The Application of a Particle Filter to Urban Ground Target Localization, Tracking, and Intercept

by

Emily A. Doucette

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2012

Keywords: Particle Filter, Urban Environment, Path Planning

Approved by

Andrew J. Sinclair, Chair, Associate Professor of Aerospace Engineering
John E. Cochran, Jr., Professor and Department Head of Aerospace Engineering
David A. Cicci, Professor of Aerospace Engineering
Gilbert L. Crouse, Associate Professor of Aerospace Engineering
George T. Flowers, Dean of Graduate School

Abstract

Ground vehicle localization is a problem of significance in an urban setting given the recent global conflicts, security interests, and rapid growth of sensor networks. This task is often difficult due to the lack of information regarding a target vehicle's position, velocity, and destination. Field operatives can provide binary measurements of a target's presence in an area, and these measurements can be processed to obtain estimates of the target's location. A particle filter is more suitable for this application than a Kalman filter due to its ability to handle non-Gaussian distributions and non-differentiable measurement models, however it is computationally expensive.

Suppose there is a mobile ground vehicle of known description but unknown position, velocity, or destination that is to be found, tracked, and intercepted by an unmanned aerial vehicle. The vehicle is known to be in an urban environment, and full knowledge of that environment (roads, obstacles, intersection constraints, and speed limits) is available. There are numerous issues of interest within this problem. A particle filter in an urban environment was developed to locate, track, and intercept a ground vehicle given soft binary measurements (measurements from human sources). Two particular issues are studied in this work: the effect of a sophisticated particle dynamic model on target localization and tracking, and the development of a real time path planning routine in the particle filter framework to enable target interception.

The contributions of this work are threefold. First, the importance and impact of an accurate particle time update on target localization and tracking is validated. Secondly, a thorough investigation into the effect of particle spatial resolution in the presence of imperfect measurements is made that will prove valuable for future particle filter applications. Finally, the path planning routine offers reduced computational expense when compared to existing

systems and lends itself to unmanned aerial vehicle implementation. Proper exploitation and implementation of the particle filter framework prove vital in the complete characterization of the urban tracking problem.

Acknowledgments

The author would like to acknowledge and sincerely thank Dr. Andrew J. Sinclair for his guidance, support, patience, and encouragement throughout her time in graduate school. The author would also like to thank the SMART Scholar Program, Dr. J. Willard Curtis, and the Munitions Directorate of Air Force Research Laboratory for the opportunity to investigate this work. Also, the author sincerely appreciates the faculty of the Aerospace Engineering Department at Auburn University, namely Dr. R. Steven Gross, for many years of invaluable advisement and instruction and for all teaching opportunities granted.

Personal gratitude is bestowed upon the sisters and alumnae of the Zeta Xi Chapter of Alpha Xi Delta for the opportunity to realize lifelong friendships, leadership skills, and potential. The author expresses heartfelt thanks to her dear family and loyal friends for their constant encouragement and unwavering support. The powerful prayers and outpouring of love provided by late grandmother Imelda S. Gares and the strength and conviction to remain true to one's self provided by grandmother Molly M. Waller have always enabled the author to overcome any obstacle.

This work is dedicated to the author's beloved parents, Judith C. Doucette and the late Roland Doucette Jr., whose strong faith, self sacrifice, firm belief in hard work, dedication to one's best, encouragement, kindness, humor, and love are ever present, unmatched, inspirational, and forever held dear.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern statistical and engineering practice requires the ability to estimate and predict the behavior of nonlinear systems with accuracy, precision, and efficiency. This need, coupled with increased computational capabilities, has lead to a focused interest in nonlinear filtering over the past 30 years. Most problems of interest require a sequential estimate of a set of variables that may or may not change over time, called the state of the dynamic system. The state completely describes the dynamic system under investigation. Estimates of the state are made by analysis of measurements taken on that system. Measurements are assumed to be imperfect or noisy.

One scenario of significance to which nonlinear filtering has been applied is that of a ground target vehicle in an urban environment. The problem of locating, tracking, and intercepting a ground target in an urban environment presents numerous challenges. Traditional measurements such as range and range rate are not often readily available at regular intervals in such a setting. Information may come in the form of binary measurement reports from sources such as human observers, fixed cameras, or intermittent satellite images. This nontraditional measurement format can provide valuable information, although it is coupled with increased uncertainty. Therefore, it must be processed in an efficient and effective manner in order to account for its irregular frequency and non-differentiable measurement model. Additionally, prior knowledge of terrain and road networks provide vital insight into the target's state. Such information cannot be properly accounted for by estimation routines based on Gaussian assumptions.

This dissertation is centered on the numerous benefits the particle filter offers in solving certain difficult nonlinear filtering problems that exist, namely in urban warfare. Chapter 2

describes the most common approaches to the nonlinear filtering problem and the differences in their probabilistic representations of the state. The particle filter is compared to its more commonly used counterpart, the Kalman Filter, and its variations. The ease of adaptability of the particle filter to any motion model and the ability to represent knowledge of the state with a probability density function of arbitrary shape decided its application to this problem. Section 2.4 details previous work done in the field of particle filters, path planning, and risk assessment, as it relates to this dissertation.

Chapters 3 and 4 describe the benefits of improving and tuning each step in the particle filtered estimation process of prediction, correction, and resampling. Chapter 3 compares two methods for modeling the prediction step: a simplistic model and a heuristic model based off of conventional traffic rules. Both dynamic models are compared in regard to the time to detect the target, their ability to track the target, and their computational expense. Chapter 4 introduces a sensor model that provides false reports. Throughout the dynamic model study, the effect of additional variables are also investigated. The measurement model in question involves a human component, which presents the possibility of a false report. In the presence of a false report, the spatial resolution of the particle cloud is compromised based on bad information. False reports diminish tracking performance, but improvements may be made by adjusting built in parameters in the particle filter framework. The measurement update step updates particle weights in accordance with the amount of confidence the filter has in the measurements it receives. The amount the particle filter trusts a measurement may be tuned to reduce error when a sensor of known false alarm rate is in play. Section 4.1 details a study to determine the effect of the amount of confidence the particle filter has in a sensor's report. This confidence parameter is present in the likelihood function in the measurement update, or correction, step. The study involves Monte Carlo runs done with three sensor models and eleven possible confidence values for each sensor.

The frequency of resampling is also a particle filter design point. Resampling is done when the number of effective particles, or particles with significant weight, falls below a desired threshold. When imperfect measurements are known to be possible, resampling after each measurement is bad practice as the spatial resolution is compromised. An investigation of the effect of the number of effective particles was done to tune the filter to an optimal value based on the sensor's estimated performance. Section 4.2 explains how the frequency of resampling was treated as a control variable through tuning the number of effective particles. Following the determination of the most beneficial values of confidence and number of effective particles for each sensor model, a Bayesian risk assessment was included as outlined in Section 4.3 as an additional means of dealing with possibly false reports.

The discussion of the second phase of this dissertation begins in Chapter 5. An unmanned aerial vehicle (UAV) is introduced into the urban scenario with the mission of intercepting the ground target. A path planning routine was developed by the use of receding horizon control. A novel path-selection metric is developed to provide for target interception in the particle filter framework. A new cost function is created to avoid high computational expense of previously used minimum entropy formulations. The path planner selects paths with high particle weight, taking anticipated travel time into account. In addition, a minimum distance between a path and the location of the heaviest particle is desired. This method selects paths with maximum likelihood, while reducing entropy, with far fewer calculations than previous work. This will prove most effective in smaller platforms where computational power is limited, as in small UAV's to be utilized in urban warfare.

The result of this work is a robust simulation with real world applications and benefits to the modern war fighter.

