Power-Aware Multisensor-Multitarget Tracking on the Micro-AMPS Platform

by

Kevin E. Atkinson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2003

Certified by ....
Anantha P. Chandrakasan
Associate Professor
Thesis Supervisor

Certified by .....
Charles E. Rohrs
Principal Research Scientist
~~Thesis Supervisor~~

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Power-Aware Multisensor-Multitarget Tracking on the Micro-AMPS Platform

by

Kevin E. Atkinson

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2003, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents the design and implementation of an energy-efficient, multiple target tracking system developed to run in real-time on the MIT μAMPS wireless sensor nodes. The system integrates line-of-bearing data obtained from a field of acoustic sensors to provide an optimal position and velocity estimate of a moving sound source such as a vehicle engine. The Kalman filter and probabilistic data association techniques used in this application provide a significant improvement in tracking performance versus basic triangulation. These algorithms are combined with harmonic line association, a frequency-domain detection algorithm, to create a complete, real-time target detection and tracking system. Functionality and performance are verified through simulation and experiment using a real-time implementation.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Associate Professor

Thesis Supervisor: Charles E. Rohrs
Title: Principal Research Scientist

Acknowledgments

I would like to thank my advisor, Professor Anantha Chandrakasan, for giving me the opportunity to work on this exciting and challenging project as part of the μAMPS team and for funding this work. His encouragement and motivation have been instrumental to the success of this project. I would also like to thank CTA and BAE for their funding support, and ARL for providing harmonic line association code.

This thesis would also not have been possible without the help of my co-advisor, Charlie Rohrs. He has done an excellent job of teaching me the Kalman filter and data association algorithms that form the core of this project, and provided a very helpful critique of this manuscript.

I would like to thank the members of Prof. Chandrakasan's research group and the μAMPS team for providing a fun and intellectually challenging atmosphere to conduct research. In particular, I would like to acknowledge Rex Min, Nathan Ickes, and Fred Lee for invaluable debugging assistance and help in understanding the μAMPS node, and Alice Wang for helping me to understand the beamforming and HLA algorithms.

I would also like to thank my parents, Edward and Anne Atkinson, and my brother, Craig Atkinson, for their support during this project and throughout my years at MIT.

Finally, I would like to thank Veena Thomas for always being there for me, and for her expert editing assistance with this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless sensor networking is an important emerging technology with a wide variety of applications, including target tracking, environmental sensing, medical monitoring, machine diagnosis, and security systems. Networks of distributed microsensors offer several advantages over the traditional approach using small numbers of macrosensors. These advantages include fault tolerance, improved sensing resolution, and potentially lower cost. This thesis focuses on the vehicle tracking application, and presents the design and implementation of a real-time tracking system to run on the MIT μAMPS wireless sensor nodes.

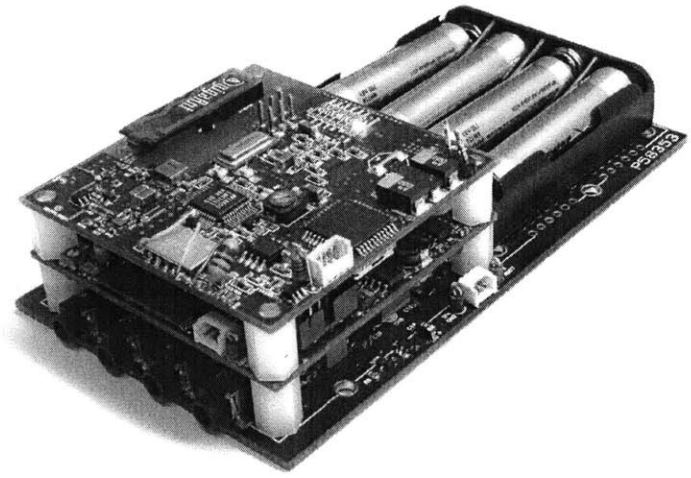

Figure 1-1: uAMPS Wireless Sensor Node

1.1 The uAMPS Wireless Sensor Node

The μAMPS sensor node [11] measures approximately 2 inches by 4 inches and consists of three stackable boards. The lower board contains the power supply, four acoustic sensor inputs, and the A/D converters. The node is currently powered by 4 AA batteries.

The middle board contains the Intel StrongARM CPU and its associated components, including SRAM and flash ROM chips. The StrongARM CPU allows dynamic voltage and clock frequency scaling to match the power consumption of the processor to the performance demands of a particular application. The clock speed can be set from 59 MHz to 221 MHz, and the voltage scales from 0.9 volts to 2.00 volts.

The top board holds a Bluetooth-based 2.4 GHz radio. It operates at a raw data rate of 1 Mbps, but Manchester encoding reduces the operational throughput to 500 Kbps. The power amplifier of the radio allows six different output power settings, ranging from 0 dBm to 20 dBm. The power consumption of the radio can thus also scale to match the range and reliability requirements of a particular application. The radio board also contains an FPGA to interface between the processor and the radio.

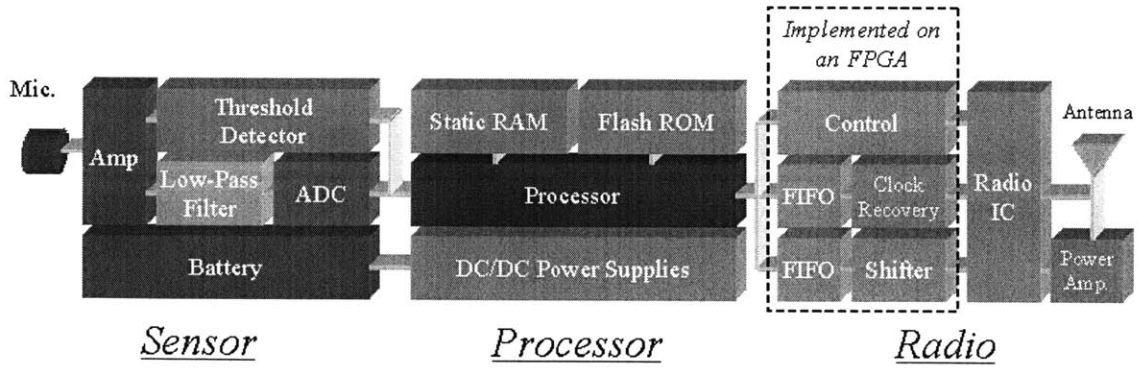

Figure 1-2: Block Diagram of uAMPS Node

1.2 Tracking Problem Formulation

The acoustic target tracking problem to be addressed is formulated as follows. Each node has three microphones attached to its acoustic sensor inputs. The microphones are arrayed at 120 degree angles from each other to form an equilateral triangle. See Figure 3 for an overview of the setup. The node then runs a beamforming algorithm, described in Section 2.5, to analyze inputs from the three microphones and compute a line-of-bearing to the most likely sound source.

The tracking application will consist of a field of these sensor nodes each communicating their line-of-bearing to a local base station. The question then becomes how to integrate the data to produce the most accurate estimate of the target position.

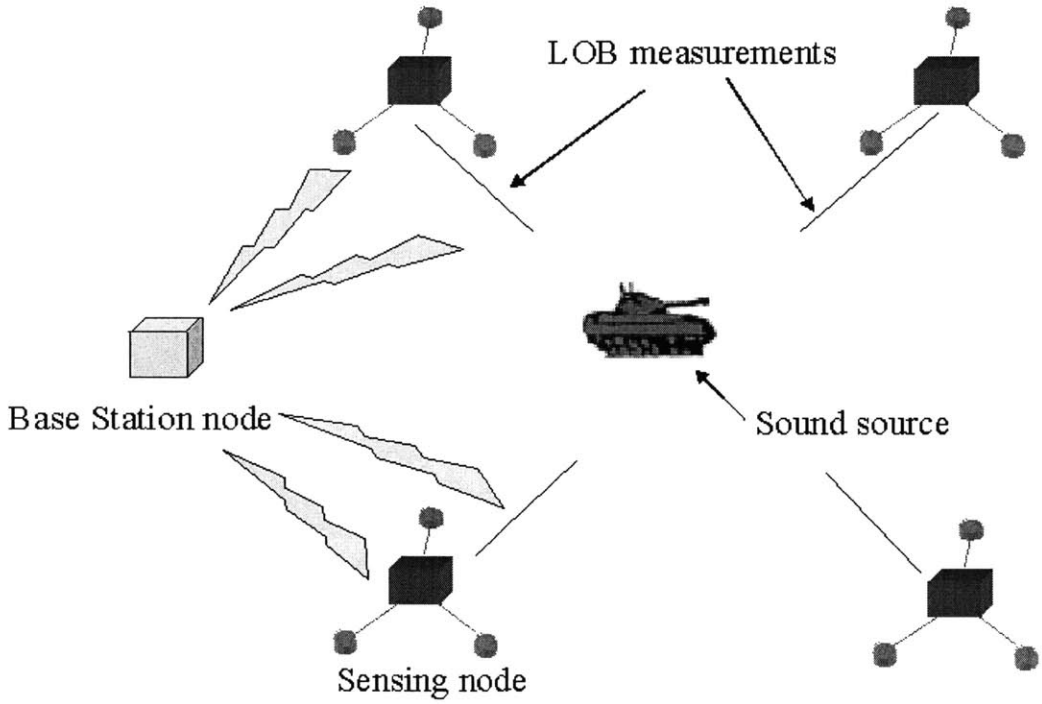

Figure 1-3: Tracking Scenario Overview

1.3 Approach

The line-of-bearing measurements produced by the beamforming algorithm are quite noisy in practice and simple triangulation would yield poor tracking performance. To

achieve superior estimation accuracy we have implemented algorithms based on the Kalman filter, a fundamental technique of estimation theory.

The Kalman filter has two main advantages. First, it incorporates information regarding the statistical properties of both the target's motion and the measurements available, allowing for improved performance in the presence of noise. Second, it produces not only the estimate of the target's state, but also probabilistic information regarding the accuracy of the estimate (covariance matrix) [2].

The covariance matrix, along with other probabilistic information computed by the Kalman filter serves a variety of purposes. For example, it can be used to validate measurements and reject those that are clearly spurious by computing the probability a particular measurement came from the target [1]. Monitoring the covariance matrix can also indicate which sensors are providing useful information to localize the target, and allow for unneeded sensors to be shut down to save power.

The second phase of the project involved extending the system to simultaneously track multiple targets. In this scenario the main challenge involves determining the measurement to target associations. The technique used is known as the Joint Probabilistic Data Association filter (JPDAF), and computes the probability that each measurement validated for a target actually originated from that target. The state estimate is then updated with an appropriately weighted combination of all validated measurements.

In addition to the estimation filters described above, a frequency-domain detection algorithm known as Harmonic Line Association (HLA) [13] [12] has been implemented to run in real-time on the sensing nodes. This algorithm analyzes the spectrum of the microphone signal looking for harmonic signatures signifying the presence of vehicle engine sounds. It then provides information regarding the likelihood a real target is present versus a false alarm caused by spurious noise or reflections.

1.4 Thesis Organization

The remainder of this paper is organized as follows. Chapter 2 presents an overview of related work and alternative approaches to the Kalman filter-based system used for this project. Chapter 3 describes the basic Kalman filter, and discusses some of the challenges involved in applying it to this specific application. In Chapter 4, the JPDAF multiple-target tracking extension is described. Chapter 5 discusses the implementation of the HLA frequency-domain detection algorithm. Finally, Chapter 6 presents conclusions and ideas for future work, including power-saving enhancements using the covariance information produced by the filter.

Chapter 2

Previous and Related Work

This chapter describes other research work related to this project, including the beamforming algorithm used to generate the line-of-bearing measurements and some alternatives to the Kalman filter-based system implemented for our application. Some additional improvements that can be made to the Kalman filter are also described.

2.1 Information Utility and Grid-Based Distributions

Work done for the CoSense project at PARC described in [14] proposes an interesting method of using information theory to determine which sensors should be involved in a given sensing task. The authors have developed a closed-form expression for the expected utility of adding the measurement for a particular sensor, given its location and characteristics. Their system uses a distributed algorithm that at each time step passes the target state estimate and distribution information to the node expected to produce the largest increase in the estimation accuracy. This system is similar to our method for reducing power consumption by shutting down unneeded sensors presented in Section 5.3.

