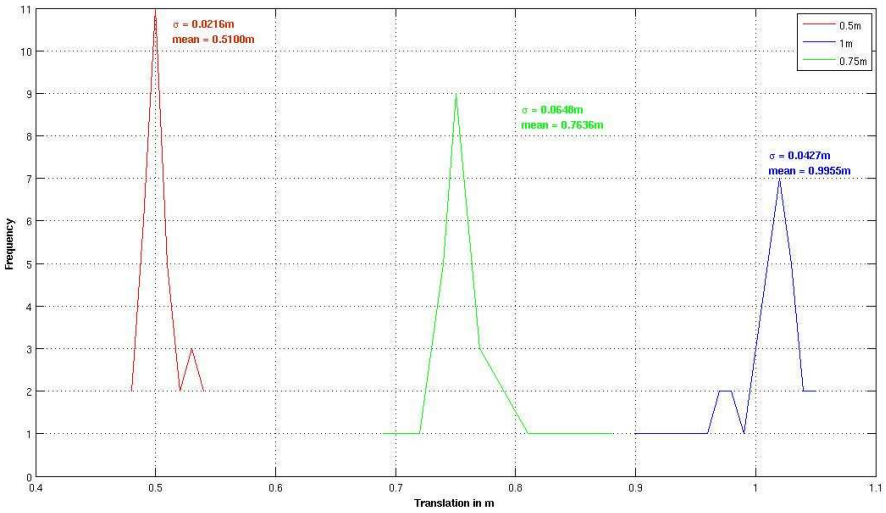

Figure 2.5: Translation error plot for Acroname's Garcia robot

ted. From the plots, the translation error was found to be Gaussian with non-zero mean as shown in Figure 2.5. The slippage in wheel is modeled as a small Gaussian Noise both before and after the translation. The translation model is similar to that shown in Figure 2.3 with the angles θ_1 and θ_2 modeling the wheel slippage errors instead of the angles to rotate. The prediction equation for the state of the robot after a translation of $T_{r,t}$ is given by

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + N(T_{r,t} * \mu_{Trans}, T_{r,t} * \Sigma_{Trans}) \cos(\theta_{t-1} + \theta_1) \\ y_{t-1} + N(T_{r,t} * \mu_{Trans}, T_{r,t} * \Sigma_{Trans}) \sin(\theta_{t-1} + \theta_1) \\ \theta_{t-1} + \theta_1 + \theta_2 \end{bmatrix} \tag{2.13}$$

The rotation errors were found to be very large. In order to keep the errors

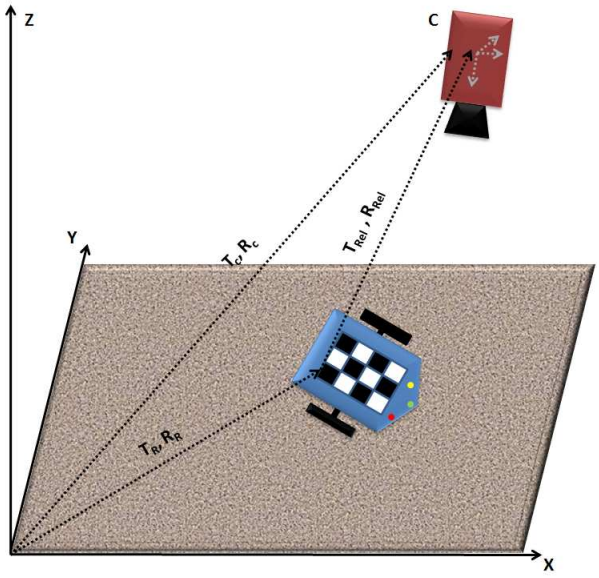

Figure 2.6: Camera Measurement

as low as possible, we recalibrate the robot in the FOV of a calibrated camera after it has made a rotation to face the direction of the next waypoint to visit. Between waypoints, we assume that there are no obstacles so that the robot can follow a straight line path.

2.4.3 Extended Kalman Filter

For each particle $X_r(t, j)$ representing the pose of the robot, the measurement that the camera makes is given by extrinsics of the camera with respect to the calibration pattern mounted on the robot. The pose of the camera is represented by a set of Gaussian distributions corresponding to each hypothesis for the position of the robot. The reason for updating the camera pose by an Extended Kalman Filter (EKF) is that the measurement equation is non-linear and the update is simple in presence of new measurements. It is also easy to measure the performance of our algorithm by observing the change in covariance of the distribution for each particle over successive iterations. The specifics of the EKF implementation are detailed next.

From Figure 2.6, the measurement equation can be written as,

$$g(M_{c_i}, X_r) = \begin{bmatrix} {R_r}^{-1} & -{R_r}^{-1}T_r \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} R_c & T_c \\ 0 & 1 \end{bmatrix} \text{ where,} \tag{2.14}$$

$$M_c = \begin{pmatrix} T_c & R_c \end{pmatrix} \tag{2.15}$$

$$T_c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}' \tag{2.16}$$

$$R_c = Rodrigues\left(\begin{bmatrix} \theta_c & \phi_c & \psi_c \end{bmatrix}\right) X_r = \begin{pmatrix} T_r & R_r \end{pmatrix} \tag{2.17}$$

$$T_r = \begin{bmatrix} x_r & y_r & 0 \end{bmatrix}' \tag{2.18}$$

$$R_r = Rodrigues\left(\begin{bmatrix} 0 & 0 & \theta_r \end{bmatrix}\right) \tag{2.19}$$

The function $Rodrigues()$ in equations 2.17 and 2.19 converts rotation vectors to rotation matrices. The usual EKF predict and update equations can now be written.

Prediction equations for pose $(\mu^j_{c_i,t}, \Sigma^j_{c_i,t})$ of camera c_i corresponding to the particle j of robot pose at time t given the measurement z_t:

$$\mu^j_{c_i,t} = g^{-1}(z_t, X^j_{r,t}) \tag{2.20}$$

$$G = g'(X^j_{r,t}, \mu^j_{c_i,t}) \tag{2.21}$$

$$\Sigma^j_{c_i,t} = G^{-1} S_t G^{-1\prime} \tag{2.22}$$

Update equations:

$$\hat{z} = g(\mu^j_{c_i,t-1}, X^j_{r,t}) \tag{2.23}$$

$$G = g'(X^j_{r,t}, \mu^j_{c_i,t-1}) \tag{2.24}$$

$$Q = G'\Sigma^j_{c_i,t-1}G + S_t \tag{2.25}$$

$$K = \Sigma^j_{c_i,t-1}GQ \tag{2.26}$$

$$\mu^j_{c_i,t} = \mu^j_{c_i,t-1} + g^{-1}(z_t, X^j{}_{r,t}) \tag{2.27}$$

$$\Sigma^j_{c_i,t} = (I - KG')\Sigma^j_{c_i,t-1} \tag{2.28}$$

S_t denotes the measurement covariance. $g'(X^j_{r,t}, \mu^j_{c_i,t})$ represents the derivative of the measurement function g taken with respect to the camera parameters computed at $\mu^j_{c_i,t}$. The prediction equations are used when the robot visits a camera for the first time. The update equations are used when the robot visits a camera that has already been localized.

2.4.4 Belief Propagation

As the robot moves from one waypoint to the next, the errors in its pose estimate start accumulating. In order to accurately localize the camera network, the robot has to, in some way, relocalize itself. One of the ways of doing this is to visit a camera that has previously been localized. The new pose of the robot along with the motion commands constraints the positions from which it arrived to the current pose. A Message Passing (MP) algorithm can be used to propagate this new information back to the previously visited cameras in order to refine the camera pose estimates.

The graph $G(t) = (V(t), E(t)$ on which MP is run is constructed at each time interval as follows: Initially, $G(0) = (V(0), E(0))$ where the vertices at time 0 is $V(0) = NULL$ and the edges at time 0 is $E(0) = NULL$. The graph at time $G(t) = (V(t), E(t))$ can be computed iteratively.

$$V(t) = V(t-1) + f(L, t) \tag{2.29}$$

$$E(t) = E(t-1) + (f(L, t-1), f(L, t)) \tag{2.30}$$

where $f(L, t) = C_i$ if camera C_i sees the robot which is at $l_t \in L$ at time t.

Since revisiting a camera introduces loops in the underlying graph, a message passing algorithm like the LBP [11] can be used. Even though no guarantees can be given on the results of the algorithm, empirically, it has been shown to perform very well.

2.5 Simulation

The proposed algorithm was simulated in Matlab. The Ground-Truth (GT) locations of the cameras for the simulation were picked from a uniform distribution over the workspace. The inputs to the simulation consisted of the GT for the cameras and the list of waypoints for the robot to visit. In order to keep the simulation as real as possible, the positions of the feature points (corners of a checkerboard pattern mounted on the robot) were initialized in the world reference frame. The positions of the features corresponding to each particle representing the robot pose were updated with every motion command. The actual positions of the feature points in 3-D were projected onto the image plane of the camera using Equation 2.1 and the GT locations of the camera that was viewing the robot.

For each particle representing the camera pose, the extrinsics of the camera were obtained from the image points corresponding to the 3-D locations of the features using the Matlab Camera Calibration Toolbox [8]. Using the extrinsics as the measurement z, the EKF prediction or update equations were used, depending on whether the camera is being visited for the first time or not, to compute the camera pose.