Chapter 2

The Nonlinear Dynamic System Filter Problem

The objective of nonlinear filtering is to recursively estimate the state of a dynamic system based on measurements. The estimation of the state is typically broken into two steps: prediction and correction. The prediction step models the propagation of the state through time between measurements. The equations of motion of the state variables are propagated through time until a measurement is taken. Process noise is included in the state equations to account for inaccurate modeling and unforseen disturbances. This noise causes the accuracy of the state estimate to diverge from the true state until a measurement is taken. In the ground-target scenario, this divergence can be attributed to uncertainty in a target's position, velocity, or destination. The correction step incorporates measurement data into the state estimate. Each measurement is taken into account in accordance with its confidence to improve the state estimate.

These models can be represented probabilistically and therefore lend themselves to the Bayesian framework. In this rigorous general approach, a probability density function (*pdf*) is used to store all knowledge and belief about the true state. During the prediction step, the state *pdf* is generally translated, deformed, and broadened, whereas following the corrector step, it is typically tightened. The state *pdf* is updated with the new measurement by applying Bayes' theorem.

Suppose the target state vector is $\mathbf{x}_k \in \mathbb{R}^n$, where the k is the time index, $\mathbb{R}$ is the set of real numbers, and n is the dimension of the state vector. The state evolves according to a discrete-time stochastic model in a first-order Markov process.

$$\mathbf{x}_k = \mathbf{f}_{k-1}\left(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}\right) \tag{2.1}$$

In Eq. (2.1), the function $\mathbf{f}_{k-1}$ describes the behavior of the state at $\mathbf{x}_{k-1}$, and $\mathbf{v}_{k-1}$ is the process noise. Because the state evolves according to a Markov process, the future state is only dependent upon the current state.

$$\mathbf{z}_k = \mathbf{h}_k \left(\mathbf{x}_k, \mathbf{w}_k \right) \tag{2.2}$$

Measurements are related to the state by the measurement vector of dimension m. In Eq. (2.2), the function $\mathbf{h}_k$ is a known and possibly nonlinear function of the state and it includes some measurement noise, $\mathbf{w}_k$. Both the measurement noise and process noise are assumed to be white, independent, and to have known probability density functions. The sequence of all measurements, $\mathbf{Z}_k \equiv \{\mathbf{z}_i, i = 1, ..., k\}$, is used to obtain filtered estimates of the state. For implementation in the Bayesian framework, a posterior *pdf* must be constructed to quantify the belief in the state $\mathbf{x}_k$ given $\mathbf{Z}_k$. This *pdf*, $p\left(\mathbf{x}_k \mid \mathbf{Z}_k\right)$, is calculated recursively using the prediction correction process. It is assumed that $p\left(\mathbf{x}_0 \mid \mathbf{Z}_0\right)$ is known where $\mathbf{z}_0$ is the set of no measurements and therefore independent of all noise.

To begin recursive estimation, it is assumed that $p\left(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}\right)$ is available. The prediction step is carried out by using the system equation (2.1) to create the predicted or prior density of the state at time k via the Chapman-Kolmogorov equation:

$$p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right) = \int p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}\right) d\mathbf{x}_{k-1} \tag{2.3}$$

The probabilistic model of the state evolution, or the transitional density, $p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}\right)$, is defined by the system equation (2.1) and the known statistical parameters of $\mathbf{v}_{k-1}$. When a measurement, $\mathbf{z}_k$, becomes available, the correction stage begins. By use of measurement $\mathbf{z}_k$, the prior density $p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)$ is modified and the posterior density of the current state

$p(\mathbf{x}_k \mid \mathbf{Z}_k)$ is obtained.

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{Z}_k) &= p(\mathbf{x}_k \mid \mathbf{z}_k, \mathbf{Z}_{k-1}) \\ &= \frac{p(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{Z}_{k-1})\, p(\mathbf{x}_k \mid \mathbf{Z}_{k-1})}{p(\mathbf{z}_k \mid \mathbf{Z}_{k-1})} \\ &= \frac{p(\mathbf{z}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{Z}_{k-1})}{p(\mathbf{z}_k \mid \mathbf{Z}_{k-1})} \end{aligned} \tag{2.4}$$

$$p(\mathbf{z}_k \mid \mathbf{Z}_{k-1}) = \int p(\mathbf{z}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{Z}_{k-1})\, d\mathbf{x}_k \tag{2.5}$$

From this posterior density, an optimal state estimate may be computed. Such forms for the estimate include the the minimum mean-square error (2.6) and the maximum a posteriori estimate (2.7), or the conditional mean of $\mathbf{x}_k$ and the maximum of $p(\mathbf{x}_k \mid \mathbf{Z}_k)$, respectively.

$$\hat{\mathbf{x}}_{k|k}^{MMSE} \equiv \mathbf{E}\{\mathbf{x}_k \mid \mathbf{Z}_k\} = \int \mathbf{x}_k p(\mathbf{x}_k \mid \mathbf{Z}_k)\, d\mathbf{x}_k \tag{2.6}$$

$$\hat{\mathbf{x}}_{k|k}^{MAP} \equiv \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k \mid \mathbf{Z}_k) \tag{2.7}$$

Recursive propagation of the posterior density through Eqs. (2.3) and (2.5) provide the basis for the optimal Bayesian solution, however it is only conceptual in that it cannot be generally determined analytically. The storage and representation of the entire *pdf* requires a finite dimensional representation of an infinite dimensional function. Consequently, approximations or suboptimal Bayesian algorithms are conventionally used in practice. Namely, the Kalman filter, the extended Kalman filter, the Unscented Kalman filter, and the particle filter are methods of note in tracking applications [1].

2.1 The Kalman Filter

In 1960, R. E. Kalman published a paper that provided a rigorous mathematical approach for sequentially processing observations of a linear dynamic system [6]. This approach was conveniently introduced at a time when computational power and technology were on

the rise, allowing for its successful implementation in numerous applications and consequential popularity. The set of recursive equations outlined in the 1960 paper are known as the Kalman filter [2].

The Kalman filter requires certain restrictive assumptions. It is assumed that the posterior density is Gaussian at every time step. This provides for the posterior density to be characterized by its mean and covariance, but it also requires that the system equation $\mathbf{x}_k$ be a linear function of $\mathbf{x}_{k-1}$ and $\mathbf{v}_{k-1}$, the measurement equation $\mathbf{z}_k$ is a linear function of $\mathbf{x}_k$ and $\mathbf{w}_k$, and the noise terms $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are drawn from Gaussian densities of known mean and variance. If these restrictions hold, the Kalman filter yields the optimal solution [1]. In the Kalman filter framework, Eqs. (2.1) and (2.2) become:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{2.8}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{w}_k \tag{2.9}$$

where $\mathbf{F}_{k-1}$ (n x n) and $\mathbf{H}_k$ (m x n) are known. Also, the sequences $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are mutually independent zero-mean white Gaussian noise with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively. Due to the assumption that all posterior *pdf's* are Gaussian, the *pdf* update equations become:

$$p\left(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}\right) = \mathcal{N}\left(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}\right) \tag{2.10}$$

$$p\left(\mathbf{x}_{k} \mid \mathbf{Z}_{k-1}\right) = \mathcal{N}\left(\mathbf{x}_{k}; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\right) \tag{2.11}$$

$$p\left(\mathbf{x}_{k} \mid \mathbf{Z}_{k}\right) = \mathcal{N}\left(\mathbf{x}_{k}; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}\right) \tag{2.12}$$

where $\mathcal{N}\left(\mathbf{x}; \mu, \mathbf{P}\right)$ is a Gaussian density with argument $\mathbf{x}$, mean μ, and covariance $\mathbf{P}$, defined by:

$$\mathcal{N}\left(\mathbf{x}; \mu, \mathbf{P}\right) \equiv \left|2\pi\mathbf{P}\right|^{-1/2} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu\right)^T \mathbf{P}^{-1}\left(\mathbf{x}-\mu\right)\right\} \tag{2.13}$$

The Kalman filter measurement update equations for the mean and covariance are given by the equations below.