The system in [14] also differs from our application in that it uses a grid-based representation for the probability distribution of the state estimate. In contrast, our

work uses a parameterized distribution, i.e. the state is assumed to have a Gaussian distribution and thus characterized entirely by its mean and covariance matrix. The grid representation can capture non-Gaussian distributions, and strikes a balance between the simple but restrictive parametric representation and the most general approach of approximating an arbitrary distribution with a very large set of discrete samples. The CoSense system offers the flexibility to handle non-Gaussian distributions, but the extra computational complexity required to compute the information utility and evaluate the estimate distribution over the grid points makes it unsuitable for implementation on our low-power sensor nodes.

2.2 Particle Filters

As mentioned above, the Kalman filter-based techniques used in our application model the target state probability distribution as Gaussian and thus characterized entirely by its mean and covariance. This technique offers the lowest complexity but is the most restrictive in its assumption that the state can be modeled as Gaussian. At the other end of the spectrum is the particle filter approach described in [4][3][5]. The particle filter models the state distribution as a large number of discrete particles, allowing it to capture an arbitrary probability distribution. This flexibility comes at the cost of increased complexity, however, in both computation and the memory required to store the larger state representation.

2.3 Neural Network Approach

The Multiple Elastic Models (MEM) algorithm described in [10] proposes an improved neural network method for solving the passive tracking problem composed of bearings-only sensors. Traditional neural network approaches involve searching over the space of all intersection points between all angles produced by the sensors, and thus do not scale well to large numbers of sensors. In addition, neural networks identify a solution by minimizing a specific objective function, and traditional neural networks are prone

to becoming trapped in local minima before they can converge to a correct solution.

The MEM system attempts to address these shortcomings using a self-organizing neural network adapted to a dynamic tracking scenario. It consists of a number of neurons, each representing a point in the surveillance space, that attempt to converge to the target locations. The MEM algorithm also introduces a parameter indicating the accuracy of a particular neuron's estimate, which is updated dynamically and enables the neuron to escape from a poor local minimum. Simulations are presented in [10] showing the effectiveness of MEM at tracking up to 20 moving targets using three bearings-only sensors. It is not clear, however, whether this system will be as robust to noisy measurements and false alarms as the probabilistic data association techniques described in Chapter 4. Nonetheless, the neural network technique offers potential advantages, particularly in its ability to dynamically adapt to changing numbers of targets by using extra neurons that continuously search for new targets.

2.4 Extended Kalman Filter Improvements

As will be discussed in Section 3.4, one of the problems in applying the Kalman filter to the bearings-only tracking scenario lies in the nonlinear relation between the measured angle and the Cartesian coordinate position of the target. The traditional solution, known as the Extended Kalman Filter (EKF), has been shown to exhibit a range and range-rate bias in its estimates due to a correlation between the filter gain and innovations sequence introduced by the linearization process [8]. A method for eliminating this bias by adjusting the a posteriori state update is presented in [7]. This improved EKF has been demonstrated to be effective in situations where the original EKF is unsuited, particularly cases involving high measurement noises and maneuvering targets.

2.5 Beamforming Algorithm

The line-of-bearing (LOB) measurements computed by the sensor nodes are obtained using a time-domain delay-and-sum beamforming algorithm [9]. This algorithm existed on the nodes prior to this project, however it has been slightly modified to output multiple angles for use with the multiple-target tracking filter described in Chapter 4.

The beamforming algorithm works with three microphones spaced 120 degrees apart at a known distance from each other. It initially considers the sound source as coming from one of a specific number of angles evenly spread around 360 degrees. This number is set to 36 in our system. While a smaller number of angles may be sufficient to distinguish a single dominant sound source, a higher resolution gain vector is needed to determine multiple angles to multiple potential targets as described in Section 2.5.1. For each of these 36 angles, the algorithm pre-computes and stores what the delay would be for each microphone relative to a fixed reference if the sound came from that angle.

Once a block of data is available for each microphone, the following steps are performed for each of the 36 angles. The signals from each microphone are shifted by their relative delay for the angle in question to align them in time with each other. The realigned values of the three microphones at each time point are then added together (Figure 2-1). If the angle used to align them is the correct angle to the sound source, the signals will constructively interfere to produce a large signal amplitude. For all other angles some samples will destructively interfere to produce a much smaller amplitude. The energy in this new signal is then computed to produce one number representing the likelihood the sound source originated from the angle in question. The final result is thus a gain vector with 36 entries such that the correct angle corresponds to the index of the largest value. The algorithm then interpolates between the angles using a weighted average of the two neighboring points.

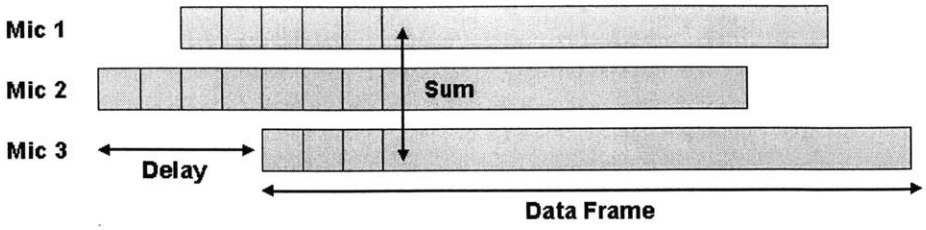

Figure 2-1: Delay and Sum Operation

2.5.1 Extension to Multiple Angles

Given we have a matrix of energy values corresponding to the sound originating from each of 36 angles, the second, third, etc. most likely source angles can in theory be identified by finding the next highest peaks in the energy vector. In practice, however, the second highest peak value is often a small peak near the main angle rather than one associated with a second sound source. Refer to Figure 2-2, which shows a sample plot of this final gain vector with two targets present. One is at an angle of 240 degrees and the other is at approximately 80 degrees. As can be seen in the figure, simply choosing the peak with the second largest value would result in assuming the second sound source is at 190 degrees, which is not correct. A better method is to pick the next peak with the largest value relative to its neighbors. This is done by passing the vector through the following filter:

```
for(i=1;i<NumAngles-1;i++) {
  peaks[i]=((float)Gain[i])/(float)((Gain[i-1]+Gain[i+1])>>1);
}
```

This code converts the gain vector to a peaks vector showing the relative magnitude at each angle compared with its two neighbors. The second largest value in this peak vector now correctly corresponds to the second sound source at 80 degrees. It must be noted, however, that this technique does diminish the chances of finding two true sources that are very close to each other.

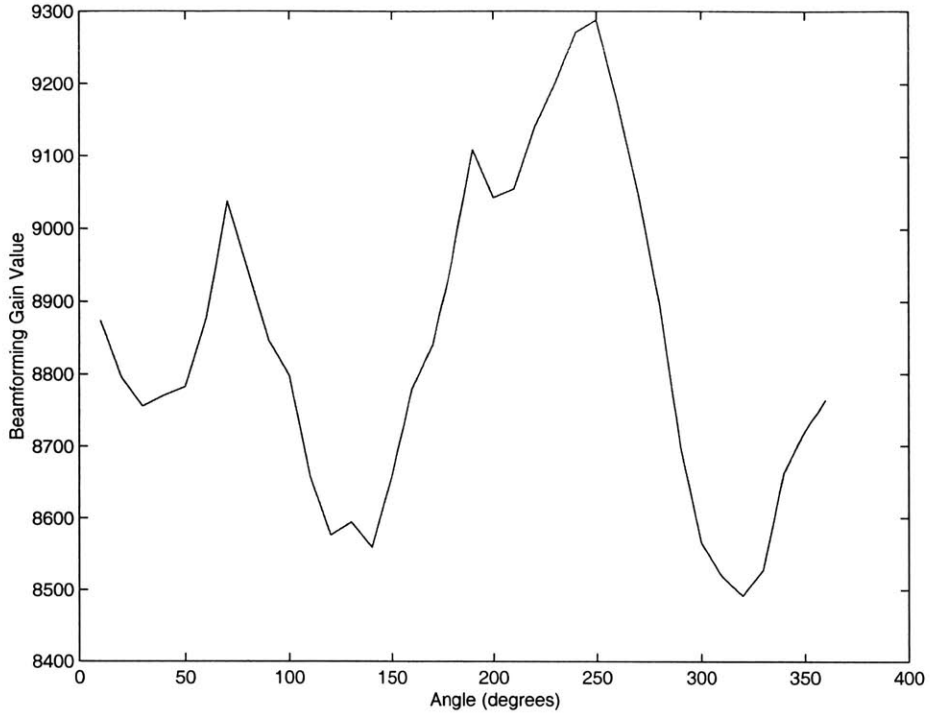

Figure 2-2: Final Gain Vector With Two Targets Present

2.5.2 Statistical Properties

The Kalman filter algorithms used in this application make the assumption that the LOB measurements come from a Gaussian distribution around the correct angle. To test this assumption we have measured the angle distribution resulting from a single sound source. Figure 2-3 gives the results. Attempting to simply fit a Gaussian distribution directly to the resulting angles by matching the mean and variance results in the wide Gaussian indicated by the solid line in the figure. The wide Gaussian does not accurately match the measurement distribution, as its variance is forced to be unnaturally large to account for outlier measurements resulting from reflections or other environmental noise. If these outliers are removed by eliminating the angles outside the range 160 to 200 degrees, then the result has a mean and variance corresponding to the thin dashed line Gaussian in the figure. This distribution is a good fit to the measured data, suggesting that if we can eliminate outliers the resulting measurements will fit the Gaussian assumption of the Kalman filter. The outliers

will be identified and removed using measurement validation techniques discussed in Section 4.1.

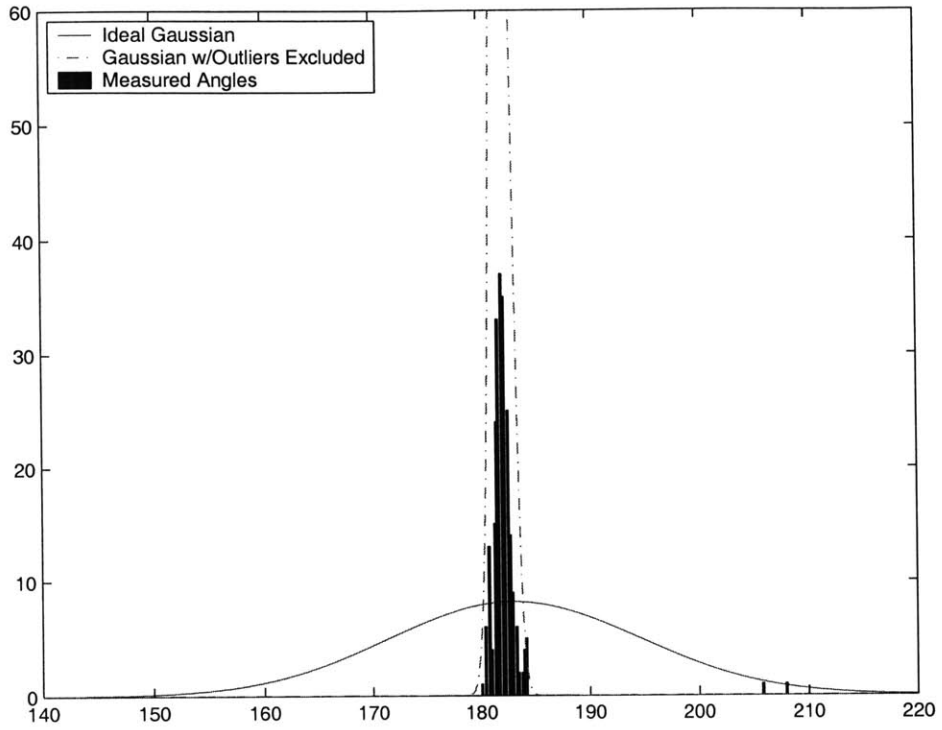

Figure 2-3: Distribution of Beamforming Results

2.6 Harmonic Line Association

In addition to the Kalman filter-based tracking algorithms, we have implemented a frequency-domain target detection/identification algorithm known as Harmonic Line Association (HLA) [13]. This algorithm operates on the principle that vehicle engines produce a harmonic signature, while most false alarms would not. The HLA algorithm was obtained as MATLAB code from the Army Research Labs, and for this project has been converted from MATLAB to C and modified to run in real-time on the μAMPS sensor node.

The HLA operates on a single channel of acoustic data. Samples from one microphone are buffered into one-second blocks and then passed to the HLA procedure. The data block is then converted to the frequency domain by a FFT routine. The power

spectrum is then passed to a peak-picking function to identify potential fundamental frequencies and harmonics. After the peaks are determined, the HLA procedure analyzes the power contained in each harmonic line set to determine which sets are the most likely to originate from a vehicle engine target [13].