The expectation of the camera pose mean is computed for each camera and is plotted in Figure 2.7. The estimated camera pose was found to improve after the Belief Propagation step. The uncertainty ellipses for the camera pose is shown in Figure 2.8. The ellipses were observed to shrink after the EKF update and Belief Propagation step.

The trajectory that the robot followed to calibrate the camera network is shown in Figure 2.9. Our algorithm assumes that this is already provided. In reality, this trajectory is obtained by an exploration stage which determines the vague directions that the robot needs to move in order to come into the FOV of each of the cameras. The performance of the Belief Propagation step depends upon the trajectory followed by the robot.

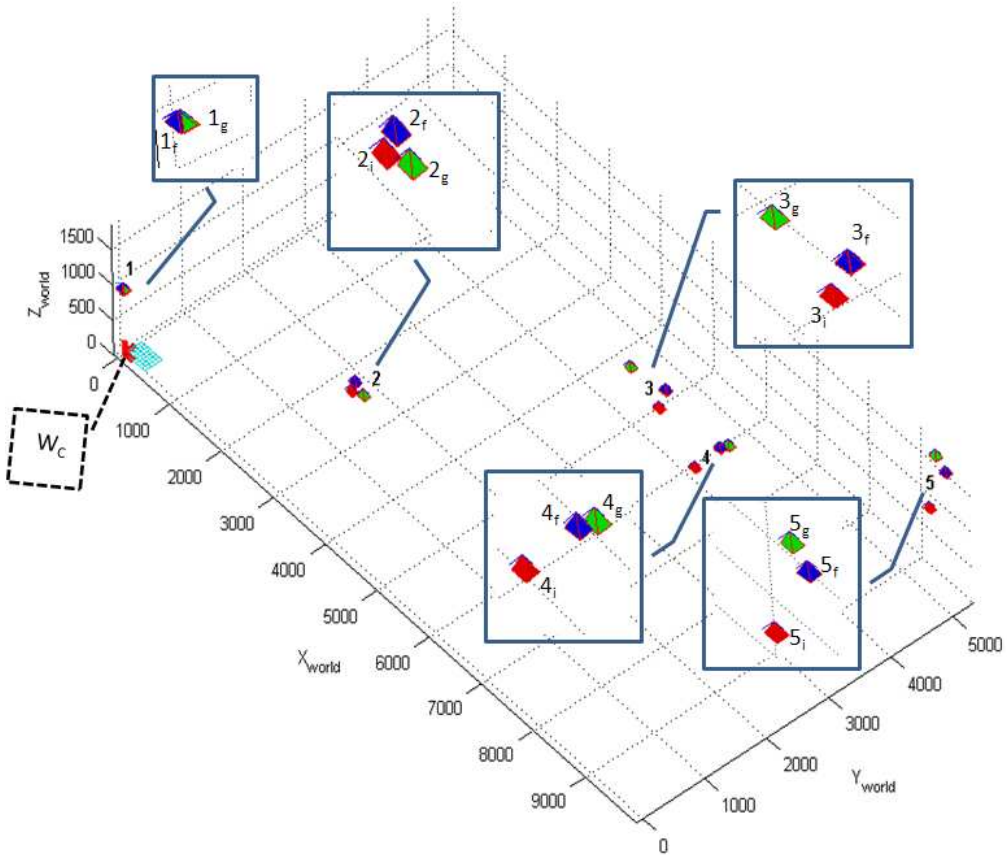

Figure 2.7: The final expected pose of the cameras in the network. C_i represents the initial estimate of the camera pose, C_f represents the final estimated pose of the camera and C_g represents the ground truth pose of the camera where $C \in$ {1, 2, 3, 4, 5}. W_C represents the World Center.

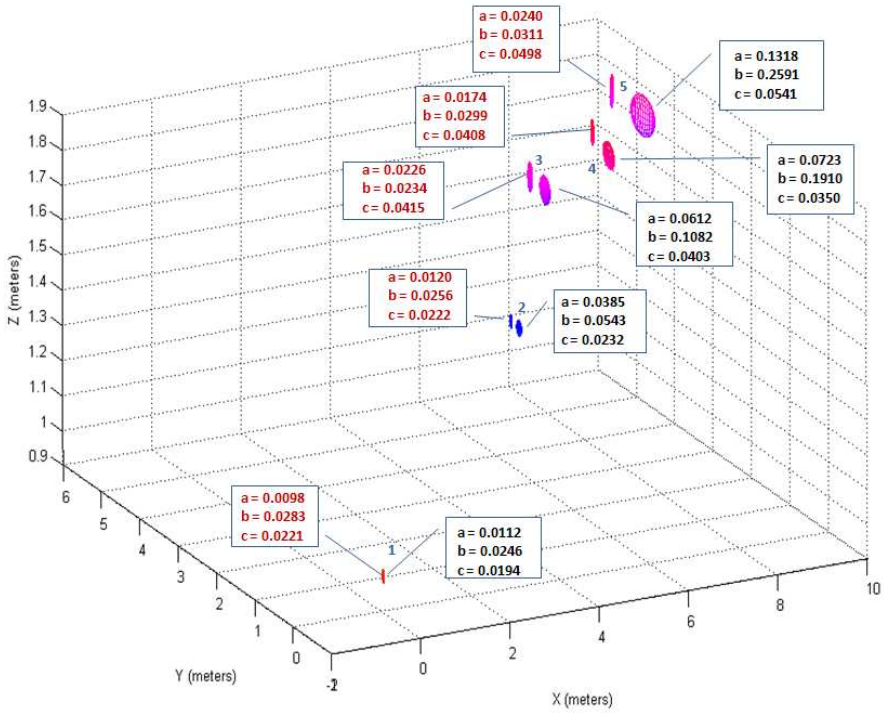

Figure 2.8: Two uncertainty ellipses are shown for each of the 5 cameras in the camera network. The ellipses on the right represent the uncertainty in camera pose before the Belief Propagation step. The ellipses on the left represent the final uncertainty in camera pose after the Belief Propagation step. The equation of an ellipsoid is given by $x^2/a^2 + y^2/b^2 + z^2/c^2 = 1$ where a, b and c are the radii of the ellipse along the x, y and z axes. The numbers corresponding to each ellipse represent its radii. The radii are measured in meters.

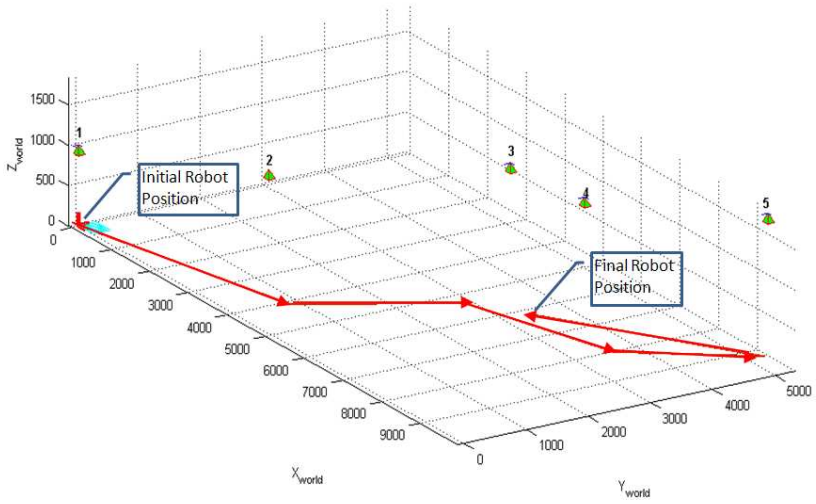

Figure 2.9: The trajectory followed by the robot to calibrate the camera network is depicted by the line segments on the XY plane. The arrows indicate the direction of motion of the robot.

CHAPTER 3
Target Tracking

A mobile sensor network is a network of mobile devices equipped with sensing, communication and computation capabilities. In this thesis, we study a typical application of mobile sensor networks: tracking multiple targets. Imagine that we are given (i) a mobile sensor network specified by the locations of the sensor nodes and (ii) a communication graph whose edges correspond to nodes that can communicate with each other. Our goal is to estimate the positions of (possibly many) targets as they move around in the work space. In this scenario, the following questions arise:

i) Target-sensor assignment: which sensors should track which targets?

ii) Motion planning: how should the sensors move, so that the overall quality of the estimation improves?

In answering these questions, one must address the following issues:

The coupling between assignment and motion planning. As the sensors move, the optimal target-sensor assignments will change. Therefore, these assignments must be updated in a dynamic fashion.

Dependencies between sensors. Sensor network nodes are typically quite modest in terms of their sensing capabilities. Consequently, sensor nodes must collaboratively obtain position estimates. However, this raises the following issue. Suppose three sensors s_1, s_2 and s_3 are tracking targets t_1, t_2 and t_3. Sensor pair (s_1, s_2) is assigned to t_1, (s_2, s_3) to t_2 and (s_1, s_3) to t_3. The quality of tracking t_1 depends also on the position of s_2. Therefore s_1 must negotiate a good position with s_2. But then s_2 must negotiate a good location with s_3 (because of t_2) who, in turn, must negotiate a good location with s_1 because of t_3! Performing these cyclic negotiations in a distributed fashion while improving the quality of the estimates is one of the primary issues addressed in our present work.