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{2.14}$$

$$\mathbf{P}_{k|k-1} = \mathbf{Q}_{k-1} + \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{T} \tag{2.15}$$

$$\mathbf{K}_{k} = \mathbf{P}_{k|k-1}\mathbf{H}_{k}^{T}\left[\mathbf{H}_{k}\mathbf{P}_{k|k-1}\mathbf{H}_{k}^{T} + \mathbf{R}_{k}\right]^{-1} \tag{2.16}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_{k}\left(\mathbf{z}_{k} - \mathbf{H}_{k}\hat{\mathbf{x}}_{k|k-1}\right) \tag{2.17}$$

$$\mathbf{P}_{k|k} = \left[\mathbf{I} - \mathbf{K}_{k}\mathbf{H}_{k}\right]\mathbf{P}_{k|k-1} \tag{2.18}$$

The Kalman Filter provides optimal solutions for a certain class of problems, but it was expanded upon in order to be applied to nonlinear systems and/or measurement models. This lead to the development of the Extended Kalman Filter and the Unscented Kalman Filter [1].

2.1.1 The Extended Kalman Filter

In practical situations, the strict restrictions on the Kalman Filter are not satisfied. The Extended Kalman filter (EKF) operates on the basic premise that the true state is sufficiently close to the estimated state. This provides for the use of a linearized first order Taylor series expansion to represent the error dynamics. The posterior *pdf* $p\left(\mathbf{x}_{k} \mid \mathbf{Z}_{k}\right)$ is assumed Gaussian and therefore Equations (2.10) through (2.12) are valid. Although the EKF is not considered optimal like the Kalman Filter, it is a well known technique for nonlinear system estimation [2].

In the Extended Kalman Filter framework, Equations (2.1) and (2.2) become:

$$\mathbf{x}_{k} = \mathbf{f}_{k-1}\left(\mathbf{x}_{k-1}\right) + \mathbf{v}_{k-1} \tag{2.19}$$

$$\mathbf{z}_{k} = \mathbf{h}_{k}\left(\mathbf{x}_{k-1}\right) + \mathbf{w}_{k} \tag{2.20}$$

The random noise sequences $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are assumed zero mean, mutually independent, white Gaussian with covariances $\mathbf{Q}_{k-1}$ and $\mathbf{R}_k$, respectively. The following recursive equations are used to compute the mean and covariance of the state:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}\left(\hat{\mathbf{x}}_{k-1|k-1}\right) \tag{2.21}$$

$$\mathbf{P}_{k|k-1} = \mathbf{Q}_{k-1} + \hat{\mathbf{F}}_{k-1}\mathbf{P}_{k-1|k-1}\hat{\mathbf{F}}_{k-1}^T \tag{2.22}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\hat{\mathbf{H}}_k^T\left[\hat{\mathbf{H}}_k\mathbf{P}_{k|k-1}\hat{\mathbf{H}}_k^T + \mathbf{R}_k\right]^{-1} \tag{2.23}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{h}_k\left(\hat{\mathbf{x}}_{k|k-1}\right)\right) \tag{2.24}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\left(\hat{\mathbf{H}}_k\mathbf{P}_{k|k-1}\hat{\mathbf{H}}_k^T + \mathbf{R}_k\right)\mathbf{K}_k^T \tag{2.25}$$

The local linearizations of $\mathbf{f}_{k-1}$ and $\mathbf{h}_k$ are $\hat{\mathbf{F}}_{k-1}$ and $\hat{\mathbf{H}}_k$. They are the Jacobian of the corresponding nonlinear equations evaluated at $\hat{\mathbf{x}}_{k-1|k-1}$ and $\hat{\mathbf{x}}_{k|k-1}$, respectively. The EKF is known as an analytic approximation because the aforementioned Jacobians must be derived analytically. Clearly, if the nonlinear functions $\mathbf{f}_{k-1}$ and $\mathbf{h}_k$ are discontinuous or nondifferentiable, the EKF may not be applied. In addition, the EKF is restricted by its assumption that the posterior *pdf* $p\left(\mathbf{x}_k \mid \mathbf{Z}_k\right)$ is Gaussian. Models with severe nonlinearities will result in increased errors due to this assumption. As a result, discrete sampling approaches, such as the Unscented Kalman Filter and the Particle Filter, were developed. These methods are more suited for urban tracking problems.

2.2 The Particle Filter

The Particle Filter is a suboptimal filter that performs sequential Monte Carlo estimation utilizing independent random samples of probability densities. These samples, called particles, are point mass representations of the posterior *pdf* of the state. Like the Kalman filter, the Particle filter (PF) gained heightened popularity with an increase of computational capabilities. Although the framework for sequential Monte Carlo estimation originated in statistics in the 1950's, the method was also limited by the use of sequential importance

sampling, which degenerates over time. The introduction of the resampling step in the early 1990's [3], coupled with increased computational capabilities, lead to the drastic increase in PF implementation.

Monte Carlo methods for the numerical evaluation of multidimensional integrals form the basis for the PF approach. Suppose the solution to the following multidimensional integral is required, where $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{I} = \int \mathbf{g}(\mathbf{x}) d\mathbf{x} \tag{2.26}$$

Monte Carlo (MC) integration methods factorize $\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x})\pi(x)$ such that $\pi(x)$ may be interpreted as the posterior *pdf* in the Bayesian framework. It is assumed possible to draw N independent samples $\{\mathbf{x}^i; i = 1, ..., N\}$, where $N \gg 1$, distributed according to $\pi(x)$. This yields the unbiased MC estimate of I, I_N, that converges to I with error of the order $\mathcal{O}(N^{-1/2})$.

$$\mathbf{I} \approx \mathbf{I}_N = \frac{1}{N} \sum_{i=1}^{N} \mathbf{f}(\mathbf{x}^i) \tag{2.27}$$

The rate of convergence is independent of the dimension of the integrand because the samples, $\mathbf{x}^i$, are chosen from regions of high importance relative to the state space. It is, however, often difficult to sample from the posterior *pdf* as it is nonstandard, multivariate, and not fully known. Consequently, the MC integration method of importance sampling is used sequentially to draw samples that represent the posterior *pdf*. The importance-sampling method samples and weighs the points from a known importance density function to simulate the samples from an unknown distribution. This method represents the posterior *pdf* with a set of random samples and associated weights, then estimates the state based on the samples and weights. When importance sampling is applied to the nonlinear filtering problem described in Section 2, the result is the Sequential Importance Sampling (SIS) algorithm, a sequential MC filter also known as the Particle Filter.

An initial estimate of the state *pdf*, $p(x_0)$, is assumed known, may be arbitrarily chosen, and is described by the set of particles x_0^i with associated weights w_0^i, such that $\sum_{i=1}^{N} w_k^i = 1$ at any time k. The PF follows the same prediction-correction pattern as the aforementioned filters. The state *pdf* at time $k-1$ is approximated by a sum of delta functions:

$$p\left(\mathbf{x}_{k-1} \mid \mathbf{Z}_{k-1}\right) \approx \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^i) \tag{2.28}$$

The prediction step consists of propagating the particles forward in time according to the dynamic model, $\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})$. This yields the predicted state *pdf*, $p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)$. When a measurement, $\mathbf{z_k}$, is received, the state *pdf* is updated in the correction step.

$$p\left(\mathbf{x}_k \mid \mathbf{z}_k\right) = \frac{p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)}{p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)} \tag{2.29}$$

The measurement likelihood function, $p\left(\mathbf{z}_k \mid \mathbf{x}_k\right)$, is used to update the particles weights according to Bayes' Rule.

$$w_k^i = \frac{w_{k-1}^i p\left(\mathbf{z}_k \mid \mathbf{x}_k^i\right)}{\sum_{j=1}^{N} w_{k-1}^j p\left(\mathbf{z}_k \mid \mathbf{x}_k^j\right)} \tag{2.30}$$

It has been shown that the variance of the importance weights can only increase over time in the presence of measurements [4]. This degeneracy phenomenon results in one particle holding all of the weight after a certain number of recursive steps. It is desirable to keep track of the number of effective particles that hold a non-negligible weight, $N_{\text{eff}} < N$. This is done through the following calculation:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{2.31}$$

The process of resampling is used when N_{eff} falls below the desired threshold, N_{thr}. Although resampling is computationally expensive because it is not parrallelizable, it is necessary to avoid particle degeneracy. Resampling eliminates particles with relatively low weight and creates duplicates of particles with relatively high weight. The new set of particles is obtained

by resampling with replacement N times from the approximation of $p(\mathbf{x}_k \mid \mathbf{z}_k)$ given by the set of particle states and weights, $\{\mathbf{x}_k^i, w^i\}$. The resulting particle set is an independent and identically distributed sample from the posterior *pdf*. There are multiple resampling methods, but one of the most computationally efficient, $\mathcal{O}(N)$, and simple methods is systematic resampling. The pseudocode for systematic reampling is given below [1], where x_k^i is the state of the ith particle, w_k^i is the weight of the ith particle, and CSW is the cumulative sum of the particle weights.