2.7 Differences in Approach

Our approach differs from that of Sections 2.1, 2.2, and 2.3 in that we use a pure Kalman filter-based technique that models the target state esitmate with a Gaussian distribution. This approach achieves lower computational complexity suitable for real-time implementation at the cost of losing flexibility to accurately arbitrary distributions. In contrast to previous work, our system focuses on the challenges involved in designing and implementing a real-time tracking system using power-constrained sensor nodes. We also explore methods for utilizing the probability information computed by the Kalman filter to lower the power consumption of the overall system.

Chapter 3

The Kalman Filter

The Kalman filter is a fundamental technique for estimating the state of a dynamic linear system observed with noisy measurements. If the assumptions of the Kalman filter, discussed below, are satisfied then the filter is known to provide the optimal state estimate for the system. In our application the state to be estimated consists of the target's position and velocity. The main advantage of the Kalman filter lies in its ability to incorporate the target's previous estimate, expected behavior, and measurement data appropriately weighted by probabilistic information regarding the accuracy of each component. Refer to [2] for more detail and derivations related to the material presented in this chapter.

3.1 Setup

The evolution function of the system state is assumed linear and known, described by the discrete-time difference equation

$$x(k+1) = F \cdot x(k) + G \cdot v(k) \tag{3.1}$$

where $x(k)$ is the n_x dimensional state vector at time k, F is the known constant system evolution matrix, and v is a sequence of zero-mean white Gaussian noise

vectors with covariance $Q = \begin{bmatrix} q_x & 0 \\ 0 & q_y \end{bmatrix}$ shaped by the matrix G. $v(k)$ is known as the process noise, and models disturbances in the state evolution of the system.

The measurement vector $z(k)$ is described by the equation

$$z(k) = H(k) \cdot x(k) + w(k) \tag{3.2}$$

where $w(k)$ is the sequence of zero-mean white Gaussian measurement noise, with diagonal covariance R.

3.2 Modeling Assumptions

The assumptions underlying the Kalman filter are as follows:

- The state evolution is driven by additive zero-mean Gaussian noise with known covariance.
- The measurements are known *linear* functions of the state with additive white zero-mean Gaussian noise with known covariance.
- The initial state is a random variable with known mean and covariance.
- The initial error and noises are mutually uncorrelated.

It can be shown that if the above assumptions hold, the Kalman filter provides the optimal minimum-mean-squared-error (MMSE) estimate of the state variable. If the initial error and noises are not Gaussian, but still have known means and covariances, then the Kalman filter provides the optimal *linear* estimate. These assumptions lead to several challenges in the use of the Kalman filter in our tracking scenario. One limitation is the requirement that the measurements be a linear function of the state. Since we have nonlinear line-of-bearing measurements, they will have to be linearized as described in Section 3.4. Another limitation is the assumption that the state evolution is driven by additive Gaussian noise. This noise represents unknown

acceleration in our model, and thus cannot necessarily be accurately modeled as Gaussian with known covariance. This issue is addressed in Section 3.5.

3.3 Algorithm Overview

The basic Kalman filter algorithm consists of a two-step predict/update cycle. See Figure 3-1[1] for a flowchart of one iteration. First, the predicted state estimate is computed using the known system evolution matrix.

$$x(k+1|k) = F \cdot x(k|k) \tag{3.3}$$

where $x(k+1|k)$ denotes the predicted value of x at time $k+1$ given all information available through time k. The covariance of the predicted estimate is given by

$$P(k+1|k) = F \cdot P(k|k) \cdot F' + Q \tag{3.4}$$

Next, the measurements are predicted using the known measurement function and the predicted state estimate,

$$z(k+1|k) = H(k+1) \cdot x(k+1|k) \tag{3.5}$$

The covariance of the predicted measurement is given by

$$S(k+1) = H(k+1) \cdot P(k+1|k) \cdot H(k+1)' + R \tag{3.6}$$

Once the estimate and measurements have been predicted, their covariances are used to obtain the *filter gain* $W(k+1)$,

$$W(k+1) = P(k+1|k) \cdot H(k+1)' \cdot S(k+1)^{-1} \tag{3.7}$$

The state estimate and covariance are now updated using information from the mea-

[1]Figure based on p. 24 of [1].

surements. As the Kalman filter is a linear estimator, the updated state estimate is a linear combination of the predicted estimate and the difference between the actual and expected measurements, weighted by the filter gain. The difference

$$\nu(k+1) = z(k+1) - z(k+1|k) \tag{3.8}$$

is known as the *innovation* or *measurement residual.* S is thus referred to as the innovation covariance.

The updated state estimate is therefore given by

$$x(k+1|k+1) = x(k+1|k) + W(k+1) \cdot \nu(k+1) \tag{3.9}$$

The filter gain W thus determines the weighting of the measurements versus the predicted state estimate. One would therefore expect W to decrease as uncertainty regarding the measurements increases, and to increase as uncertainty regarding the predicted state increases. Indeed, this relationship is shown in (3.7), as W depends directly on $P(k+1|k)$, and inversely with $S(k+1)$.

Finally, the update estimate covariance is computed as

$$P(k+1|k+1) = P(k+1|k) - W(k+1) \cdot S(k+1) \cdot W(k+1)' \tag{3.10}$$

Note that in the previous expressions the values of $P(k+1|k)$, $S(k+1)$, $W(k+1)$, and $P(k+1|k+1)$ do not depend on the measurements $z(k)$. Thus in this formulation these quantities can be pre-computed offline. As we shall see in the next section, however, the use of line-of-bearing measurements requires linearization techniques that introduce a measurement-dependence into these quantities.

3.4 Extended Kalman Filter

The basic Kalman filter described above requires several modifications before it may be used for the line-of-bearing tracking problem in our application. First, the mea-

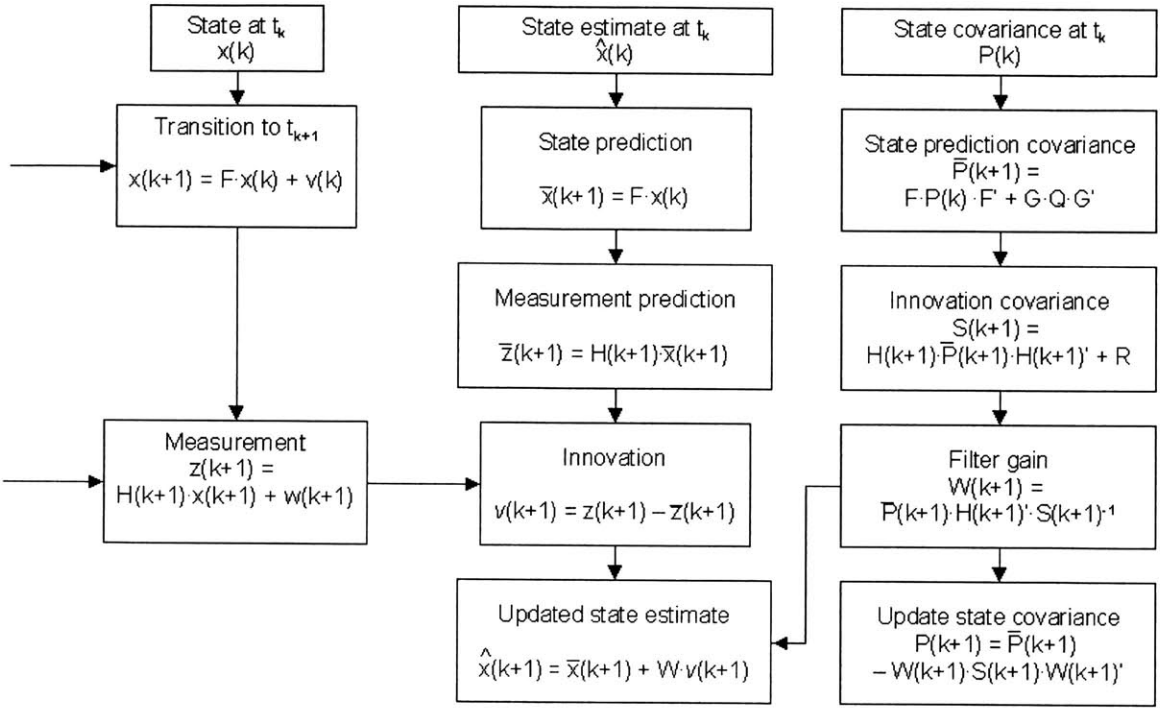

Figure 3-1: Basic Kalman Filter Flowchart

surement equation (3.2) assumes the measurements are a linear function of the state, i.e. H is a matrix with constant, known elements. We have only line-of-bearing measurements, however, which cannot be written in this form. As shown in Figure 3-2, we have measurements θ given by the nonlinear function

$$\theta = tan^{-1}\left[\frac{y - y_s}{x - x_s}\right] \tag{3.11}$$

where x and y give the position of the target in rectangular coordinates and x_s and y_s denote the sensor position.

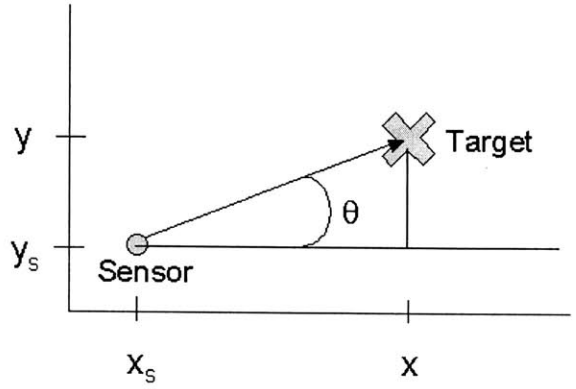

Figure 3-2: LOB Measurement

The method for handling nonlinear measurement functions is known as the Extended Kalman Filter, or EKF. It involves taking a first-order linearization of the measurement function using a Taylor series expansion. This linearization introduces several sources of error into the estimate. One error source results from discarding the higher-order terms in the Taylor expansion. This error can be mitigated by using a higher-order expansion, though in practice moving to a second-order expansion may introduce increased complexity for little gain. Second, in order for the linearization to be accurate it must be done around the true state of the target. However, the true target state is of course not known by the filter, so the estimated state is used instead. This substitution introduces an additional source of error.

3.5 Adaptive Filtering

Another aspect of the standard filter that poses a problem for our implementation is the requirement that the system evolution function, F, be known in advance. For a realistic vehicle tracking scenario the motion models of the vehicles are in general unknown. For example, a vehicle may stop or execute a maneuver (e.g. turn) at an unknown time. The simplest solution to this problem is to artificially increase the process noise parameter to account for variations in the target's motion model. A larger process noise allows the filter to track larger deviations from the expected motion, and thus adapt to situations such as a changing velocity. Changing the process noise presents a tradeoff, however, as making it larger allows for quick adaptation to unexpected maneuvers, but reduces tracking performance when the target is behaving as expected (i.e. moving with constant velocity). Conversely, a low process noise allows improved noise reduction and tracking performance when the target follows the model, but poor adaptation to maneuvers.

Given this process noise tradeoff, if one could detect the onset of a maneuver then the process noise could be switched to a high value to allow the filter to quickly adapt, then returned to a low value to provide improved noise reduction when the target has returned to expected behavior. A maneuver manifests itself as a larger than expected innovation, leading to the detection procedure described below. This technique, known as adaptive discrete process noise-level switching, offers improved tracking performance in practice with little additional complexity.

The maneuver detection is performed by monitoring a quantity known as the normalized innovation squared (NIS), given by

$$NIS = \nu(k)' \cdot S^{-1} \cdot \nu(k) \tag{3.12}$$

Under the linear Gaussian assumptions, the pdf of the NIS is chi-square distributed with n_z, the dimension of the measurement vector, degrees of freedom. If the NIS exceeds a certain value, the upcrossing threshold, then the process noise is switched to its high value. Once the NIS drops below a downcrossing threshold the process noise

is returned to its low value. The upcrossing threshold is chosen from the chi-square probability tables such that the probability of the NIS exceeding it under normal conditions is very small.

A more complex adaptation technique that offers higher performance is known as the Interacting Multiple Model estimator, or IMM [6]. This method can be thought of as running several Kalman filters in parallel, each with a different target motion model, then combining the results based on the probability that each model is correct. The IMM requires a non-negligible increase in complexity, however, and has not been implemented in our system.

3.6 Sequential vs. Block Processing of Measurements

The Kalman filter algorithm described above processes all measurements simultaneously using the n_z dimensional measurement vector, $z = [z_1(k), z_2(k), ..., z_{n_z}(k)]'$. If the measurement noise covariance matrix R is diagonal, meaning the measurement noise vector components are uncorrelated, then the state update can be computed by taking the measurement vector components one at a time. This technique is known as sequential measurement processing, and reduces complexity and memory requirements compared to the standard block processing. Sequential processing is a natural fit to our application, as each measurement comes from a different sensor and the measurement noises can be reasonably modeled as independent across sensors.