Communication and energy constraints. Battery power is the primary limitation determining the lifetime of the network. An important source of power con-

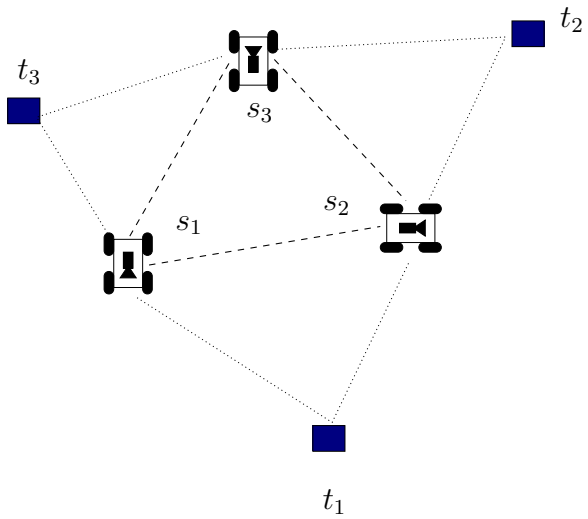

Figure 3.1: A setup where three targets are tracked by three sensors. Sensors s_1 and s_2 are tracking target t_1, s_2 and s_3 are tracking t_2, and s_3 and s_1 are tracking t_3.

sumption is communication. When two nodes are tracking a target, they will need to exchange information. Therefore, it is desirable that the energy consumption required to transfer the information from one node to another is minimized.

We study algorithms to compute sensor assignment and motion planning strategies for mobile sensor networks. We focus on triangulation based position estimation where two sensors are utilized to estimate the position of a target.

We start with an overview of related work and continue with a formal statement of the one-step tracking problem (Section 3.2) for which we present an iterative and distributed algorithm. In Section 3.3.1, we investigate the computational complexity of one-step tracking and observe that the general form is not only NP-hard but also hard to approximate.

In Section 3.3.2, we show how one-step tracking can be formulated as an energy minimization problem. Even though energy minimization with pairwise interactions is hard to approximate, the problem has been extensively studied and there are algorithms in the literature which work well in practice. In particular, two algorithms, loopy belief propagation and tree-reweighted message passing, have been demonstrated to solve various problems (e.g. data association) in sensor-networks efficiently. After demonstrating the performance of these two algorithms with sim-

ulations, we present the complete tracking algorithm in Section 3.4. We conclude the chapter with results from real experiments conducted on a sensor network.

3.1 Related work

Tracking is a fundamental problem in sensor networks and therefore the problem received significant attention recently. In [12], an *information driven sensor query* approach was proposed. In this approach, at any given time, only a single sensor (leader) is active. After obtaining a measurement, the leader selects the most informative node in the network and passes its measurement to this node which becomes the new leader. In subsequent work, researchers addressed leader election, state representation, and aggregation issues [13, 14]. A sensor selection method based on the mutual information principle is presented in [15]. Recently, an *entropy based heuristic approach* was proposed [16] which greedily selects the next sensor to reduce overall uncertainty. Other aspects of tracking in sensor networks have received significant attention as well. In [17], the problem of estimating target's location and velocity using minimal information has been addressed. The problem of assigning n disjoint pairs of sensors to n targets so as to minimize the overall error in the estimation has been studied in [18]. The problem of choosing the best subset of cameras for a given placement has been studied recently in [19]. A related line of research is cooperative localization, where a group of robots or network-nodes localize themselves by collecting information over the network [20–22].

For mobile sensor networks, the problem studied in our present work is related to the coverage problem. The problem of relocating sensors to improve coverage has been studied in [23]. In this formulation, the sensors can individually estimate the positions of the targets. However, the quality of coverage decreases with increasing distance. The algorithm presented studies a similar problem but for sensors which cannot estimate the targets' positions by themselves.

The problem of controlling the configuration of a sensor team which employs triangulation for estimation has been studied in [24] where the authors present a numerical, particle-filter based framework. A recent related result was presented in [25] where the problem of relocating a sensor team whose members are restricted

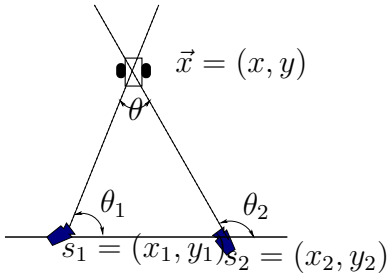

Figure 3.2: The uncertainty in estimating the position of the target at x is given by: $U(s_1, s_2, x) = \frac{d(s_1,x) \times d(s_2,x)}{\sin\theta}$

to lie on a circle and charged with jointly estimating the location of the targets was studied.

The one-step tracking problem (Section 3.2) has been studied recently in [26] for the special case where the communication graph is a tree. We present (i) a solution to the general case and (ii) a full description of a distributed, iterative tracking algorithm. In solving the one-step tracking problem we utilize message-passing based distributed estimation algorithms. These algorithms have been utilized in solving other problems in sensor networks and demonstrated to work well in practice [27–30].

3.2 Problem statement

Let $S(t) = [s_1(t), s_2(t), \ldots, s_n(t)]$ be the state of the sensor network at time t where $s_i(t)$ is the state (position) of the i^{th} sensor at the time. Let us construct a dynamic graph $G(t) = (V(t), E(t))$ whose nodes $V(t)$ correspond to the set of sensors at time t and there is an edge between two nodes if the corresponding nodes in the network can communicate without spending more energy than a given budget. This energy budget can be specified as, for example, the number of hops required to transmit a message.

In this thesis, we focus on triangulation based estimation where two sensors s_i and s_j can estimate the position of an object at location x. However, the uncertainty in this estimation is given by a function $U(s_i, s_j, x)$. For example if the sensors measure bearing then $U(s_i, s_j, x) = \frac{d(s_i,x)d(s_j,x)}{\sin \angle s_i x s_j}$ where $d(s_i, x)$ is the Euclidean distance between s_i and x and $\angle s_i x s_j$ denotes the angle between the target and the sensors [31]. Therefore if the target is collinear with the sensors, the uncertainty

becomes infinite which leads to an arbitrarily bad estimation (Figure 3.2). We will use this uncertainty measure throughout this chapter to demonstrate our results.

Suppose at time t we are given the location $x(t)$ of a target, the uncertainty function, the state of the sensor network $S(t)$ and the communication graph $G(t)$. We can define a function

$$assign(x(t), S(t), G(t)) = \arg \min_{(s_i,s_j)\in E(t)} U(s_i, s_j, x(t))$$

which returns the best sensor pair in the network for tracking the target at $x(t)$. Similarly, we define the error in tracking this target as

$$error(x(t), S(t), G(t)) = U(assign(x(t), S(t), G(t)), x(t))$$

which is simply the uncertainty achieved by the optimal selection. Suppose there are m targets and let $x_i(t)$ be the position of the i^{th} target at time t. We are now ready to define the *tracking problem*:

Compute an admissible trajectory $s_i(t)$ for each sensor in the network so as to minimize the overall uncertainty in tracking given by:

$$\frac{1}{mT} \sum_{i=1}^{m} \int_{t=0}^{T} error(x_i(t), S(t), G(t)) \tag{3.1}$$

where T denotes a (given) terminal time.

Clearly, the value given by the Equation 3.1 for the optimal solution of the tracking problem is the best possible error that can be achieved by the network. However, in practice, there are significant challenges in computing trajectories that achieve this value. First of all, the target trajectories may not be available. Second, the sensor pair for a target can change from one sensor pair to another instantaneously and hence computing the best pair for the next time instance in a distributed fashion can be very difficult. A related issue is that the graph $G(t)$ depends on the locations of the nodes as well as the environment (occlusions may prevent communication) and the nodes may not have access to the entire graph at all times.

To deal with these issues, we propose a solution where the time interval $[0, T]$ is divided into epochs of possibly varying length. Each epoch starts with an initialization phase, where the network computes target/sensor-pair assignments based on the estimated positions of the targets and its current state (we defer the details of initialization and a full description of the sequence of events in an epoch to Section 3.4). Once the initialization phase is complete, we ask the question: Given target/sensor-pair assignments, where should each sensor move to minimize the error until the end of the epoch? We call this problem the *one-step tracking problem*. Formally, the one-step tracking problem is to compute a location s_i for each sensor such that s_i is reachable within the epoch, so as to minimize

$$\sum_{i=1}^{m} U(s_i^1, s_i^2, x_i) \tag{3.2}$$

where x_i is the position of target i at the beginning of the epoch and s_i^1 and s_i^2 are the computed locations of the two sensors assigned to target i during this epoch.

In essence, we force the target/sensor-pair assignments to remain static within an epoch. This is justified, because as we will see shortly, the length of an epoch is required to be only long enough to pass a few rounds of messages across the network – which is much shorter than the time it takes for either the sensor nodes or the targets to travel a significant distance. Therefore, the solutions to one-step tracking problems can be combined into a filter to solve the tracking problem: compute the assignments, compute the next locations, repeat.

Before we present the details of the overall algorithm, let us start with the one-step tracking problem.