Initialize the CSW : $c_1 = w_k^1$

for $i = 2 : N$ **do**

Construct CSW : $c_i = c_{i-1} + w_k^i$

end for

Start at the bottom of the CSW : $j = 1$

Draw a starting point : $u_1 \sim \mathcal{U}[0, N^{-1}]$

for $i = 1 : N$ **do**

Move along the CSW : $u_i = u_1 + (i-1)/N$

while $u_i > c_i$ **do**

$j = j + 1$

end while

$parent(i) = j$

end for

Assign sample : $x_k = x_k(:, parent)$

Assign weight : $w_k = 1/N$

Figure 2.1 pictorially describes the process for determining which particles should be eliminated and which should be duplicated. The cumulative sum of particle weights (CSW) is compared to the random variable $u_i, i = 1, ..., N$ that is uniformly distributed on the interval $[0, 1]$. If a particle's weight does not significantly increase the cumulative sum when compared to the next element in the uniform distribution, that particle will be eliminated.

Conversely, if a particle's weight significantly increases the cumulative sum when compared to the next element in the uniform distribution, that particle will be duplicated at least once. Given sufficient computing power and trustworthy sensor input, resampling should be

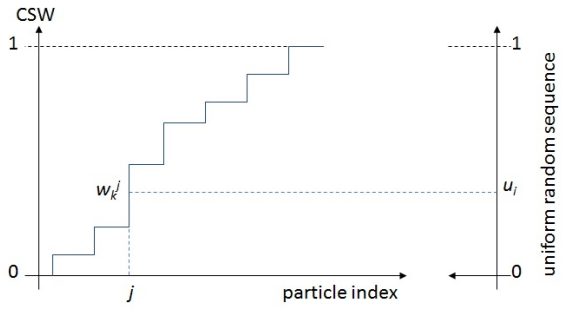


Figure 2.1: The Resampling Process

done following each measurement to maintain sufficient particle resolution. Figure 2.2 [1] illustrates the introduction of a measurement and the effect of resampling. More particles are introduced in areas of higher probability and particles are removed from areas of low probability.

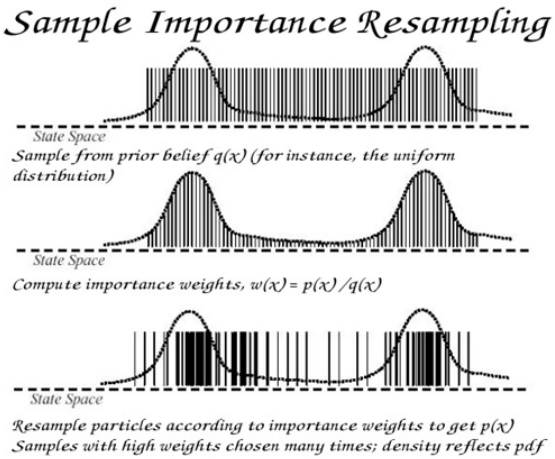


Figure 2.2: Effect of Resampling

It is evident that the PF framework does not make assumptions regarding the shape of the posterior *pdf*. Also, it puts no restriction on the forms of the dynamic or measurement

models. Therefore, it is utilized in this work to estimate the location of a moving ground vehicle.

2.3 Entropy of a Probability Density Function

Entropy is often used as a measure of uncertainty of an estimate in the field of target tracking, [24, 26, 28]. It is a positive scalar quantity that represents the amount of uncertainty in an estimate. Entropy reduction has been used in cost functions in problems regarding sensor assignment to determine the best configuration to reduce uncertainty in target location.

For a general filtering density, $p\left(\mathbf{x}_k|\mathbf{Z}_k\right)$, the entropy is given by:

$$\mathbf{H}\left(\mathbf{p}\left(\mathbf{x_k}|\mathbf{Z_k}\right)\right) = -\int_X \log_2\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) p\left(\mathbf{x}_k|\mathbf{Z}_k\right) d\mathbf{x}_k \tag{2.32}$$

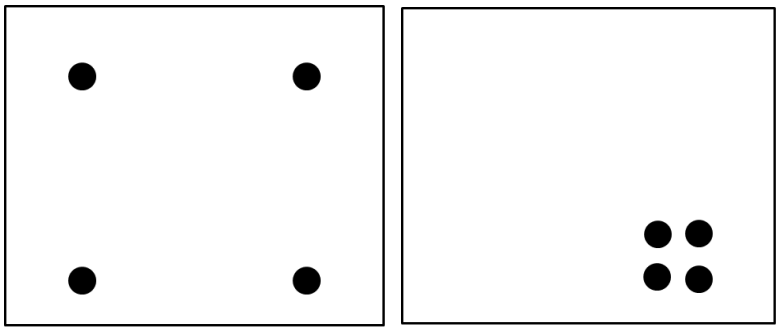
For an n-dimensional Gaussian variable with covariance matrix P, as in the Kalman filter framework, the entropy H is given by:

$$\mathbf{H} = \log\sqrt{(2\pi e)^n |P|} \tag{2.33}$$

When put into the particle filter framework, the resulting expression for entropy is:

$$\mathbf{H}\left(\mathbf{p}\left(\mathbf{x_k}|\mathbf{Z_k}\right)\right) \approx -\sum_{i=1}^{N} w_k^i \log_2(w_k^i) \tag{2.34}$$

This result is incorrect as it gives no mention to the location and local density of the particle distribution. For example, the two sample distributions illustrated by Fig. 2.3 each contain four particles of equal weight (0.25) that are used to represent the location of a target [5].



(a) (b)

Figure 2.3: (a) Low density *pdf* and (b) High Density *pdf*

The distribution in Fig. 2.3(a) is more spread out than that in (b), corresponding to more uncertainty in target location given the distribution. However, both distributions would have an entropy value of 2 if Eq. (2.34) is used.

The development of an approximate expression for the entropy of a non-Gaussian *pdf* as is often present in particle filter implementations has been the subject of recent research. Driessen et al. [5] used a 2-D bimodal distribution in a target tracking application to validate the following entropy approximation:

$$\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) = -\int_X \log_2\left(p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{Z}_{k-1}\right)\right)p\left(\mathbf{x}_k|\mathbf{Z}_k\right)d\mathbf{x}_k + \log_2\left(p\left(\mathbf{z}_k|\mathbf{Z}_{k-1}\right)\right)$$

$$\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) \approx \log\left(\sum_{i=1}^{N} p\left(\mathbf{z}_k|\mathbf{x}_k^i\right)w^i\right) - \sum_{i=1}^{N}\log\left(p\left(\mathbf{z}_k|\mathbf{x}_k^i\right)\left(\sum_{j=1}^{N} p\left(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i\right)w_{k-1}^j\right)\right)w_k^i$$

Skoglar et. al [26] implemented and tested the following expression on a 1-D Gaussian sum density:

$$\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) \approx \sum_{i=1}^{N} w_k^i \log p\left(z_k|x_k^i\right) - \sum_{i=1}^{N} w_k^i \log \sum_{i=1}^{N} w_{k-1}^i p\left(x_k^j|x_{k-1}^i\right) + \log \sum_{i=1}^{N} w_{k-1}^i p\left(z_k^i|x_{k-1|k}^i\right)$$

In addition, Ryan [28] derived a piecewise linear approximation of a *pdf* represented by a particle set by using particle locations as vertices of triangular elements, and the height of the elements being $p(x_k^i|\mathbf{Z}_k)$. Figure 2.4 illustrates the evolution of the particle cloud after a measurement update and the piecewise linear approximation of the *pdf* implemented by Ryan [28] to calculate the entropy of the distribution.