The sequential update algorithm begins with the same predicted estimate and covariance as the block case, now denoted as $x(k|k,0)$ and $P(k|k,0)$. The following sequence of calculations is then performed for $i = 1, ..., n_z$.

The innovation covariance is now the scalar quantity

$$s(k,i) = h_i(k)' \cdot P(k|k,i-1) \cdot h_i(k) + r_i(k) \tag{3.13}$$

where $h_i(k)$ is the row of $H(k)$ corresponding to sensor i, and $r_i(k)$ is the correspond-

ing measurement noise variance.

The filter gain is then computed as

$$W(k,i) = \frac{P(k|k,i-1) \cdot h_i(k)}{s(k,i)} \tag{3.14}$$

Note the matrix inversion has been replaced by a scalar division. The updated state estimate is now

$$x(k|k,i) = x(k|k,i-1) + W(k,i) \cdot [z_i(k) - h_i(k)' \cdot x(k|k,i-1)] \tag{3.15}$$

with covariance

$$P(k|k,i) = P(k|k,i-1) - W(k,i) \cdot h_i(k)' \cdot P(k|k,i-1) \tag{3.16}$$

For the ith update, $x(k|k,i-1)$ and $P(k|k,i-1)$ play the roles of predicted state update and covariance, respectively. This approach reduces the computational and memory requirements of the algorithm as no matrix sizes now depend on the dimension of the measurement vector, n_z. This dimension corresponds to the number of sensors used in our application, meaning the matrix sizes used in the algorithm system are now independent of the number of sensors. The memory requirements are thus constant with the number of sensor nodes, as opposed to linear in the block case. In addition, computational complexity now increases linearly with n_z as opposed to exponentially with block processing.

3.7 Real-Time Implementation

Based on the above algorithms, we have implemented a first-order Extended Kalman Filter using adaptive noise-level switching and sequential measurement processing to run in real-time on the StrongARM CPU of the μAMPS sensor node. The following sections describe the parameters used for implementation and present some tracking results showing the improvement over simple triangulation.

3.7.1 Filter Parameters

Our filter uses a constant-velocity motion model with process noise modeling any unknown acceleration. We thus have

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

and

$$G = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{3.18}$$

so the system evolves as in (3.1), where the state vector is $x(k) = [x, y, dx/dt, dy/dt]'$ and the vector $v(k) = [v_x(k), v_y(k)]'$ represents the white Gaussian process noise for each coordinate.

The process noise variance q is assumed the same for each coordinate and is switched dynamically between $q_l = 2$ and $q_h = 20$. The switching algorithm works by monitoring an exponential weighted moving average of the NIS (3.12),

$$NIS^{\alpha}(k) = \alpha NIS^{\alpha}(k-1) + NIS(k) \tag{3.19}$$

where $\alpha = 0.8$. The effective window length of (3.19) is the sum of the weights multiplying $NIS^{\alpha}(k)$, given by the geometric series

$$s_{\alpha} = 1 + \alpha + \alpha^2 + ... = \frac{1}{1-\alpha} \tag{3.20}$$

which yields an effective window length of 5 for $\alpha = 0.8$. One can assume as a first approximation that $NIS^{\alpha}(k)$ is chi-square distributed with $s_{\alpha} n_z$ degrees of freedom. The switching thresholds described in Section 3.5 are then determined from

the chi-square distribution tables. The up-crossing threshold has been set to 16.7, which has a 0.01% probability of being exceeded if everything in the model is correct. The down-crossing threshold is set to 2.67, which corresponds to a 75% probability that the high-noise model is no longer correct. This noise-level switching provides a good complexity/performance tradeoff by yielding the majority of the benefits of an adaptive filter without the extra complexity of the IMM estimator.

The standard deviation of the LOB measurements produced by the beamforming algorithm has been measured to be 1.45 degrees under ideal conditions (i.e. no multipath reflections). In practice, however, the LOB measurements do not fit a Gaussian distribution, but are more akin to a Gaussian with "fat tails." These tails correspond to the fact that measurements many standard deviations from the mean can be received quite frequently due to multipath reflections or other environmental noise. For this reason a higher measurement noise variance of $r = 0.05$ is used in the filter, which corresponds to a LOB standard deviation of 12.8 degrees.

3.7.2 Real-Time Display

The filter described above runs on a base station node, which then sends the results to a PC over a serial port to a Java program providing a visual display. Figure 3-3 shows a screenshot of the display tracking a single target with three sensor nodes.

Figure 3-4 gives an example of how the shape of the covariance matrix changes as measurements are received from different sensor positions. As shown in the figure, when measurements are only available along one dimension the covariance matrix quickly grows into an elongated ellipse along the corresponding axis.

3.7.3 Results

In this section we present real data traces from the real-time Kalman filter-based tracking system described above to provide a clear picture of the improvement over simple triangulation. The Java PC display mentioned in Section 3.7.2 is capable of logging all received LOB measurements and Kalman filter outputs for offline analysis

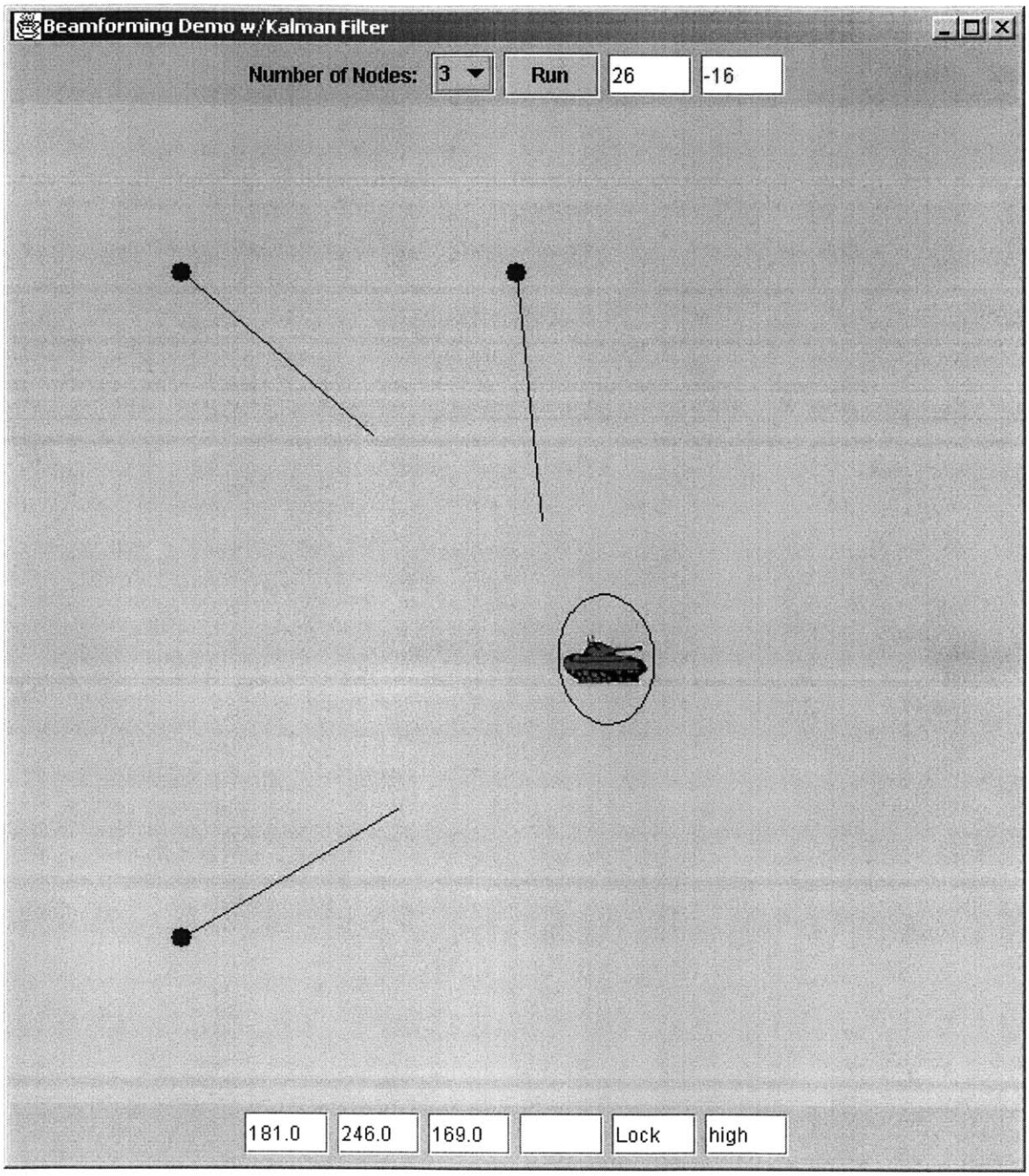

Figure 3-3: Real-Time Display

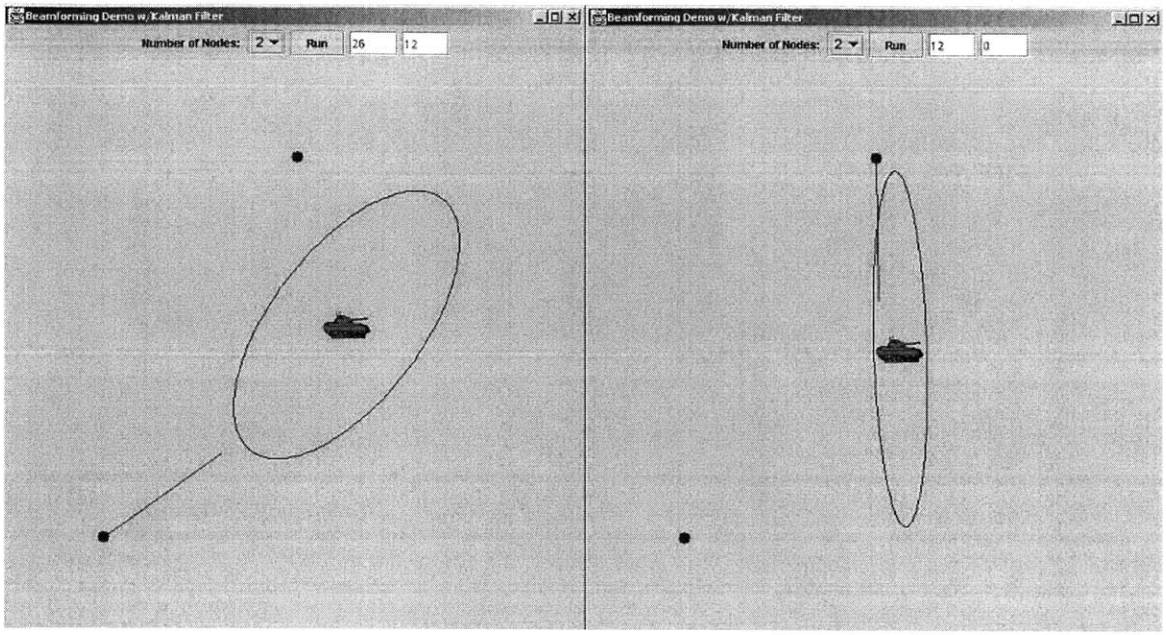

Figure 3-4: Covariance Shape: One Sensor at a Time

using MATLAB. For the test setup the target is a speaker/subwoofer combination playing the actual tank sound recorded by the μAMPS nodes at the Army Proving Ground in Aberdeen, Maryland. Two sensor nodes are used, positioned at the northwest and southwest corners of the surveillance area. The subwoofer "tank" is pushed slowly past the two sensors at a velocity of approximately 0.5 m/s.

For the first run the target is positioned in the southern center of the surveillance region and then moves in a straight line to the north. Figure 3-5 presents the results of the run and clearly demonstrates the advantages of the Kalman-filter based system. The positions computed by simple triangulation are not at all consistent with the motion of the target, while the filter estimated position tracks much closer. Also note that the true target position remains within the confidence region derived from the Kalman filter covariance matrix.