3.3 One-step tracking problem

In this section, we first establish the computational complexity of the one-step tracking problem. Next, we show how to formulate the problem as an instance of energy minimization with pairwise constraints. This allows us to utilize distributed message passing algorithms for energy minimization. We show how two of these

algorithms which have been extensively studied in the literature can be adapted to solve the one-step tracking problem and compare them in the context of one-step tracking with simulations.

3.3.1 Hardness results

In this section, we present computational lower bounds on the hardness of the one-step tracking problem. One-step tracking is closely related to the well-known graph coloring problem.

In the graph k-coloring problem, we are given a graph $G = (V, E)$ and the goal is to choose one of the k available colors for each vertex in such a way that adjacent vertices receive different colors. The coloring problem is NP-hard. Further, unless $P = NP$, it is not possible to find efficient algorithms to approximate coloring within a factor better than $o(|V|)$ [32].

Graph coloring can be reduced to one step-tracking as follows: Given a graph G with n vertices, we produce an instance of one-step tracking with n sensors. The communication graph is identical to G. We assign a target for each edge. There are k available locations where sensors can move (each location corresponds to a color). The cost of tracking a target is infinite if the sensors move to the same location. It is zero otherwise. It is easy to see that G is k colorable if and only if there is a solution to the one-step tracking problem that achieves zero error.

Since graph coloring can not be approximated in general, it is also not possible to find a general solution to solve the one-step tracking problem. In the next section, we show how one-step tracking can be formulated as an energy minimization problem.

3.3.2 Energy minimization formulation

Energy minimization with pairwise constraints, also known as the graph labeling problem is defined as follows. We are given a graph $G = (V, E)$ and a finite set of labels $L = \{l_1, \ldots, l_k\}$. The objective is to assign labels to the nodes of the graph to minimize the energy given by

$$\sum_{v \in V} E_1(v, l(v)) + \sum_{(u,v) \in E} E_2(u, l(u), v, l(v)) \tag{3.3}$$

In the equation above $l(v)$ denotes the label of node v. The function E_1 is the cost of assigning a label to a node and is known as the data cost. The function E_2 is the smoothness term which is a function of the labels assigned to the endpoints of edges.

We express the sensor placement task as a graph labeling problem as follows: the labels are possible locations for nodes to travel until the end of an epoch. The data cost is the energy spent in traveling from the node's current location to the assigned location. The smoothness function is obtained by summing the quality of coverage $U(\cdot)$ on all targets for a given placement of the sensors (see Figure 3.2 for an example uncertainty function).

Energy minimization on graphs with cycles is NP-hard (it is harder than the graph coloring problem.). Nevertheless, the problem has been extensively studied and researchers have designed algorithms which work well in practice by exploiting substructures in the graph. In particular, two algorithms, loopy belief propagation (LBP) and tree-reweighted message passing (TRW), have been demonstrated to solve various problems (e.g. data association) in sensor-networks. Descriptions of the LBP and TRW algorithms can be found in [11] and [33] respectively. In Section 3.2, we evaluate the performance of these algorithms for the one-step tracking problem using simulations. We first describe the full tracking algorithm in the next section.

3.3.3 Loopy Belief Propagation

The loopy belief propagation (LBP) algorithm is essentially the message passing procedure described above for trees adapted to general graphs. A detailed description of the algorithm can be found in [11].

The min-sum algorithm for finding the MAP estimate of the labels is used in our implementation. The specifics of the algorithm follows. In the initialization phase, the robots localize themselves using external sensors like GPS. Assuming an initial estimate of the target locations, the sensor-target assignment is done in the

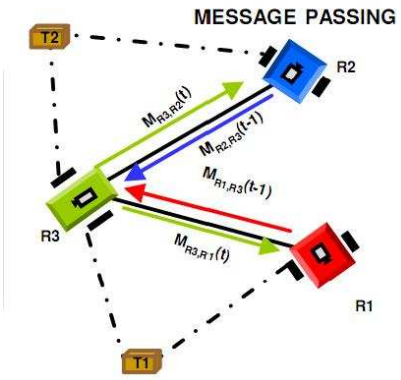

Figure 3.3: $R1$**,** $R2$ **and** $R3$ **are the robots.** $T1$ **and** $T2$ **are the targets.** $M_{r2,r1}(t-1)$ **represents the message passed from robot** $R2$ **to** $R1$ **at time** t **which consists of the position of the robot** $R2$ **at time** $(t-1)$ **and so on.**

second phase. The messages are then passed in parallel along the edges until the convergence criterion is met, i.e. the positions (labels) of the nodes are unchanged at the end of an iteration.. The message that each node passes to its neighbor consists of the best possible location (label) that it can move to given the neighbors positions as shown in Figure 3.3. Since the messages are passed in parallel, the algorithm lends itself to distributed implementation.

3.4 Full Tracking Algorithm

The tracking algorithm consists of a series of epochs. The sequence of events within an epoch is shown in Figure 3.4. We assume that during an epoch a spanning tree T of the communication graph is available and, further, that this tree is rooted at an arbitrary node to establish child-parent relationships. Building a spanning tree in distributed manner is a well-studied problem. We refer the reader to [27] for details.

We use the notation $T(u)$ to denote the subtree of T rooted at node u.

The first step in an epoch is *initialization*. All nodes contribute to the initialization phase. The process starts from the leaves. Let u be a leaf node and

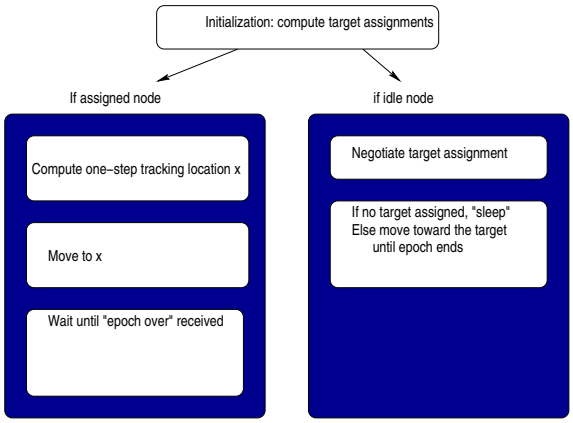

Figure 3.4: Sequence of events in an epoch.

v be its parent. Node u passes a message to v. The message consists of u's own location along with a list of targets that u can contribute in tracking. The initialization process propagates up to the root as follows: Let w be an internal node and $U = \{u_1, \ldots, u_k\}$ be its children. Node w will compute a list L_w which has an entry for each target that can be tracked by nodes in $T(w)$. The entry $L_w(x)$ for a target at x contains the best pair in $T(w)$ to track x and the expected error in tracking x with this pair. The value of the error is computed by the position of the target and the sensor at the time of the computation.

The computation of $L_w(x)$ is achieved as follows: Let $G(w)$ be the subgraph of the communication graph G induced by the vertex set $U \cup \{w\}$. Upon receiving tables L_{u_i} from all of its children, node w compares the values in L_{u_i} with the values obtained from the pairs in $G(w)$ and picks the best among them. When all entries of L_w are filled, node w passes L_w to its parent.

After the root node computes its table, the best sensor pair for each target has been computed. The root propagates these assignments downwards toward the leaves. Nodes which are assigned to targets begin the one-step tracking algorithm described in Section 3.3. When a node reaches its destination, and receives "epoch over"signals from its children, it passes the "epoch over" signal to its parent. The epoch ends when the root has reached its destination and received "epoch over" signals from its children. Nodes which are not assigned may be kept idle to conserve power, or commanded to move toward targets which may lead to favorable

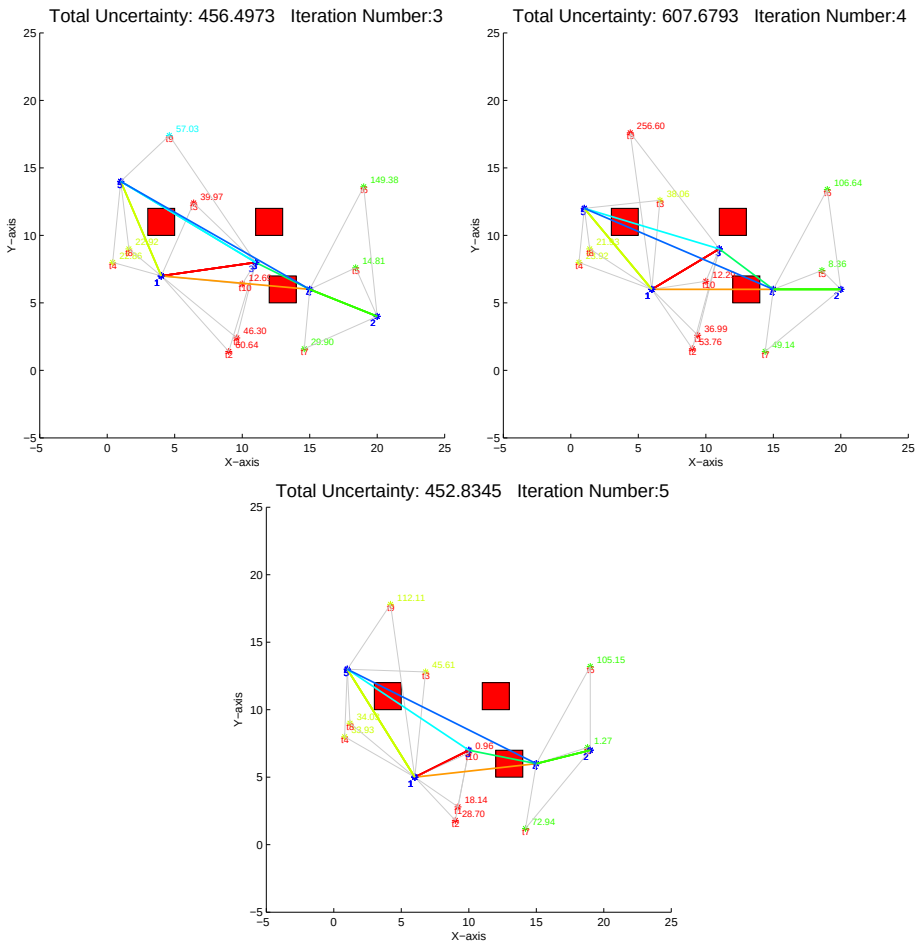

Figure 3.5: A tracking instance with visibility constraints. Internode communication is distance based. The quality of the tracking is a function of both geometry and visibility. Nodes can not see through the obstacles (large red squares).

configurations in the future.