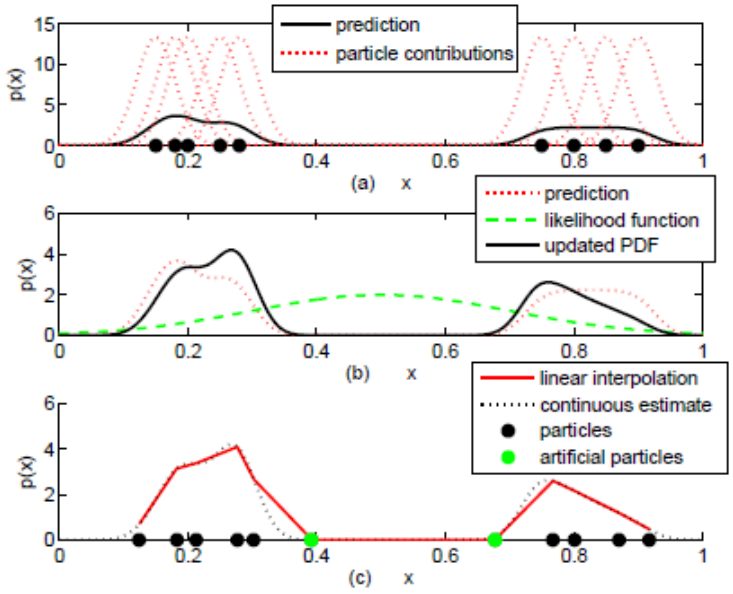


Figure 2.4: Piecewise linear approximation of a PDF

As there is yet to be a standard expression for the entropy of a distribution represented by particles, an indirect method for reducing entropy is desired. Hoffmann and Tomlin [24]

present an information utility function, $\mathbf{I}(z_k|x_k)$, that when maximized, will reduce the expected entropy of the target state, $\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right)$.

$$\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) = \mathbf{H}\left(\mathbf{x}_k\right) - \mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right)$$

where

$$\begin{aligned}
\mathbf{H}\left(\mathbf{x}_k\right) &= -\int p\left(\mathbf{x}_k\right)\log_2 p\left(\mathbf{x}_k\right)d\mathbf{x} \\
\mathbf{H}\left(p\left(\mathbf{x}_k|\mathbf{Z}_k\right)\right) &= -\int p\left(\mathbf{x}_k,\mathbf{z}_k\right)\log_2 p\left(\mathbf{x}_k|\mathbf{z}_k\right)d\mathbf{x}d\mathbf{z} \\
\mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right) &= \int p\left(\mathbf{x}_k,\mathbf{z}_k\right)\log_2 \frac{p\left(\mathbf{x}_k,\mathbf{z}_k\right)}{p\left(\mathbf{x}_k\right)p\left(\mathbf{z}_k\right)}
\end{aligned}$$

Additionally, Hoffmann and Tomlin express $\mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right)$ as:

$$\mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right) = \mathbf{H}\left(\mathbf{z}_k\right) - \mathbf{H}\left(p\left(\mathbf{z}_k|\mathbf{x}_k\right)\right)$$

which can be approximated by:

$$\begin{aligned}
\mathbf{H}\left(\mathbf{z}_k\right) &\approx -\int\left\{\sum_{i=1}^{N}\left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right)\log_2\left[\sum_{i=1}^{N}\left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right)\right]\right\}d\mathbf{z} \\
\mathbf{H}\left(\mathbf{z}_k|x_k\right) &\approx -\int\sum_{i=1}^{N}\left[w_k^i p\left(\mathbf{z}_k|x_k^i\right)\log_2 p\left(\mathbf{z}_k|x_k^i\right)\right]d\mathbf{z}
\end{aligned}$$

The Information Utility Function is based on the idea that minimizing the posterior uncertainty is equivalent to maximizing the difference between the uncertainty that any particular observation will be made $\mathbf{H}\left(\mathbf{z}_k\right)$, and the uncertainty of the measurement model, $\mathbf{H}\left(p\left(\mathbf{z}_k|\mathbf{x}_k\right)\right)$. The terms above are readily available in a particle filter and may be used to select sensor placement to indirectly reduce the entropy of a particle distribution.

2.4 Applications to Target Localization and Tracking

The most computationally efficient algorithms used for ground target tracking are based on Kalman filtering [5, 6, 7]. However, the aforementioned strict assumptions in Kalman filtering do not hold in urban target tracking, as the system is nonlinear, non-Gaussian, and possibly multi-modal. A particle filter requires no assumption about the distribution of the uncertainties and linearity of the system dynamics, consequently a non-Gaussian posterior distribution is admissible [1, 3, 4, 8]. The urban terrain is characterized by a road network, traffic signs and signals, speed limits, crossings, and traffic participants such as vehicles, bicycles, and pedestrians. Solutions that do not utilize these features may lead to suboptimal performance. Exploitation of prior knowledge of the terrain attributes, for instance, road maps and traffic information, are therefore highly desirable to significantly improve the target tracking performance [9].

The earliest usage of particle filters for target tracking with road network information can be found in [10], where Arulampalam et al. presented an algorithm termed Variable Structure Multiple Model Particle Filter (VS-MMPF) with ground moving target indicator (GMTI) radar using non-standard information such as road maps and terrain-related visibility conditions. Subsequently, Agate and Sullivan [11] demonstrated the feasibility of jointly tracking and identifying targets using a particle filter. Yang et al. [12] explored several ways to include terrain database and road maps to assist the tracking of ground targets and discussed motion models for brake to stop, road intersections, and target interactions. Further, Ulmke and Koch [13] developed a GMTI based target tracking approach using the discrete Gauss-Markov target dynamics and road-map information. Ekman and Sviestins [14] introduced the Multiple Likelihood Model Particle Filter (MLM-PF) for target tracking in the presence of hard constraints such as speed or acceleration demonstrating excellent tracking performance. Concurrently, Kamrani and Ayani [15] presented a method for path planning of an unmanned aerial vehicle with the task of tracking a moving target on a road network.

Orguner et al. [16] considered the problem of tracking targets, which can move both on-road and off-road, while the results suggested the preference of Interacting Multiple Model Particle Filter over Bootstrap Multiple Model Particle Filter. Kreucher et al. [23] utilized a particle filter and the expected information gain as the metric to assign a sensor to track multiple targets. Each second, a 100 m x 100 m section of the simulation space was inspected and a binary response of target presence was given.

With the intention of reducing computational time, Hong et al. [17] presented a Multi-rate Interacting Multiple Model Particle Filter that reduced by half the computational cost in comparison to the Multiple Model Particle Filter for terrain based ground target tracking. Thereafter, Kravaritis and Mulgrew [18] introduced the Variable-mass Particle Filter (VMPF) for terrain-aided tracking problem by allocating particles with variable masses to the modes as against VS-MMPF. The simulation results in [18] showed the improved efficiency of the VMPF. Skoglar et al. [19] proposed a Rao-Blackwellized Particle Filter to reduce the dimension of a state space in road target tracking application, and showed the use of prior information about the probability of detection can be used to improve the estimation performance.