For the second run the target is initially positioned between the two sensors and then moves in a straight line to the east. Figure 3-6 presents the results. As shown in the figure, the y-coordinate of the initial position cannot be determined with any accuracy, as measurements are only available along one dimension. Once the target

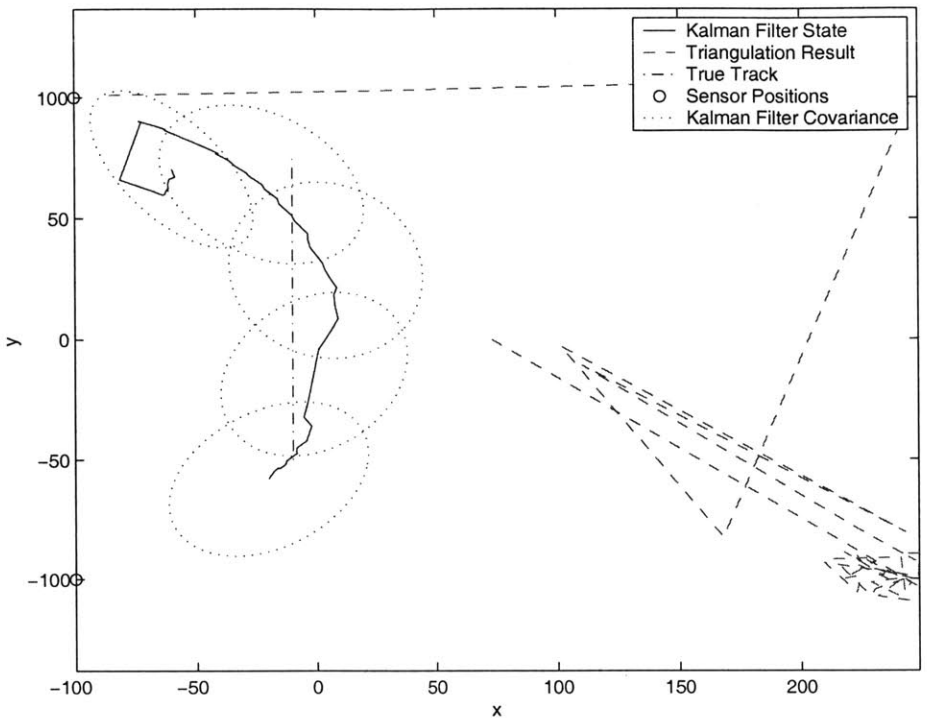

Figure 3-5: Tracking Results: Target Moving North

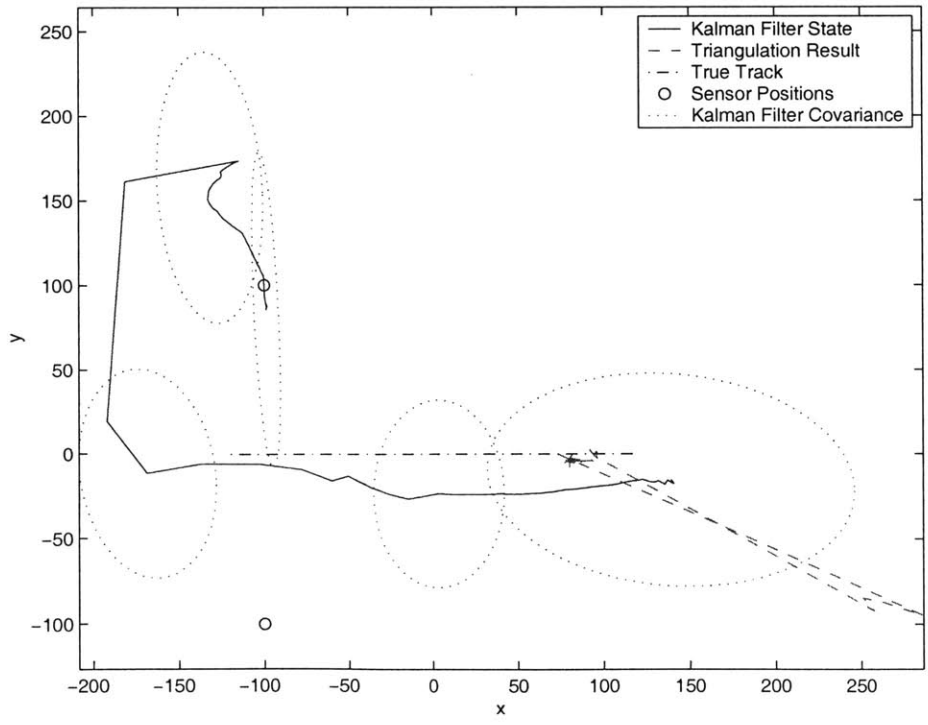

Figure 3-6: Tracking Results: Target Moving East

moves, however, the filter is able to converge to the correct y-position and track the motion of the target. Again the advantage of the Kalman filter is clearly visible, as the simple intersection of the two angles does not come close to accurately tracking the motion. Note in the beginning of the run on the left side of the graph no triangulation results are plotted since the angles do not intersect in this region. Only when the target begins to move eastward do the angles begin to intersect and the triangulation results appear on the graph. The Kalman filter, however, does not depend on the angles intersecting and is able to successfully track the eastward motion of the target.

3.8 Computational Requirements

The computational requirements of the Kalman filter increase as $O(n^3)$, where n is the dimension of the state vector $x(k)$ [2]. This complexity is due primarily to the matrix operations involved in the filter. As mentioned in Section 3.6, the sequential processing of measurements prevents the complexity from also increasing as $O(n^3)$ with the dimension of the measurement vector, reducing complexity growth to $O(n)$.

The current beamforming setup produces four angles per second that are transmitted to the base station from each sensor node for processing by the Kalman filter. The number of instructions required for the filter have been measured at 0.57 million instructions per iteration for a three sensor system using profiling tools on the SGI platform. Since the filter must run four iterations per second this corresponds to a processing requirement of 2.3 MIPS. On the StrongARM CPU of the μAMPS node running at 60 MHz, the filter requires 8 milliseconds to execute out of the 250 milliseconds available between sample sets. The processing requirements are thus well within the scope available on a low-power sensor node.

It should be noted that the Kalman filter runs on one base station node that collects measurements from several sensor nodes. The computational burden is thus spread unevenly throughout the system, leading to non-uniform power consumption. This effect can be mitigated by using a distributed algorithm that rotates the filter task among all nodes.

Chapter 4

Probabilistic Data Association

We now extend the system to track multiple targets simultaneously. The main challenge in this scenario is determining which measurement is associated with each target. The simplest approach would be to choose the measurement closest to the expected measurement for each target. This technique is called the nearest-neighbor standard filter (NNSF). As we shall show, however, this approach leads to very poor tracking performance in the presence of false measurements. A more powerful technique, known as the probabilistic data association filter (PDAF), involves calculating the probability that each measurement is associated with a particular target. The state update for a particular target is then carried out using an appropriately weighted combination of all measurements that could have originated from that target. The PDAF assumes the number of targets is known; in practice this number will be determined by a separate track initialization/maintenance model. The material presented in this chapter is described in more detail in [1], which also contains the derivations of the results presented here.

4.1 Measurement Validation

This scenario assumes multiple LOB measurements are available from each sensor node, some of which may be target originated and some of which will be false alarms. The first step in the PDAF involves determining which measurements could poten-

tially be associated with each target. This process, called measurement validation, makes use of the idea that, given a state estimate and information about its accuracy and the measurement accuracy, only certain measurement ranges have any significant probability of originating from the target. See Figure 4-1 for an illustration of this idea.

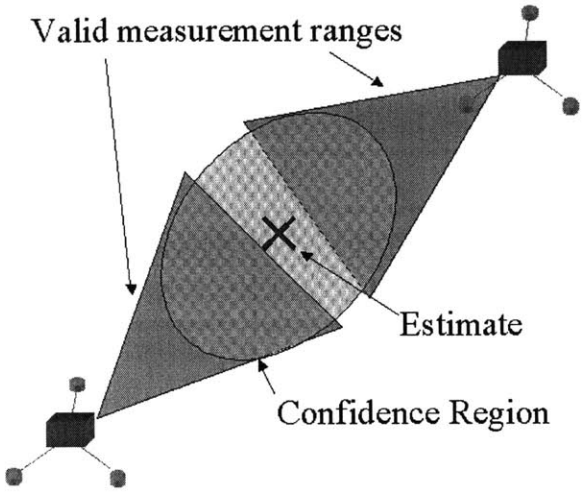

Figure 4-1: Measurement Validation

4.1.1 Validation Procedure

The validation region for each target is in general the elliptical region

$$V(k,\gamma) = \{z : [z - z(k|k-1)]' S(k)^{-1} [z - z(k|k-1)] \leq \gamma\} \tag{4.1}$$

In our case with sequential measurement processing the innovations and innovation covariance S are all scalar quantities and the validation corresponds to an arc of valid angles. Note the expression on the left-hand side of the inequality in (4.1) is the same as that given for the NIS in (3.12). The *gate threshold* γ is chosen from the chi-square tables to give the desired probability that a valid measurement is inside the gate.

4.1.2 Association Events

Once the measurements at each sensor have been validated as described in section 4.1, a validation matrix Ω is constructed to summarize the potential measurement associations for a given sensor. The number of rows is equal to the number of measurements available from the sensor, the number of columns is equal to the number of targets plus one, and the binary elements indicate if a particular measurement has been validated for a particular target. An example of a validation matrix for two targets and four measurements per sensor is

$$\Omega = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The leftmost column, called column 0, corresponds to "none of the targets", or a false alarm. Note column 0 contains all 1's as all measurements are potential false alarms. Here the two targets are "coupled" by the second measurement, which has been validated for both targets.

A joint association event θ is represented by an *event matrix*, which is a subset of the validation matrix Ω. For example, a possible association event taken from the validation matrix (4.2) is

$$\theta = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.3}$$

This event corresponds to measurement 1 originating from target 1, measurement 2 originating from target 2, and measurements 3 and 4 being false alarms. The set of all feasible joint association events is generated from Ω based on the following assumptions:

- a measurement can have only one source, i.e. θ can have only one '1' per row

- at most one measurement can originate from a target for a given sensor, i.e. θ can have only one '1' per column except for column 0 where the number of '1's, which is the number of false measurements, is not restricted

Once the set of feasible association events has been generated, the probability of each event is calculated for use in the JPDAF as described in section 4.2.2. It is convenient to define two binary detection indicator variables for use in later expressions. The first is defined as

$$\delta_t(\theta) = \sum_{j=1}^{m} w_{j,t}(\theta) \tag{4.4}$$

where $w_{j,t}(\theta)$ is the value of the association event matrix θ for measurement j and target t. $\delta_t(\theta)$ is called the target detection indicator since it indicates whether a measurement has been associated with target t in event θ.

Next is the measurement association indicator,

$$\tau_j(\theta) = \sum_{t=1}^{N_T} w_{j,t}(\theta) \tag{4.5}$$

where N_T is the known number of targets. $\tau_j(\theta)$ indicates whether measurement j is associated with a target in event θ. Note that using this definition the number of false measurements in event θ is

$$\phi(\theta) = \sum_{j=1}^{m}[1 - \tau_j(\theta)] \tag{4.6}$$

4.2 The Joint Probabilistic Data Association Filter

The Joint Probabilistic Data Association Filter (JPDAF) computes the measurement-to-target association probabilities jointly across all targets. This approach is necessary to distinguish between multiple targets when measurements from one target may appear inside the validation region of another target. (See Figure 4-2) Simply running a standard PDAF separately for each target is not sufficient as the PDAF assumes all

incorrect measurements are uniformly distributed false alarms. The joint probability calculations of the JPDAF are necessary to distinguish the case where a neighboring target gives rise to persistent interference.

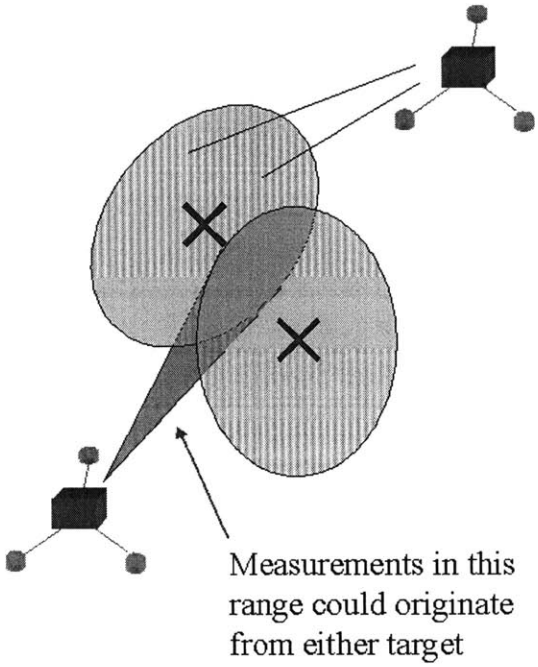

Figure 4-2: Measurements Valid for Multiple Targets

4.2.1 Assumptions

The assumptions behind the JPDAF are as follows:

- There is a known number of targets established in the presence of clutter (false alarms).
- Measurements from one target can fall in the validation region of a neighboring target, potentially over several sample times, leading to persistent interference.
- All past information is summarized by an appropriate sufficient statistic- the state estimates $x(k|k)$ and covariances $P(k|k)$ for each target.
- Each targets has a dynamic and measurement model as in the standard Kalman filter. The models for each target do not need to be the same.