In Figure 3.5, we demonstrate the full algorithm using moving targets and reassignment with the addition of visibility constraints. This constraint is enforced by simply adding a large penalty to the cost function in configurations where motion or visibility is obstructed. In iteration 4, a target is reassigned to a different sensor pair. The total uncertainty increases temporarily before returning to approximately the same value as in iteration 3.

3.5 Simulations

In this section, we compare the performance of LBP and TRW for the one-step tracking problem. In the first simulation, the targets remained stationary throughout the simulation. In this experiment, the target assignments were not updated dynamically. The evolution of the total uncertainty as a function of the number of epochs in the networks is shown in Figure 3.9-left. In this experiment, LBP achieved near optimal performance and in fact converged to the value achieved by OPT: the optimal solution obtained by enumeration.

In the next simulation, we updated the target assignments as the sensors moved. Comparing the results with the previous simulation, we observed that target reassignment improves the performance of tracking. The evolution of the total uncertainty as a function of the number of epochs in the networks is shown in Figure 3.9-middle.

Finally in the last simulation we studied a scenario where the targets are moving. The evolution of the total uncertainty as a function of the number of epochs in the networks is shown in Figure 3.9-right.

In all these experiments we observed that LBP consistently outperformed TRW and its performance was close to OPT. To further test this observation, the outcomes of 25 randomly generated tracking instances is reported in Figure 3.10. It is seen that when target reassignment was not utilized, on the average LBP was within factor 1.2 of OPT's performance whereas TRW's average performance was a factor of 1.35 (in these experiments OPT did not utilize reassignment as well.) With reassignment, LBP was within a factor of 1.45 which is better than TRW's performance of 2.1.

It is interesting to note that both LBP and TRW occasionally converged to a better value than OPT. Therefore optimal local performance does not necessarily imply the best performance in time.

Since the average performance of LBP is better than our implementation of TRW, LBP seems to be a better choice for the one step tracking problem in our experiments.

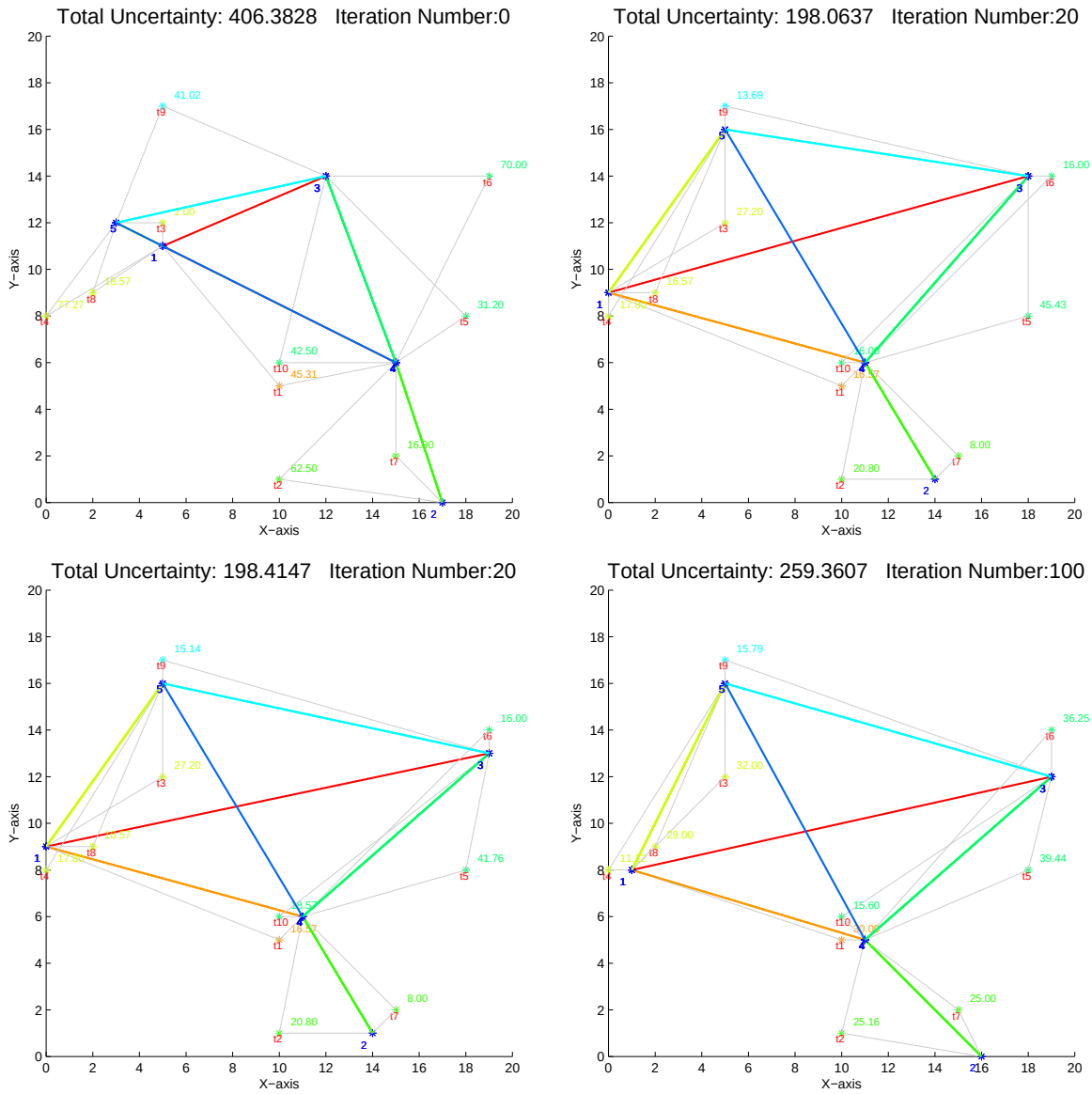

Figure 3.6: The top-left figure shows the initial locations of the targets and the configuration of the network. Next three figures (top-right, bottom-left and bottom-right) show the final state that the network converged to when running OPT, LBP and TRW respectively. In this experiment, both the targets and the sensor assignments remained static.

3.6 Experiments

The experiments were conducted on a sensor network platform consisting of Acroname Garcia robots equipped with Intel Stargate boards and Ambicomm wire-