The goals of the search problem are well framed in the context of information theoretic costs. Information theoretic cost metrics have been used to manage sensors [20], and have led to algorithms to control sensor networks for information gathering over an area by parameterizing the motion of collectives of vehicles [21]. The optimal probing control law to minimize Shannon entropy for the dual control problem has been shown to be the input that maximizes mutual information [22]. The expected alpha-divergence of a particle filter distribution was used for sensor management, and specialized to select modes for binary sensors, though scalability in sensor network size was not addressed, and the Shannon entropy was only found in the limit of the presented equations [23]. An earlier version of entropy approximation techniques was presented in [24]. The continuation of that work, presented in [25], develops an entropy-based metric to maximize the mutual information in a mobile

sensor network. Significant work has been done in the field of sensor assignment. Spletzer and Taylor select the difference in the expected value of the *pdf* and each particle's location as the metric for path selection [31].

Wang et al. study a linearized system with a Kalman Filter in the Bayesian framework to optimize when to accept an estimate and the risk associated with it based on Renyi information divergence. [32]. O'Reilly applied Bayesian likelihood ratio tests to the detection of faults within a sensor by observing potential changes in the variance of the estimate [33].

Chapter 3

The Effect of the Particle Motion Model

A ground vehicle is to be located and tracked in an urban environment. This chapter will investigate the effect of a high fidelity particle motion model on localization and tracking performance given unconventional measurements. Two motion models will be compared to explore the benefits and development of a sophisticated dynamic model. The performance of each motion model will be studied in three ways: computational expense, the amount of time to detect the target, and the tracking capabilities post-detection. The particle filter is not restricted in the form of the dynamic models or probability distributions. This chapter will investigate the multi-resolution problem of maintaining sufficient spatial resolution while distributing particle weight according to the likelihood of target presence.

A thorough description of the simulation environment will be discussed, as well as the necessary assumptions that were made. The motion model utilized to simulate the target ground vehicle will be introduced, followed by the measurement model and the two particle motion models. Section 3.5 presents comparison results for the two motion models. Reduced performance by the high-fidelity model in the amount of time it takes to detect the target motivated further investigation into the role of particle spatial resolution by means of particle redistribution and allowing for the belief of the presence of false measurement reports.

3.1 Problem Description

Suppose there is a mobile ground vehicle of known description but unknown position, velocity, and destination that is to be found, tracked, and intercepted by an unmanned aerial vehicle. The vehicle is known to be in an urban environment, and full knowledge of that environment (roads, obstacles, intersection constraints, and speed limits) is available. Urban

warfare often relies on information from passersby, agents in the field, and traffic cameras. The inclusion of such non-traditional and non-differentiable measurements in addition to the shape of a complex road network necessitates the use of a particle filter to estimate the location of the target vehicle. In this problem, measurements are received from field operatives in the form of binary responses of target presence within a sector of the city. There are numerous issues of interest within this problem. The effect of a sophisticated particle motion model on target localization time and tracking capabilities in the presence of unconventional measurements will be studied in this chapter.

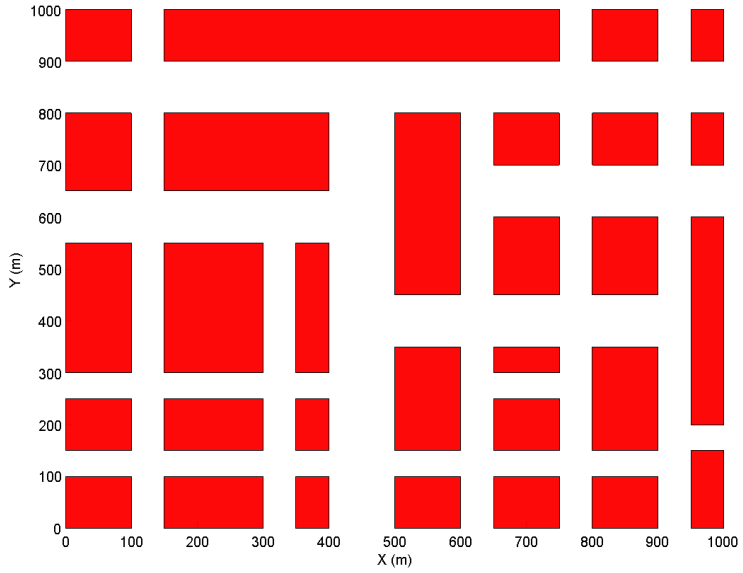


Figure 3.1: Map of Urban Environment

The urban environment utilized in this work is illustrated by Fig. 3.1, where obstacles are orange and roads are white. Complete knowledge of the map is assumed, including the location of obstacles and roads.

Several assumptions are made throughout the simulation. Roads have known speed limits related to their width (as consistent with practice), are assumed to have constant heading, and permit two way travel. Large avenues have a speed limit of 35 mph and smaller roads have a speed limit of 25 mph. The target is restricted to ± 10 mph of the

specified road speed limit to simulate a realistic traffic scenario. Although the same map is used throughout this work, the particle filter framework is suitable for any road network [29].

3.1.1 Target Motion Model

It is known that the target vehicle is located within the urban environment, with no bias given to any particular road initially. Therefore, the initial particle distribution, $p(x_0)$, consists of N particles of equal weight scattered throughout the roadways. The x and y locations of each particle is drawn from the distribution $1000\mathcal{U}[0,1]$, under the restriction that the point $[x, y]$ lies within a road. A uniform distribution was used to create the initial particle distribution because there is equal probability that the target lies in any location throughout the map.

for $i = 1 : N$ **do**
  $rand \sim \mathcal{U}[0,1]$
  $x_k^i = 1000 rand$
  $rand \sim \mathcal{U}[0,1]$
  $y_k^i = 1000 rand$
  $point = [x_k^i, y_k^i]$
  while $point \in$ Obstacle **do**
    $rand \sim \mathcal{U}[0,1]$
    $x_k^i = 1000 rand$
    $rand \sim \mathcal{U}[0,1]$
    $y_k^i = 1000 rand$
    $point = [x_k^i, y_k^i]$
  end while
  $w_k^i = 1/N$
end for

It is assumed that the target is a mobile ground vehicle that is modeled as a simple car illustrated by Fig. 3.2 [30], with position [x,y], wheelbase L, turn angle ϕ, and heading θ measured clockwise from the positive y-axis.

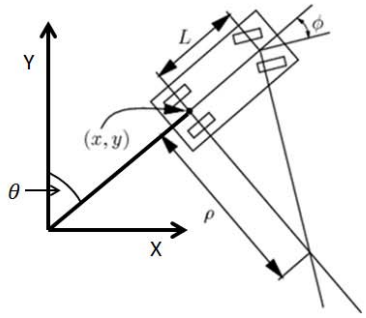


Figure 3.2: Geometry of a Simple Car

$$\rho_{min} = \frac{L}{\tan \phi_{max}} \tag{3.1}$$

The wheelbase, L, is set to 2.8 m and a minimum turning radius of 11.5 m is utilized throughout the simulation, as consistent with a standard sedan. Equation (3.1) is used to compute a maximum turning angle, ϕ_{max}, of 13.7°.