4.2.2 Association Probabilities

Determining the association probabilities first requires computing the probability of each feasible association event $\theta(k)$. This probability can be written, using Bayes' formula, as

$$\begin{aligned} P\{\theta(k)|Z^k\} &= P\{\theta(k)|Z(k), m(k), Z^{k-1}\} \\ &= \frac{1}{c} p[Z(k)|\theta(k), m(k), Z^{k-1}] P\{\theta(k)|Z^{k-1}, m(k)\} \\ &= \frac{1}{c} p[Z(k)|\theta(k), m(k), Z^{k-1}] P\{\theta(k)|m(k)\} \end{aligned} \tag{4.7}$$

where c is the normalization constant and $m(k)$ is the number of measurements in the union of the validation regions at time k. The irrelevant conditioning term Z^{k-1} has been omitted from the second term in the last line of (4.7).

The first factor of (4.7), known as the likelihood function of the measurements, is given by

$$p[Z(k)|\theta(k), m(k), Z^{k-1}] = \prod_{j=1}^{m(k)} p[z_j(k)|\theta_{j,t_j}(k), Z^{k-1}] \tag{4.8}$$

The product form of (4.8) requires the assumption that the states of the targets conditioned on the past measurements are mutually independent.

The conditional pdf of a measurement given its origin is

$$p[z_j(k)|\theta_{j,t_j}(k), Z^{k-1}] = f_{t_j} \tag{4.9}$$

if $\tau_j[\theta(k)] = 1$, meaning the origin is assumed a target, or

$$p[z_j(k)|\theta_{j,t_j}(k), Z^{k-1}] = V^{-1} \tag{4.10}$$

if $\tau_j[\theta(k)] = 0$, corresponding to a uniform distribution for false measurements.

Here

$$f_{t_j}[z_j(k)] = N[z_j(k); z^{t_j}(k|k-1), S^{t_j}(k)] \tag{4.11}$$

and $z^{t_j}(k|k-1)$ is the predicted measurement for target t_j, with associated innovation

covariance $S^{t_j}(k)$. Measurements not associated with a target are assumed uniformly distributed in the surveillance region of volume V.

Using (4.9) and (4.10), the measurement likelihood function (4.8) can be rewritten as

$$p[Z(k)|\theta(k), m(k), Z^{k-1}] = V^{-\phi} \prod_j \{f_{t_j}[z_j(k)]\}^{\tau_j} \tag{4.12}$$

In (4.12) V^{-1} is raised to the power $\phi(\theta)$, the toal number of false measurements in event $\theta(k)$, while the indicators $\tau_j(\theta)$ select the measurement probability density functions according to their associations in event $\theta(k)$.

The second factor in (4.7), known as the prior probability of an event $\theta(k)$, is computed as follows. First, note that, given θ, the vector $\delta(\theta)$ of target detection indicators (4.4) is completely determined, as is the number ϕ of false measurements given by (4.6). Therefore,

$$P\{\theta(k)|m(k)\} = P\{\theta(k), \delta(\theta), \phi(\theta)|m(k)\} \tag{4.13}$$

which can be rewritten as

$$P\{\theta(k)|m(k)\} = P\{\theta(k)|\delta(\theta), \phi(\theta), m(k)\} P\{\delta(\theta), \phi(\theta)|m(k)\} \tag{4.14}$$

The first factor on the r.h.s. of (4.14) is determined as follows. Note that in event $\theta(k)$ the number of targets assumed detected is $m(k) - \phi$. The number of association events in which the same set of targets detected is thus given by the number of permutations of the m(k) measurements taken as $m(k) - \phi$, the number of targets to which a measurement is assigned under the same detection event. Assuming each such event is a priori equally likely, one has

$$P\{\theta(k)|\delta(\theta), phi(\theta), m(k)\} = \left(\frac{m(k)!}{\phi!}\right)^{-1} \tag{4.15}$$

The second factor on the r.h.s. of (4.14) is, assuming δ and ϕ are independent,

$$P\{\delta(\theta), \phi(\theta)|m(k)\} = \prod_t (P_D^t)^{\delta_t}(1 - P_D^t)^{1-\delta_t}\mu_F(\phi) \tag{4.16}$$

where P_D^t is the detection probability of target t and $\mu_F(\phi)$ is the prior pmf of the number of false measurements.

Combining (4.15) and (4.16) into (4.14) gives the prior probability of a joint association event as

$$P\{\theta(k)|m(k)\} = \frac{\phi!}{m(k)!}\mu_F(\phi)\prod_t (P_D^t)^{\delta_t}(1 - P_D^t)^{1-\delta_t} \tag{4.17}$$

Combining (4.12) and (4.17) into (4.7) yields the posterior probability of a joint association event as

$$P\{\theta(k)|Z^k\} = \frac{1}{c}\frac{\phi!}{m(k)!}\mu_F(\phi)V^{-\phi}\prod_j \{f_{t_j}[z_j(k)]\}^{\tau_j}\prod_t (P_D^t)^{\delta_t}(1 - P_D^t)^{1-\delta_t} \tag{4.18}$$

Note that ϕ, δ_t, and τ_j are all functions of the event $\theta(k)$ under consideration. The above still requires the pmf of the false measurements, which we will assume is simply a constant,

$$\mu_F(\phi) = \epsilon \tag{4.19}$$

Combining this with (4.18) and combining $m(k)!$, the constant ϵ, and the constant c into the constant c_2 yields the following expression for the association event probability:

$$P\{\theta(k)|Z^k\} = \frac{1}{c_2}\frac{\phi!}{V^\phi}\prod_j \{f_{t_j}[z_j(k)]\}^{\tau_j}\prod_t (P_D^t)^{\delta_t}(1 - P_D^t)^{1-\delta_t} \tag{4.20}$$

where c_2 is the appropriate normalization constant.

To obtain the marginal association probabilities needed for the state estimation in the next section, one simply adds the probabilities for each joint event in which the marginal event of interest occurs. The marginal probability that measurement j

is associated with target t is thus

$$\begin{aligned}\beta_{j,t} &= P\{\theta_{j,t}|Z^k\} \\ &= \sum_{\theta} P\{\theta|Z^k\} w_{j,t}(\theta) \\ &= \sum_{\theta:\theta_{j,t}\in\theta} P\{\theta|Z^k\} \end{aligned} \tag{4.21}$$

4.2.3 State Estimation

We denote the number of validated measurements at time k by $m(k)$. The *association events* $\theta_{j,t}(k)$ are then defined to signify that the j-th validated measurement is correct for the target t under consideration, for $j = 1, ..., m(k)$, or that none of the measurements are target originated for $j = 0$. These events are mutually exclusive and exhaustive, allowing the use of the total probability theorem to write the conditional mean of the state at time k as

$$\begin{aligned} x(k|k) &= E[x(k)|Z^k] \\ &= \sum_{j=0}^{m(k)} E[x(k)|\theta_{j,t}(k), Z^k] P\{\theta_{j,t}(k)|Z^k\} \\ &= \sum_{j=0}^{m(k)} x_j(k|k)\beta_{j,t}(k) \end{aligned} \tag{4.22}$$

where $x_j(k|k)$ is the updated state conditioned on the event that the j-th validated measurement is correct, and

$$\beta_{j,t}(k) = P\{\theta_{j,t}(k)|Z^k\} \tag{4.23}$$

is the conditional probability that the j-th validated measurement is correct, given by (4.21).

The prediction equations for the state $x(k|k-1)$ and measurement $z(k|k-1)$ are the same as the standard Kalman filter of Chapter 3. Likewise, the covariance of the predicted state $P(k|k-1)$ and the innovation covariance $S(k)$ come from the same equations as the standard filter.

The state estimate conditioned on measurement j being correct is given by the familiar Kalman filter equation

$$x_j(k|k) = x(k|k-1) + W(k)\nu_j(k) \quad j = 1, ..., m(k) \tag{4.24}$$

where the corresponding innovation is

$$\nu_j(k) = z_j(k) - z(k|k-1) \tag{4.25}$$

The gain W is the same as the standard Kalman filter 3.7, as conditioned on $\theta_{j,t}(k)$ there is no measurement uncertainty.

If j=0 or $m(k)$=0 then the state estimate is simply

$$x_0(k|k) = x(k|k-1) \tag{4.26}$$

If at least one validated measurement is available then the final state update is then given by

$$x(k|k) = x(k|k-1) + W(k)\nu_c(k) \tag{4.27}$$

where ν_c is the *combined innovation* given by

$$\nu_c(k) = \sum_{j=1}^{m(k)} \beta_{j,t}(k)\nu_j(k) \tag{4.28}$$

The state estimate is thus performed using all validated measurements, each weighted by its probability of being the correct target originated measurement. If no valid measurements are received then the state estimate is simply equal to the state prediction from the model.

4.2.4 Covariance Update

The covariance update equation consists of three components. The first is the predicted covariance $P(k|k-1)$. It appears weighted by $\beta_0(k)$, the probability that no

measurements are correct, as no update would occur if no correct measurement was received. The second term is the covariance updated with the correct measurement,

$$P^c(k|k) = P(k|k-1) - W(k)S(k)W(k)' \tag{4.29}$$

which is the familiar update equation from the standard filter. Another term is then added to the updated covariance to account for the uncertainty regarding which of the $m(k)$ validated measurements is correct. This term is known as the *spread of innovations* and is defined as

$$\bar{P}(k) = W(k)\left[\sum_{j=1}^{m(k)} \beta_{j,t}(k)\nu_j(k)\nu_j(k)' - \nu(k)\nu(k)'\right]W(k)' \tag{4.30}$$

This matrix is positive semidefinite and always acts to increase to resulting covariance update to reflect the measurement origin uncertainty.

The final covariance update is then given by

$$P(k|k) = \beta_0(k)P(k|k-1) + [1-\beta_0(k)]P^c(k|k) + \bar{P}(k) \tag{4.31}$$

The zero-measurement (prediction only) term is weighted by the probability that no measurements are correct, the standard update term is weighted by the probability that at least one measurement is correct, and the spread of innovations term is added to reflect the uncertainty over which measurement is correct.

4.3 Real-Time Implementation and Results

The JPDAF algorithm described above has been implemented in C to run in real-time on the CPU of the μAMPS node. Measurement processing is done sequentially as described in (3.6) to allow scaling to large numbers of sensors and reduce complexity and memory requirements. The algorithm process is described by the following pseudocode:

```
for each target {
```

```
   predict state estimate
   predict state covariance
}
for each sensor {
   if data received for this sensor this time period {
      for each target {
         predict measurements
         compute H by linearizing around predicted state
         compute innovation covariance
         validate measurements to create validation matrix
      }
      generate set of feasible association events
      compute probability of each event
      for each target {
         compute marginal association probabilities for each valid measurement
         compute combined innovation
         compute filter gain W
         update state estimate
         update covariance
      }
   }
}
```

4.3.1 Real-Time Display

The graphical java display for the PC has been modified to detect the presence of the JPDAF and display the multiple angles per sensor node and the multiple targets with covariance matrices. Figure 4-3 shows a screenshot of the system tracking two targets (speakers) using three angles per sensor. Note the covariance ellipse of the left target is elongated along the vertical axis, as it is only receiving measurements along that dimension. In contrast, the right target is receiving measurements along