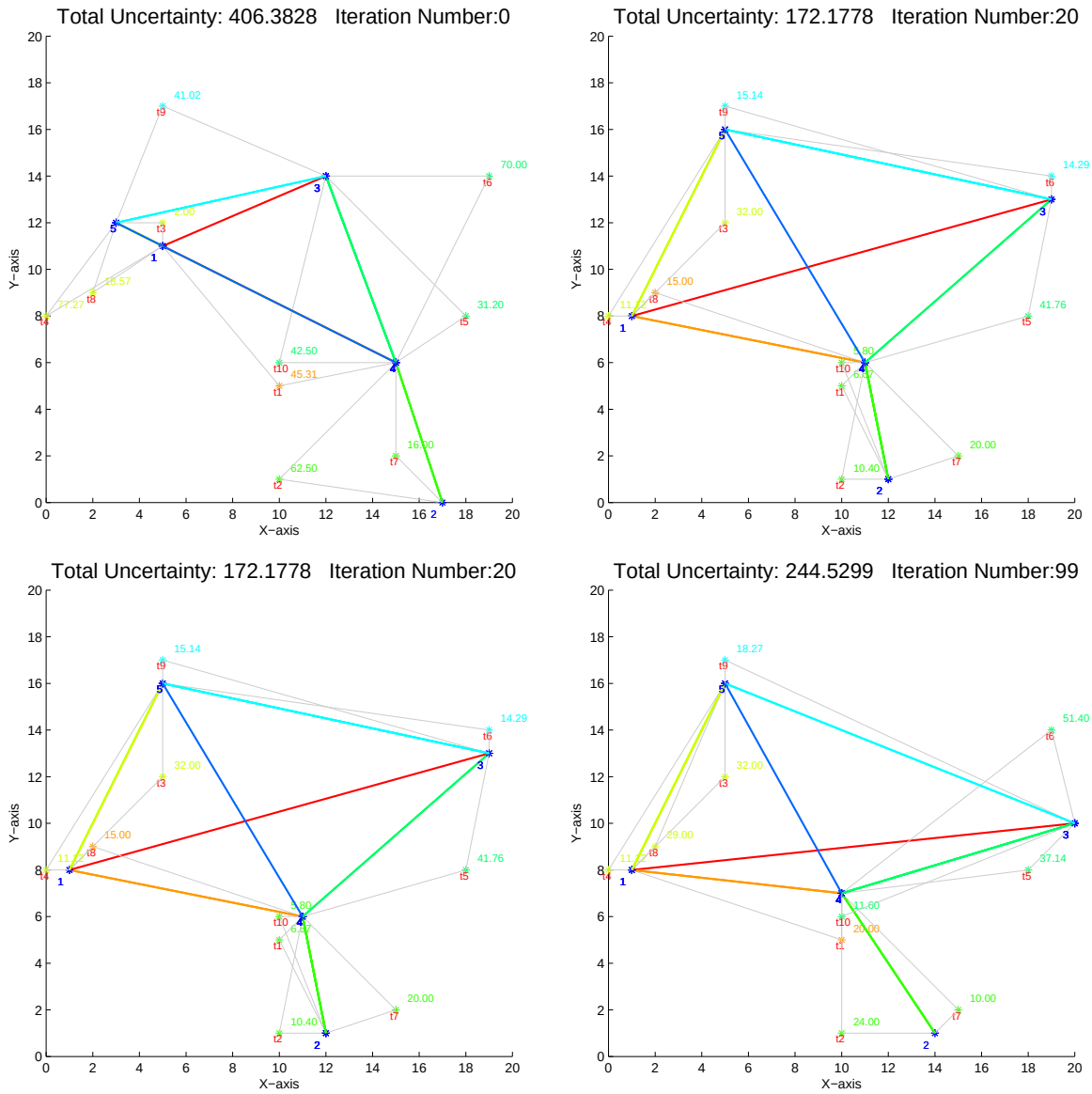

Figure 3.7: The top-left figure shows the initial locations of the targets and the configuration of the network. Next three figures (top-right, bottom-left and bottom-right) show the final state the network converged to when running OPT, LBP and TRW respectively. In this experiment, the targets remained stationary but sensor assignments were updated at the end of every epoch.

less cards. Logitech Pro 4000 webcams were mounted on each of these robots and used as the sensor. A Point Grey Bumblebee camera was mounted on the ceiling and was used to localize the robots.

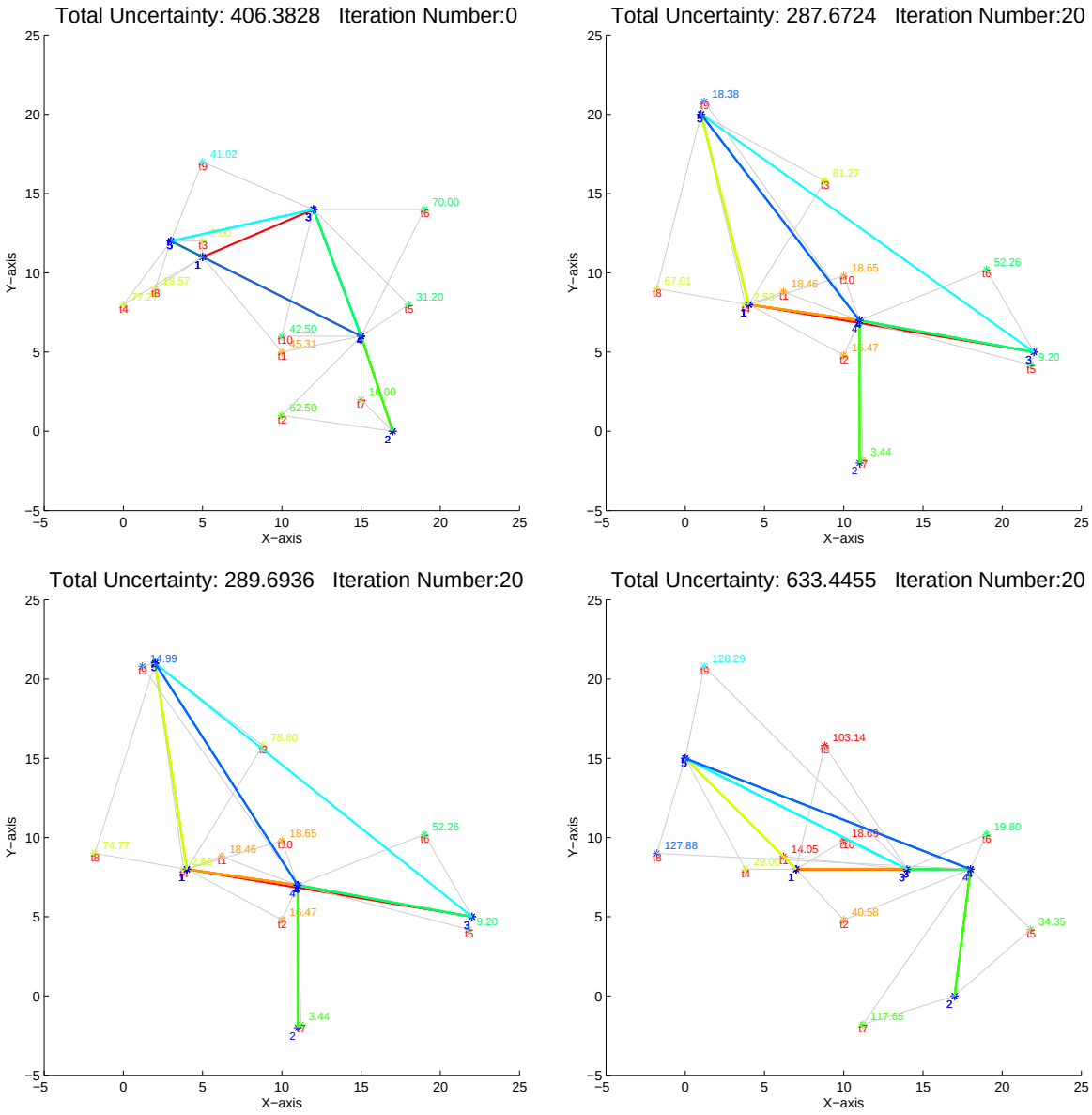

Figure 3.8: The top-left figure shows the initial locations of the targets and the configuration of the network. Next three figures (top-right, bottom-left and bottom-right) show the final state the network converged to when running OPT, LBP and TRW respectively. In this experiment, the targets were moving in a random direction and sensor assignments were updated at the end of every epoch.

Figure 3.11 shows the control flow for the experiments. The details of each step are as follows.

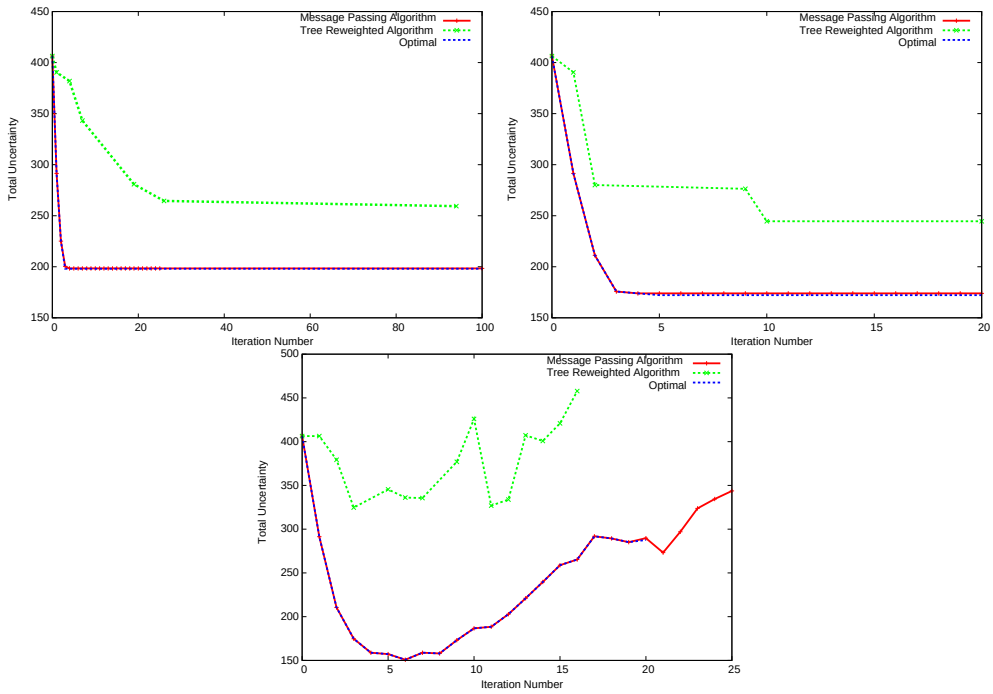

Figure 3.9: Each figure shows the evolution of total uncertainty for the three algorithms OPT, LBP and TRW. The leftmost figure corresponds to the case when the targets are static. The second figure from the left shows the total uncertainty over successive epochs when the sensor assignment were updated at the end of every epoch. The rightmost figure shows the total uncertainty over epochs for moving targets with sensor assignments being updated at the end of each epoch.