The target is required to remain on the right side of the road and stay within ± 10 mph of the road specific speed limit. The speed at each time step is updated with noise to account for unknown accelerations. The value for the noise, ν_k, is a random draw from the distribution $\mathcal{U}[0, 1]$.

$$V_{k+1} = V_k + \frac{V_k(\nu_k - 0.5)}{2} \tag{3.2}$$

The resulting range of possible values for V_{k+1} is the set $[0.75V_k, 1.25V_k]$. If the speed escapes the bounds of ± 10 mph of the speed limit, it is adjusted by the following control sequence:

$V_{min} = V_{road} - 10$

$V_{max} = V_{road} + 10$

while $V_{k+1} < V_{min}$ **do**

$V_{k+1} = V_k + V_k\,|(\nu_k - 0.5)/2|$

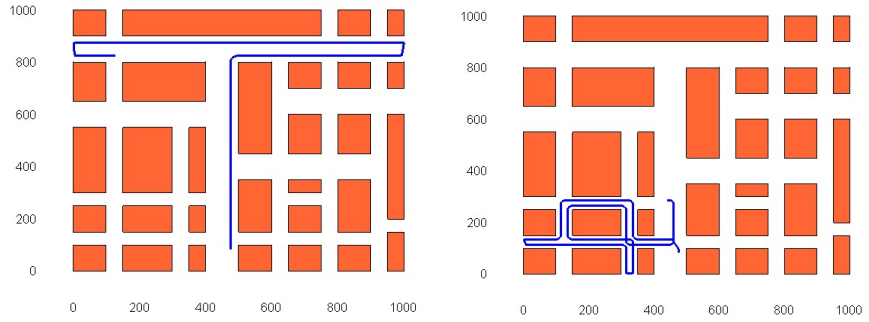
while $V_{k+1} > V_{max}$ **do**

$V_{k+1} = V_k - V_k\,|(\nu_k - 0.5)/2|$

end while

end while

In addition, the target may not reverse and must remain within the simulated map, and therefore is required to make a U-turn at the map's edge. Although the target's path is unknown to the estimation routine, two candidate paths were chosen for simulation, one predominantly straight and one winding, and are illustrated by Fig. 3.3.



(a) (b)

Figure 3.3: (a) Straight Path and (b) Winding Path

3.1.2 Measurement Model

It is assumed that a soft measurement source (a measurement from a human) is able to provide information on each 200 m × 200 m sector depicted by the grid squares in Fig. 3.4, and that there is no cost associated with utilizing any sensor. Every three seconds, a measurement is taken at the sector containing the highest particle weight sum. The highlighted square region in Fig. 3.5 represents the sector being measured. The structure of the sensor grid was arbitrarily chosen. The filter framework is also applicable to overlapping

sensor regions, sparse sensor coverage, non-stationary sensors, and sensor regions of irregular shape. In addition, the measurement report rate of one measurement per 3 seconds was chosen to observe the effect of receiving reports at a reduced frequency. Any frequency could be used in this framework. Finally, only one sensor region was polled per measurement. However, if in practice more regions may be polled at once, the particle filter framework presented herein is suitable for such an application.

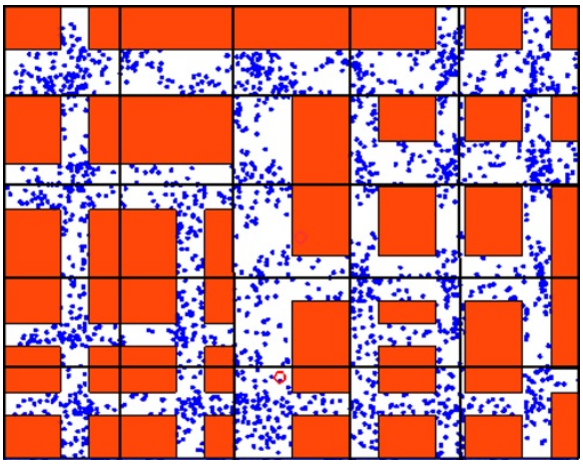


Figure 3.4: Determine Region of Highest Particle Weight

The polled sensor returns a binary response, where 0 means the target is not present and 1 means the target is present somewhere in the associated region. No specific location within the region is given. This information is fused into the measurement likelihood function to update particle weights. A sensor that always reported the truth was initially used. However, the assumption of a perfect sensor is unrealistic, especially as the sensor is modeled after a human operator. Because of this, false positive and false negative rates of 10 % were later introduced into the sensor model. Methods used to account for an imperfect soft binary sensor will be discussed in Chapter 4.

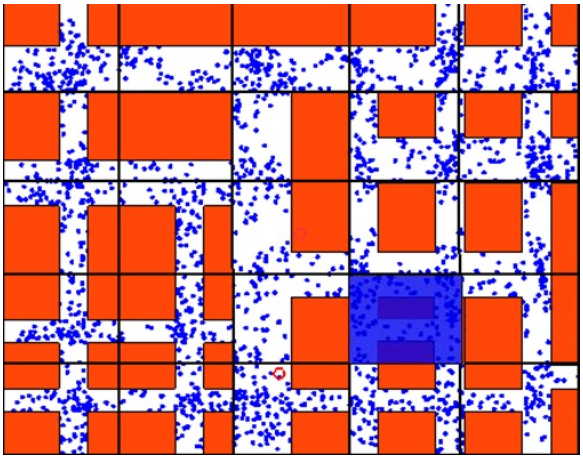


Figure 3.5: Measure Region of Highest Particle Weight

3.2 Dispersion Motion Model

The effect of the particle motion model was the first major investigation in this work. The initial particle dynamic model was that of a constant-velocity point-mass unicycle, as illustrated by Figure 3.6 where r = 0. This model caused the particles to randomly disperse through the road network until they reached an obstacle, at which point their heading was rotated 180 deg. The dispersion model was chosen for comparison because of its ease of implementation and its tendency to fill the roadway. A similar model was used by Kreucher et al. in [23] to track a target with the same dynamic model on a random walk, with no obstacles obstructing its motion. The following equations describe the particle motion in the dispersion model:

$$x^i_{k+1} = x^i_k + V^i_k \sin\theta^i_k dt \tag{3.3}$$

$$y^i_{k+1} = y^i_k + V^i_k \cos\theta^i_k dt \tag{3.4}$$

$$V^i_{k+1} = V^i_k \tag{3.5}$$

$$\theta^i_{k+1} = \theta^i_k + (\nu_k - 0.5)\frac{\pi}{10} \tag{3.6}$$

where the ith particle's position at time k is denoted $[x_k^i, y_k^i]$, its heading θ_k^i is measured clockwise from the positive y-axis, the time step dt is 1 second, and the noise ν_k is a random draw from the distribution $\mathcal{U}(0,1)$, thus limiting the change in heading per second to the range $[-9°, 9°]$.

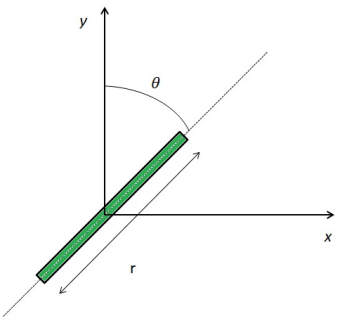


Figure 3.6: Geometry of Unicycle

3.3 Traffic Motion Model

In addition to the dispersion model, an alternative first-order Markov model was developed to exploit the knowledge of vehicle tendencies in an urban environment. In an urban setting, state variables may not be solely time dependent, but also location dependent. The traffic model accounts for road conditions such as in-lane travel, speed limits, two-way traffic, changing lanes to execute a future turn, and pauses at intersections. A complete characterization of the road network is assumed available. This requires categorizing regions into four categories: road, intersection, obstacle, or off-map. Each road has two possible headings that reflect two-way traffic. The required heading is based on a particle's position on the road, as to mimic two way traffic flow and driving on the right side of the road. Figure 3.7 illustrates the required headings of two roadways and their intersection at the grey region. Each particle stores a variable in its state vector that is related to the required heading of the road it is on, or the required heading of the road it is turning on to. This variable, q, becomes important during a turning maneuver.

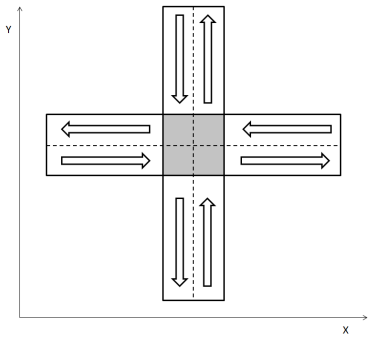


Figure 3.7: Intersection of Two Roads

Because state variables in an urban setting are both time and location dependent, each particle stores a variable in its state that corresponds to the motion model it is to use. The value of this variable m_k^i is based on whether the particle is on a road, going straight in an intersection, turning, left, turning right, or making a U-turn. The motion models for each of these modes are described below.