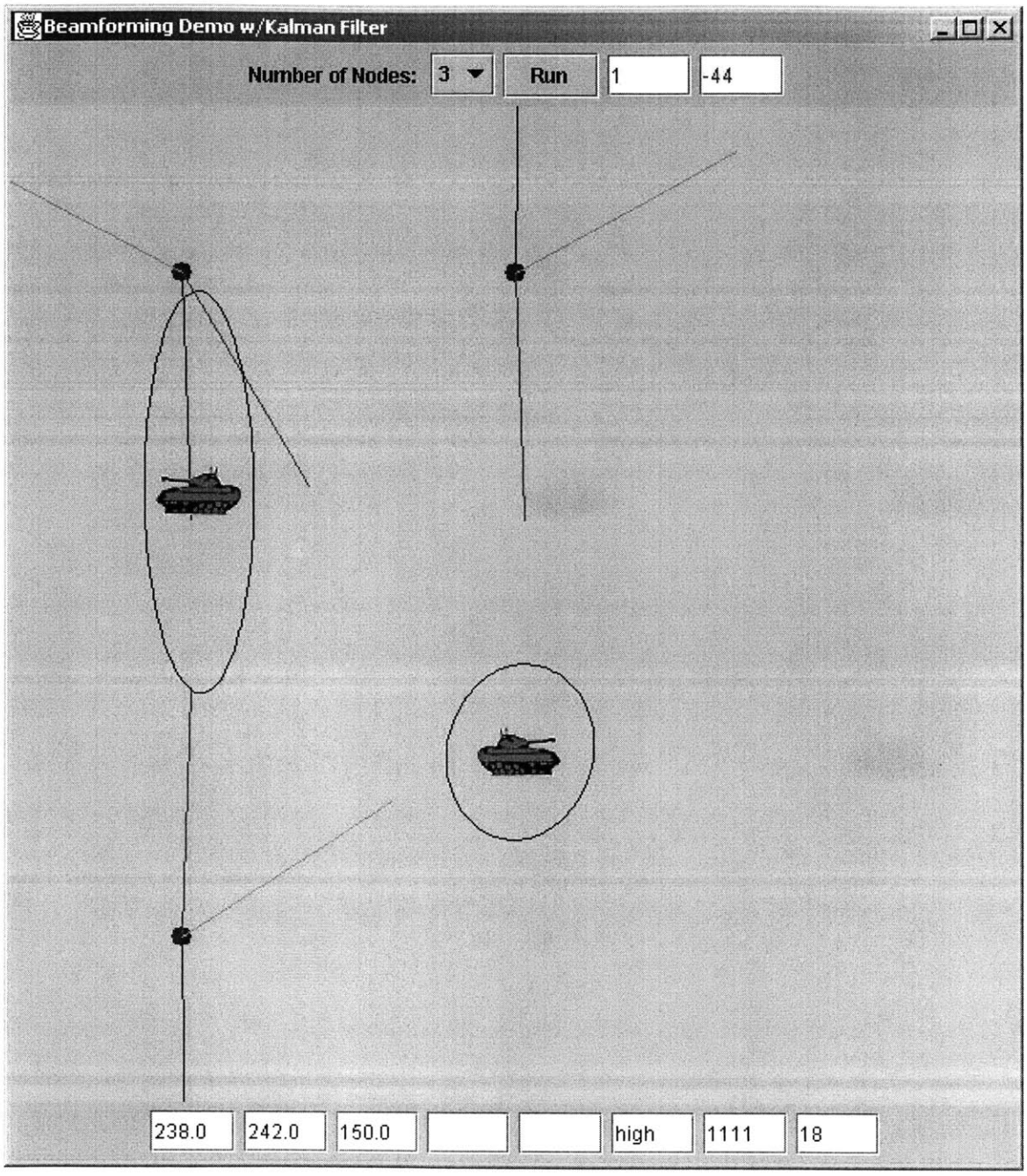

Figure 4-3: Real-Time Display Tracking Two Targets

both axes, so its covariance ellipse is close to circular.

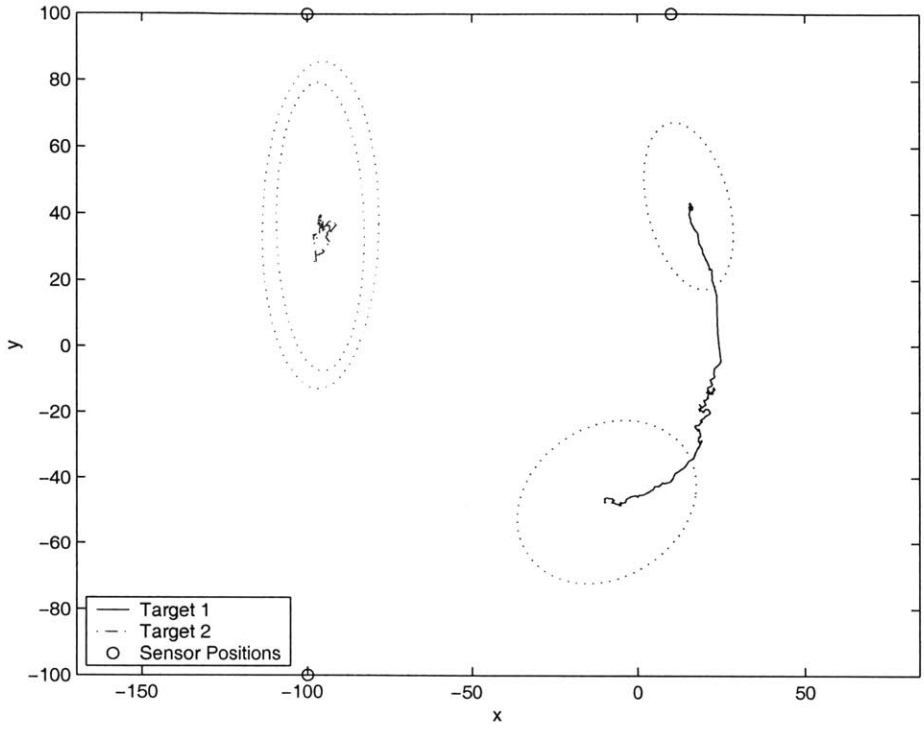

Figure 4-4: Sample Tracking Run: Two Targets

Figure 4-4 shows the results of a tracking run on the μAMPS hardware with three sensor nodes and two targets. Target 1 is moving straight up through the surveillance area, while target 2 remains stationary. The JPDAF system is able to successfully distinguish the targets and produce an accurate track.

4.3.2 Simulation Results

To test the performance of the JPDAF independently of the beamforming algorithm, we have used MATLAB to generate the angle inputs corresponding to two targets crossing paths. The following results come from the same JPDAF C code that runs on the nodes; MATLAB is used only to generate the inputs and analyze the results. The results are also compared to the NNSF under the same scenario to illustrate the significant advantages of the JPDAF approach.

The simulation scenario is setup as follows. There are two crossing targets ob-

served by three sensor nodes. Each node outputs three angles. For each node, one angle is Gaussian distributed centered around the correct angle to the target with a standard deviation of 5.7 degrees. Another angle is given by the same distribution centered around the second target. The third angle is a random number uniformly distributed between 0 and 360 degrees.

Figure 4-5 shows the results of the simulation. As the figure shows, the JPDAF can successfully track crossing targets when given reasonable LOB inputs in the presence of noise. By comparison, Figure 4-6 gives the results of the NNSF, which simply chooses the measurement closest to the expected measurement to use in the state update without carrying out the probability calculations of the JPDAF. The simulation results show that the NNSF yields significantly reduced performance, and in fact cannot successfully track two targets crossing paths. Once the two targets are near each other the same measurement will be used to update each of them, causing their state estimates to converge and making them indistinguishable.

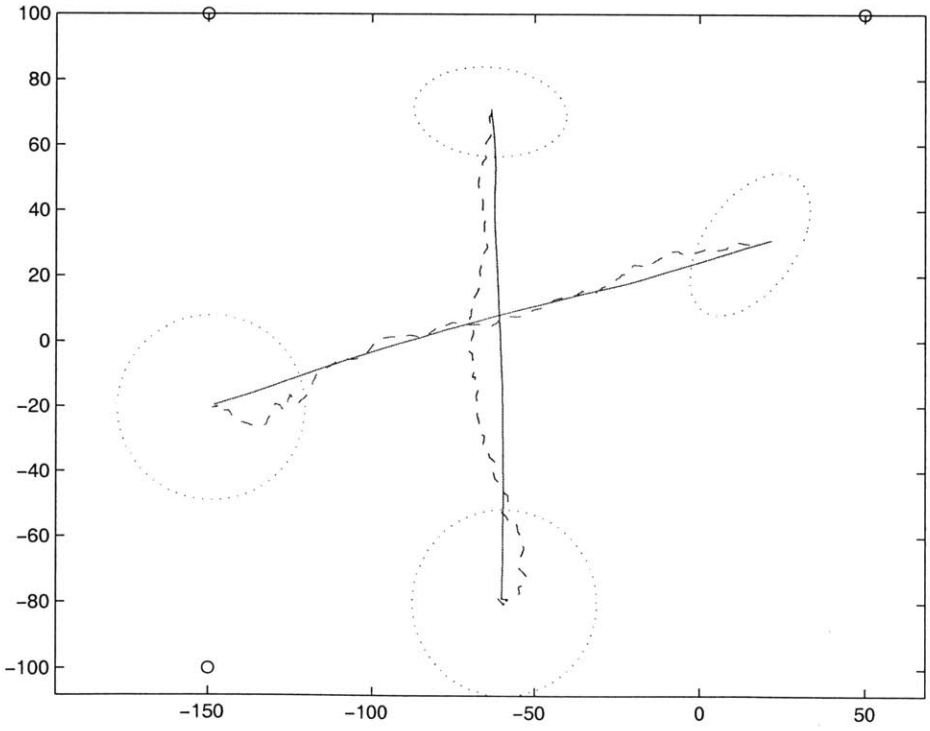

Figure 4-5: Crossing Targets Using JPDAF

Next we investigate how robust the JPDAF will be to non-ideal beamforming

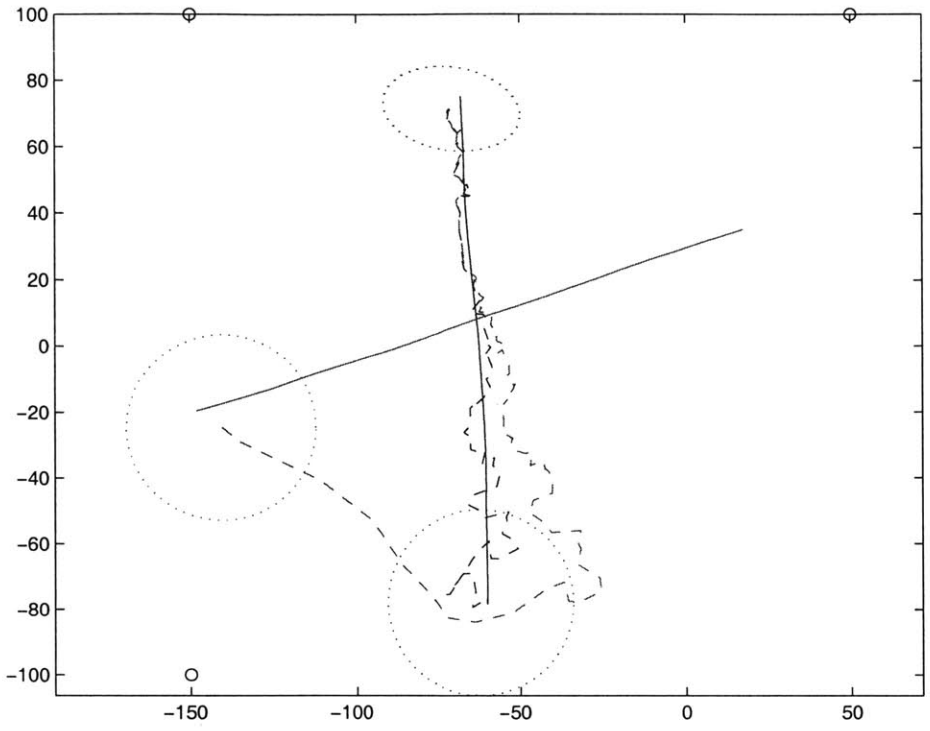

Figure 4-6: Crossing Targets Using NNSF

outputs. For this scenario the measurements that before were always Gaussian distributed around the angle to the target can now be a uniformly distributed random number with probability p, or drawn from the correct Gaussian distribution with probability $1 - p$. Table 4.1 gives the success rate for the JPDAF tracking crossing targets for values of p from 0 to 1.0. The results are based on 100 trials for each p value. Here "success" is defined as having each final state estimate be within a 2-σ ellipse of the true final state. As shown in the table, even if 30% of the beamforming outputs are random numbers, the JPDAF still has a 98% probability of successfully tracking the crossing targets. At 50% false measurements the filter retains a 79% probability of success. Figure 4-7 shows a sample run with only 50% correct measurements. Note the final covariances in 4-7 are larger than those of Figure 4-5, as the reduced number of good measurements increase the calculated uncertainty of the estimate. These results clearly show the power of the probabilistic data association techniques used in the JPDAF.