1. The stereo camera provides position information to the robots.

2. The cameras mounted on the robots take pictures of the target.

3. The pictures taken from the camera on the robots are sent to a central processor which performs the triangulation.

4. The result of the triangulation which is an estimate of the position of the target is sent back to the robots.

5. The robots decide the best position to move to in the next epoch via LBP.

The setup for the experiment consisted of three robots (r_1, r_2, r_3) tracking one target (t) as shown in Figure 3.13. The communication graph for the robot

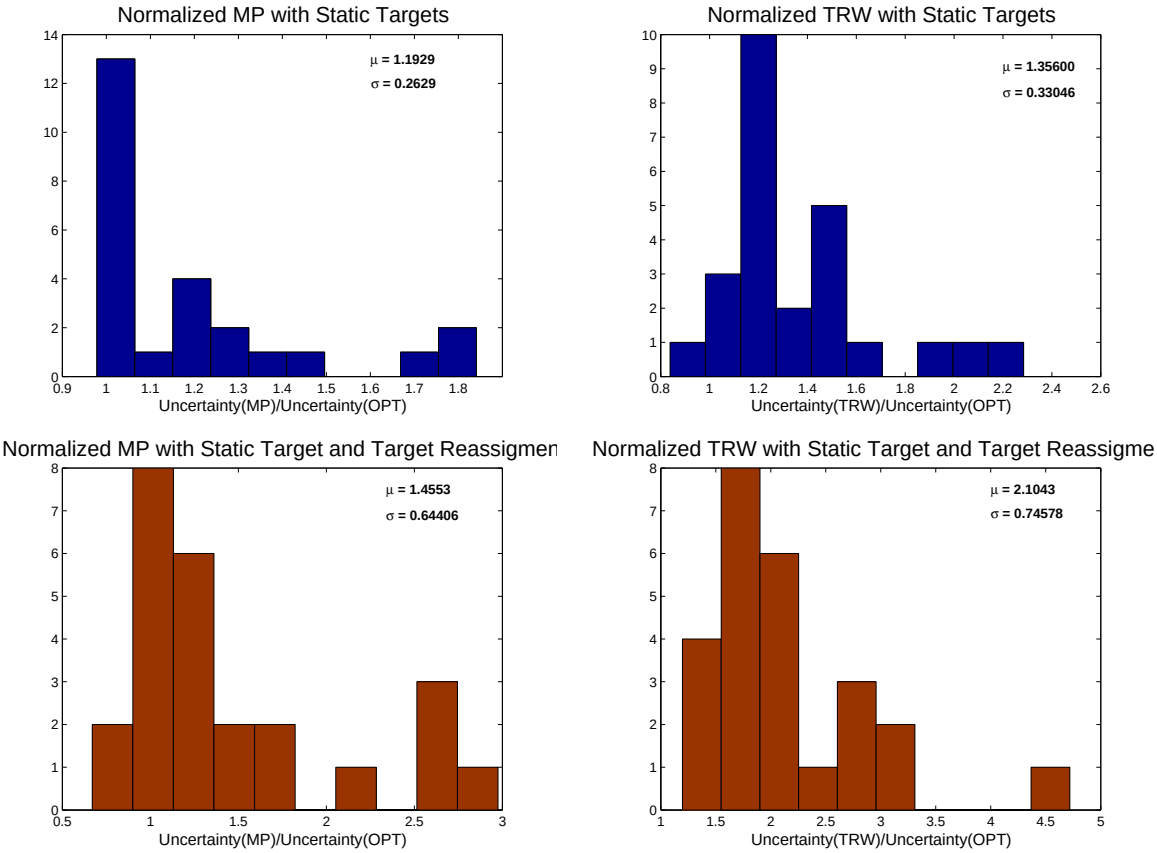

Figure 3.10: TOP TWO FIGURES: Static targets, no reassignment. Performance of LBP (left) and TRW (right). BOTTOM TWO FIGURES: Static targets, with reassignment. Performance of LBP (left) and TRW (right). The numbers on the x-axes indicate the performance of an algorithm normalized with the performance of OPT.

network was static and had links between r_1 and r_2 and r_1 and r_3. There were no loops in the network. The tracking algorithm was run for the case where the target is stationary over consecutive epochs. The robots were placed such that $\angle r_1 t r_2$ was approximately 90 degrees and $\angle r_1 t r_3$ was approximately 45 degrees. The robots were localized using an external stereo camera and triangulation was performed using correspondences in the images of the target obtained from webcams mounted on the robots. Using the estimated positions of the target due to the two robot pairs got from the triangulation step and the positions of the robots from the stereo camera, the tracking algorithm (using Loopy Belief Propagation) was run and the best

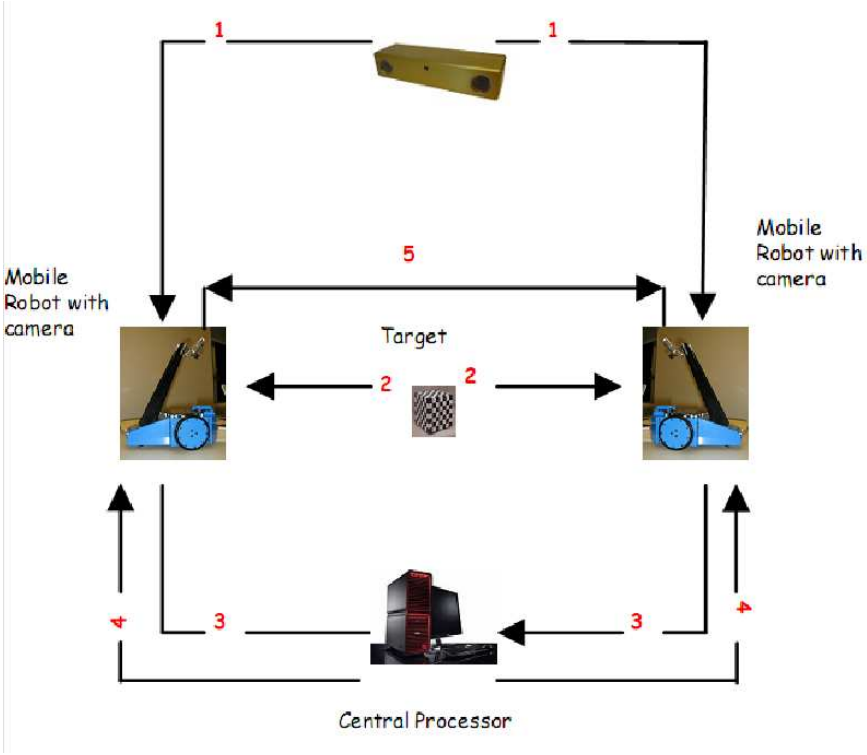

Figure 3.11: Experimental setup

position to move to in the next epoch was estimated. This was done by calculating the distances of the robots to the estimated position of the target and calculating $\angle r_1 t r_2$ and $\angle r_1 t r_3$.

The uncertainty measure was calculated for each robot pair and the target (t) was assigned to the pair having minimum uncertainty. The experiment showed that the robots r_1 and r_2 were assigned to track target t since the uncertainty in estimation due to these robots was the lowest. The ground truth for the position of the target was obtained from the stereo camera. The errors between the estimated target position and the ground truth for the target were calculated for both the robot pairs. It was seen that the error in triangulation due to the robots r_1 and r_2 was less than that due to robots r_1 and r_3. The experimental results also clearly showed that the uncertainty due to r_1 and r_2 was less than that due to r_1 and r_3.

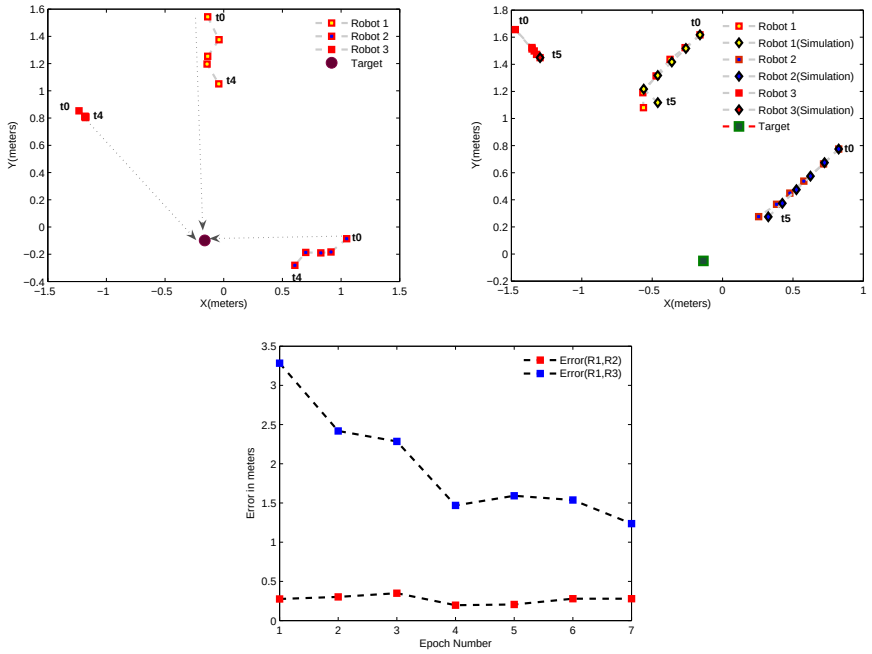

Figure 3.12: The leftmost figure shows the path for the robots r_1, r_2 and r_3 for the experiment where the target is stationary. The center figure shows the path of the robots from simulation and those that are obtained from experiments taking into account the error in robot motion. The rightmost figure shows the plot of the distance between the ground truth for the position of the target and the estimate of the target position obtained from triangulation over successive epochs for the two robot pairs.