Probablility of False Measurement	JPDAF Success Rate
0	1.0
0.1	1.0
0.2	0.98
0.3	0.98
0.4	0.86
0.5	0.79
0.6	0.65
0.7	0.43
0.8	0.12
0.9	0.03
1.0	0

Table 4.1: JPDAF Success Rate with False Measurements

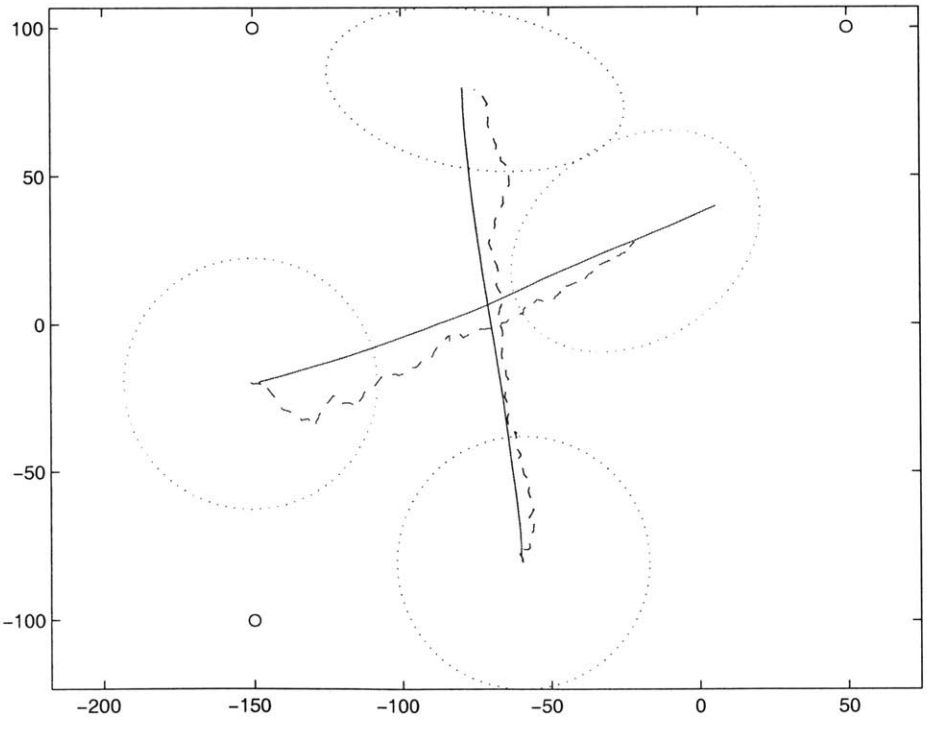

Figure 4-7: Crossing Targets Using JPDAF with 50% False Measurements

Chapter 5

Conclusions and Future Work

5.1 Kalman Filter Modifications

Several opportunities exist for future work extending this application. On the implementation side, moving from a standard Kalman filter to a square-root filter would yield increased numerical accuracy. The square-root filter carries out the covariance computations using the square root of the covariance matrix, which effectively doubles the numerical precision and reduces vulnerability to rounding errors. Also, implementing an adaptive IMM estimator would provide better tracking performance for maneuvering targets.

5.2 Initialization Procedure

In this application we have deferred the track initialization and maintenance problem to focus on the tracking itself. As mentioned in Chapter 4, the JPDAF assumes a known number of targets. The task of determining this number falls to a separate track initialization/maintenance module. One possible method would be to compute all intersections of the LOB measurements at each time step and consider each new intersection as a potential target. If a potential target exists for two time steps then a Kalman filter is initialized for it, and if it persists for a certain number of subsequent sampling times it is accepted as a target. Otherwise it is dropped. This method could

work well for relatively small numbers of sensors. It would have problems scaling, however, as the number of intersections would become prohibitively large. Another initialization method would be to simply initialize a grid of potential targets with large covariances covering the entire surveillance area, then prune off those that do not persist over subsequent sampling times.

5.3 Power-Saving Enhancements

A main focus of the μAMPS research has been designing systems that are both low power and power-aware. While low power refers to components designed to minimize power consumption, power-aware refers to systems designed to adapt their power consumption intelligently in response to the demands of a particular application at a particular time. The tracking system designed here has the potential to reduce power consumption by using the estimate uncertainty (covariance) produced by the Kalman filter to identify sensors that may be shut down without unacceptably degrading performance.

One such method involves monitoring the covariance matrix as the measurements from each sensor are processed, and flagging sensors whose measurements are not decreasing the covariance. These sensors are not contributing to a more accurate estimate and thus may be shut down to save power without losing performance. This technique has been implemented on the μAMPS nodes as part of this project.

5.3.1 Covariance Reduction Metric

The first decision to be made in this system is the choice of metric to capture a decreasing estimate uncertainty. The correct method would be to calculate the area of the ellipse given by the position elements of the state covariance matrix. A less computationally complex metric, however, which still captures the same effect is to simply add the magnitudes of the position elements of the covariance matrix. This simplified metric retains the desired property that an increase in the uncertainty along one dimension can be offset by a larger decrease along the other dimension.

5.3.2 Sleep Command System

The next design choice involves the method of shutting down the node. For this application, we have extended the μAMPS communication protocol to include a "sleep" control packet, which when received tells a node to enter a sleep mode for the next three seconds. Whenever the tracking application flags a particular node as not contributing to the estimation accuracy, the base station sends a sleep packet to that node. No explicit acknowledgement is sent for a sleep packet, as the system contains an inherent feedback mechanism. If the sleep packet is not received then the node in question will still send unhelpful measurements, causing the base station to send more sleep packets. Once the node does go to sleep, the base station will not receive measurements from it and thus no longer send sleep packets.

5.3.3 Power States During Sleep

Several choices exist for setting power consumption modes for the node sleep state. The processor power modes include active, idle, or sleep mode in decreasing order of power consumption. The radio board can likewise be set to active, standby, or shutdown. In shutdown mode all power to the radio board is turned off, while standby mode shuts down all radio circuitry but retains power to the FPGA providing the processor interface and the TDMA timing.

The lowest power consumption upon receiving a sleep packet would involve setting the processor to sleep mode and the radio to shutdown. Doing this, however, would lose TDMA synchronization, which would have to be reestablished upon wakeup before any communication could occur. This synchronization process can take several seconds on the current node, which is significant overhead. For this reason the processor is instead set to idle mode and the radio to standby, which allows TDMA synchronization to be maintained during the sleep state and communication to resume immediately after wakeup. The extra power consumed by using idle and standby during sleep mode is more than compensated by the avoidance of full power radio overhead that would otherwise be needed to reestablish TDMA synchronization.

5.3.4 Other Power-Saving Techniques

Several other power saving methods could be explored in addition to those implemented here. For example, the shape of the covariance matrix can be used to determine which sensors are expected to provide the most improvement in the estimate. In Figure 5-1, nodes B and C have provided good information about the target position along the axis perpendicular to them but little is known about the position along the axis between them. It would therefore be more power-efficient to briefly shut down nodes B and C and bring up nodes A and D to provide a balanced estimate. Another idea would be to make the application adapt to specific quality demands set by the user. The system could power on additional sensors when the computed uncertainty is greater than the limit set by the user, and power down sensors to save power when the quality of the estimate is greater than necessary. This adaptation would allow the power consumption of the system to scale gracefully with the quality demands of the user.

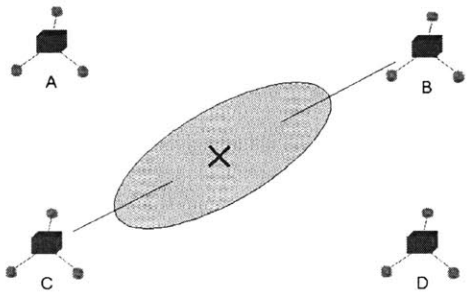

Figure 5-1: Selecting Sensors for Efficient Tracking

5.4 Conclusion

This work has successfully designed and implemented a tracking application capable of fusing measurements from a number of LOB sensor nodes to produce the optimal position and velocity estimates for one or more targets. Three algorithms have been demonstrated: the single-target Kalman filter, the multiple target tracking JPDAF,

and the frequency-domain Harmonic Line Association detection algorithm. We have presented both simulation and real-time results showing the ability of the Kalman filter-based algorithms to compensate for noisy measurements and to successfully track targets in situations where basic triangulation fails.

The Kalman filter returns not only the state estimate, but also its covariance matrix. We utilize the covariance matrix, represented as an ellipse in 2-D Cartesian space, to determine which sensor nodes are providing useful information by reducing the estimate covariance. The overall power consumption of the system is then reduced by shutting down unnecessary sensor nodes.

These power-saving techniques, combined with the three algorithms mentioned above, form the basis of a complete real-time target detection and tracking system. HLA provides information on target presence, probabilistic data association techniques associate the LOB measurements to targets, and the underlying Kalman filter produces the optimal target state estimate and covariance. The system has been successfully implemented to run in real-time on the μAMPS sensor nodes, and provides significant performance improvements over a standard triangulation approach.

Appendix A

Documentation

The source code for the programs described in this thesis is located on the machine eleventh-hour.mit.edu in subdirectories off of /usr/local/uamps/demos/.

The directory /usr/local/uamps/demos/kalman/ contains the single-target Kalman filter described in Chaper 3. The file kalman.c contains the filter itself, while beam_rx.c is the top-level file for a base station node. The file beam_tx.c contains the top-level code for a sensor node, which uses the beamforming algorithm code in lob.c. Running the makefile in this directory will produce beam_rx.s19 to be loaded into the base station, and beam_tx_X.c to be loaded into sensor node number X.

The Kalman filter requires knowledge of the location and orientation of each sensor node. Sensor location is set by calling kalman_set_pos(int pos[ ][ ]) with a two-dimensional array specifying x and y coordinates for each node in arbitrary units. Orientation is specified in the NODE_OFFSETS[ ] array in kalman.c, which contains the positive angle in degrees needed to rotate each nodes zero-degree reference to point directly east. Refer to comments in kalman.c for more detailed information regarding filter parameters.

The directory /usr/local/uamps/demos/kalman_jpdaf/ contains he code for the JPDAF filter described in Chapter 4. The JPDAF code is contained in kalman.c, and names for the remaining files are the same as the single-target case described above. The lob.c file in this directory contains the modified beamforming code that produces three angles per sensor node. This file can be used with a base station running a

single target filter – the extra angles will simply be ignored.

The code for the HLA algorithm is stored in /usr/local/uamps/demos/hla-seq/. The algorithm itself is contained in hla.c, while beam_tx.c provides a modified sensor node top-level file integrating the HLA procedure with the beamforming. The beamforming output rate will be reduced to one angle per second in an integrated node due to the long delay required for the HLA procedure.

Bibliography

[1] Y. Bar-Shalom and X. Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS Publishing, 1995.

[2] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation.* John Wiley and Sons, Inc., 2001.

[3] A. Doucet and N. Gordon. Efficient particle filters for tracking manoeuvring targets in clutter. *IEE Colloquim on Target Tracking: Algorithms and Applications*, pages 4/1–4/5, 1999.

[4] C. Hue, J.P. Le Cadre, and P. Perez. Tracking multiple targets with particle filtering using multiple receivers. *Target Tracking: Algorithms and Applications*, 1:6/1–6/4, 2001.

[5] R. Karlsson and F. Gustafsson. Monte carlo data association for multiple target tracking. *Target Tracking: Algorithms and Applications*, 1:13/1–13/5, 2001.

[6] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):103–123, January 1998.

[7] M.J. Moorman and T.E. Bullock. A new estimator for passive tracking of maneuvering targets. *First IEEE Conference on Control Applications*, 2:1122–1127, September 1992.

[8] M.J. Moorman and T.E. Bullock. A stochastic perturbation analysis of bias in the extended kalman filter as applied to bearings-only estimation. In *Proc. of the 31st IEEE Conference on Decision and Control*, 1992.

[9] R.A. Mucci. A comparison of efficient beamforming algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(3):548–558, June 1984.

[10] Soheil Shams. Neural network optimization for multi-target multi-sensor passive tracking. *Proc. of the IEEE*, 84(10):1442–1457, October 1996.

[11] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven algorithm and protocol design for energy-efficient wireless sensor networks. *Proceedings of MOBICOM*, July 2001.

[12] Xiaoling Wang and Hairong Qi. Acoustic target classification using distributed sensor arrays. *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing*, 4:IV–4186, 2002.

[13] Mark C. Wellman, Nassy Srour, and David B. Hillis. Feature extraction and fusion of acoustic and seismic sensors for target identification. *SPIE*, 30381:139–145.

[14] Feng Zhao, Jaewon Shin, and James Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, March 2002.