The robots then moved to the calculated best position for the next epoch and the experiment was repeated using the new positions of the robots and the pictures taken from the webcams from the new positions.

3.7 Discussion

In this chapter, we studied the problem of designing motion-planning and sensor assignment strategies for tracking multiple targets with a mobile sensor network. We presented a distributed strategy for triangulation based tracking where two sensors merge their measurements in order to estimate the position of a target.

An important problem for future research is to incorporate target dynamics

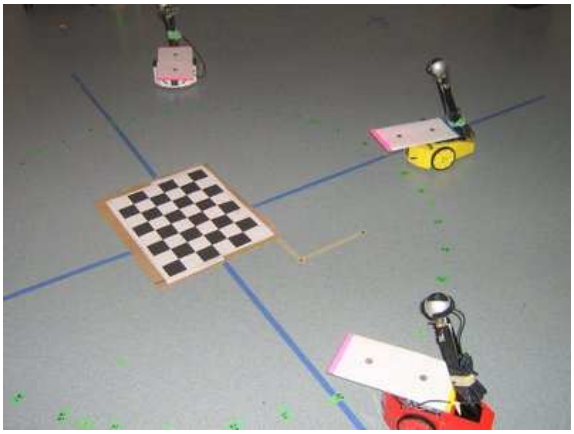

Figure 3.13: The experimental setup in which a network of three mobile robots with webcams are tracking a target (checkered board)

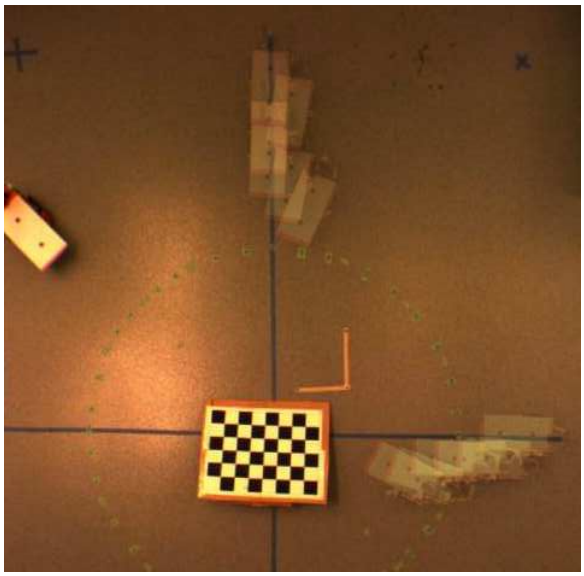

Figure 3.14: Snapshots of the robot positions at each epoch superimposed on one another. This figure corresponds exactly to the leftmost figure in Figure 3.12. The target used is the checkered board.

into the strategy. Currently, we treat the targets as static within an epoch. However, if accurate representations of the targets' motion are available, then the strategy can be optimized in the time dimension as well.

CHAPTER 4
Conclusions

The thesis studied localization in two different contexts in camera networks. The problem of localizing sensors in a static camera network was studied in Chapter 2. The accuracy of localization was observed to improve with the additional Belief Propagation step. Since Loopy Belief Propagation was used, guarantees cannot be given with respect to the convergence of the algorithm. An alternative message passing algorithm like the Tree Re-Weighted algorithm may be used in order to address convergence issues. Another interesting area of research is designing exploration strategies that can optimize the underlying graph on which message passing is run to improve the performance of the algorithm. Exploration strategies have been studied in different contexts as in [5] but not with the view of optimizing the performance.

An iterative heuristic algorithm was proposed in Chapter 3 to find the locally optimal configuration for a network of mobile cameras tracking multiple targets at any given time instant. A proof of concept experiment was conducted to demonstrate that the proposed algorithm is feasible in practice. Due to the large errors in the robots' (the mobile platform on which the cameras were mounted) motion, the cameras needed to be relocalized at the beginning of every iteration. Probabilistic algorithms that account for the errors in the robots' motion and those in the camera measurements can be used to alternatively solve the motion planning problem and improve its accuracy.

Camera networks are information rich; potentially leading to a lot of exciting applications. Some of the applications include 3D modeling, surveillance and target tracking. These require outputs from multiple cameras to be fused in order to obtain useful information introducing complex dependencies between the cameras. Also, due to the massive amounts of data collected by each camera, it becomes necessary to design algorithms that can be implemented in a distributed fashion. Further, mobility of the cameras adds its own unique challenges. This thesis is a first step in

addressing some of these challenges.

REFERENCES

[1] E.; Isler V. Kamath, S.; Meisner. Triangulation based multi target tracking with mobile sensor networks. *Robotics and Automation, 2007 IEEE International Conference on*, pages 3283–3288, 10-14 April 2007.

[2] Christopher Taylor, Ali Rahimi, Jonathan Bachrach, Howard Shrobe, and Anthony Grue. Simultaneous localization, calibration, and tracking in an ad hoc sensor network. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 27–33, New York, NY, USA, 2006. ACM.

[3] Xiaotao Liu, Purushottam Kulkarni, Prashant Shenoy, and Deepak Ganesan. Snapshot: A self-calibration protocol for camera sensor networks. *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–10, 1-5 Oct. 2006.

[4] W.E. Mantzel, Choi Hyeokho, and R.G. Baraniuk. Distributed camera network localization. *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, 2:1381–1386 Vol.2, 7-10 Nov. 2004.

[5] Ioannis Rekleitis, David Meger, and Gregory Dudek. Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems*, 54(11):921–932, November 2006.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[7] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[8] Jean-Yves Bouguet. Camera calibration toolbox for matlab.

[9] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, September 2005.

[10] P.; Fearnhead P. Carpenter, J.; Clifford. Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings* -, 146(1):2–7, Feb 1999.

[11] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–475, 1999.

[12] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *EURASIP JASP*, 4(378-391), 2003.

[13] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: An information directed approach. *Proc. of the IEEE*, 91(8), 2003.

[14] J. Liu, M. Chu, J. Liu, J. Reich, and F. Zhao. Distributed state representation for tracking problems in sensor networks. In *Proc. IPSN 2004*, pages 234–242. ACM Press.

[15] E. Ertin, J. W. Fisher III, and L. C. Potter. Maximum mutual information principle for dynamic sensor query problems.

[16] H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *Proc. IPSN 2004*, pages 36–45. ACM Press.

[17] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proc. First intl'l conf. on Embedded networked sensor systems*, pages 150–161. ACM Press, 2003.

[18] V. Isler, J. Spletzer, S. Khanna, and C.J. Taylor. Target tracking in sensor networks: the focus of attention problem. *Computer Vision and Image Understanding Jrnl.*, 100:225–246, October 2005.

[19] V. Isler and R. Bajcsy. The sensor selection problem for bounded uncertainty sensing models. *IEEE Tran. Automation Science and Engineering*, 2006.

[20] L. Doherty, K. S. J. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *Proc. IEEE Infocom*, 2001.

[21] A. Howard, Maja J. Matarić, and G. S. Sukhatme. Cooperative relative localization for mobile robot teams: An ego-centric approach. In *Proc. Naval Research Laboratory Workshop on Multi-Robot Systems*, Washington, D.C., Mar 2003.

[22] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, and J.P. Ostrowski. Cooperative localization and control for multi-robot manipulation. In *Proc. IROS 2001.*

[23] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[24] J. Spletzer and C. Taylor. Dynamic sensor planning and control for optimally tracking targets. *IJRR*, 22(1):7–20, 2003.

[25] S. Aranda, S. Martínez, and F. Bullo. On optimal sensor placement and motion coordination for target tracking. In *Proc. ICRA*, Barcelona, Spain, April 2005.

[26] V. Isler. Placement and distributed deployment of sensor teams for triangulation based localization. In *Proc. IEEE ICRA*, 2006.

[27] Mark A. Paskin, Carlos E. Guestrin, and Jim McFadden. A robust architecture for inference in sensor networks. In *IPSN 2005*.

[28] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. *IEEE Jrnl. of Selected Areas in Communication*, 2005.

[29] Rahul Biswas, Sebastian Thrun, and Leonidas J. Guibas. A probabilistic approach to inference with limited information in sensor networks. In *IPSN 2004*, pages 269–276, New York, NY, USA. ACM Press.

[30] L. Chen, M. Wainwright, M. Cetin, and A. Willsky. Multitarget-multisensor data association using the tree-reweighted max-product algorithm. In *Proc. SPIE Aerosense conf.*, Orlando, FL, March 2003.

[31] A. Kelly. Precision dilution in mobile robot position estimation. In *Intelligent Autonomous Systems*, Amsterdam, Holland, 2003.

[32] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps and non-approximability-towards tight results. In *FOCS '95: Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 422, Washington, DC, USA, 1995. IEEE Computer Society.

[33] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation via pseudo-moment matching. In *Proc. AISTATS*, 2003.