While on the rth road with speed limit V_r and $m_k^i = 2$, the ith particle is propagated forward in time using the following discrete-time dynamic model:

$$x_{k+1}^i = x_k^i + V_k^i \sin\theta_k^i dt \tag{3.7}$$

$$y_{k+1}^i = y_k^i + V_k^i \cos\theta_k^i dt \tag{3.8}$$

$$V_{k+1}^i = V_k^i + \frac{V_k^i(\nu_k^i - 0.5)}{2} \tag{3.9}$$

$$\text{Subject to } V_r - 10\text{mph} \leq V_{k+1}^i \leq V_r + 10\text{mph} \tag{3.10}$$

$$\theta_{k+1}^i = \theta_k^i \tag{3.11}$$

where x_k^i and y_k^i define the position relative to the origin, which is located in the bottom left corner of the map in Fig. 3.1, θ_k^i is the heading angle measured clockwise from the positive y-axis, the time step dt is 1 second, and ν_k is a random draw from the distribution $\mathcal{N}(0,1)$. Because the road network used in this work is modeled after a grid-like street pattern, the roadways are straight and a constant heading is desirable along them. However, a directed

graph could be used to determine a road's heading in order to apply this model to any road network [29].

As mentioned in Section 3.1.1, the target is required to make a U-turn at the map's edge. All particles are subject to the same restriction. If when propagated forward to the next time step, a particle would be located outside of the map's boundaries, a U-turn is performed. During this maneuver, it is necessary to know which road the particle is on so that the proper heading and position may be obtained following the U-turn.

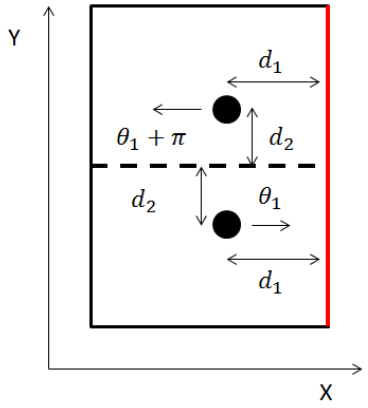


Figure 3.8: U-turn course adjustment

Figure 3.8 illustrates how a particle's position and heading are adjusted when it is to cross the map's boundary, drawn in red. In this example, the center of the road is the dashed line and the particle is initially on the lower half of the road, heading to the right. During the U-turn, it moves to the other side of the road. The distance between the particle and the edge of map is d_1, and the distance between the particle and the center of the road is d_2. The particle's proximity to the center of the road is preserved following a U-turn as that distance is used to determine what turns a particle is capable of performing.

The heading variable is changed by π radians and the x and y position variables are adjusted based on heading and the values of d_1 and d_2, respectively. The equations for the U-turn motion model depend on the particle's initial heading. The equations below are used during the U-turn motion model for the example in Fig. 3.8 where the particle's initial

heading is $\pi/2$. The three additional sets of equations for the U-turn motion model are derived in a similar manner.

$$x^i_{k+1} = x^i_k \tag{3.12}$$

$$y^i_{k+1} = y^i_k + 2d_2 \tag{3.13}$$

$$V^i_{k+1} = V^i_k + \frac{V^i_k(\nu^i_k - 0.5)}{2} \tag{3.14}$$

$$\text{Subject to } V_r - 10\text{mph} \leq V^i_{k+1} \leq V_r + 10\text{mph} \tag{3.15}$$

$$\theta^i_{k+1} = \theta^i_k + \pi \tag{3.16}$$

If a particle will be located in an intersection at the next time step, it must make a decision on which way to go based on what is permitted by the road network. At the start of the simulation, a list of possible turns from every direction at each intersection is generated based on the map. This turn list provides for obstacle avoidance. When a particle enters an intersection, its location is used to determine which intersection it is in and its heading is used to determine the turning possibilities. The list of possible turn choices is compiled and a random selection is made to indicate the chosen turn direction, either left, right, or straight. Next, the particle's proximity to the obstacles on its right and left, velocity, span-wise location on its current road, and the width of all possible future roads are taken into account to determine what adjustments must be made to complete the chosen turn. Figure 3.9 illustrates the various parameters that are taken into account to prepare for a turn, where the particle's initial position is in the bottom right with a heading of $\theta = 0$ in the positive y direction.

First, the width of the possible future road, denoted as r_2 in Fig. 3.9, is calculated. In the map used in this simulation, the width of the road is determined by the size of the intersection and is assumed constant along the length of the road. If either r_{right} or r_{left} are greater than r_2, a lane change would be necessary to successfully make the turn in their respective directions. In the example illustrated by Fig. 3.10, the ith particle at position

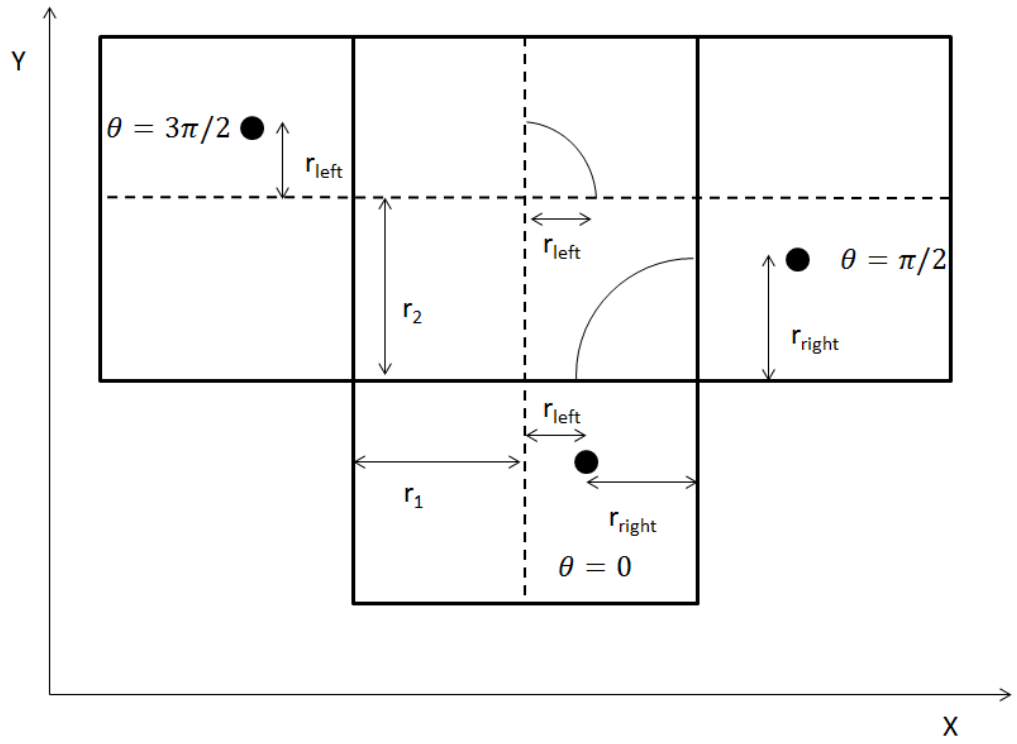


Figure 3.9: Geometric characterization of an intersection

$[x_k^i, y_k^i]$ has chosen to turn right, but its distance from the nearest obstacle to the right, r_{right}, is larger than half of the width of its future road, r_2. If it were to turn in its current position, it would be on the wrong side of the road when it reached the new road. Therefore, it must merge to the right. The particle will merge to its new distance from the obstacle, r_{new}, which is equal to $\rho_{min} + 1$.

$$x_{k+1}^i = x_{1,obs} - (\rho_{min} + 1) \tag{3.17}$$

The lane change maneuver depends on the chosen turn direction and the particle's heading. Similar update equations are derived for the remaining configurations.

Lane changes are permitted when approaching an intersection to provide for improved tracking. This was motivated by the resampling step in the particle-filter framework. As a result of resampling, particles of significant weight are duplicated into multiple particles of smaller weight. Because the noise in the velocity term will only separate duplicated particles

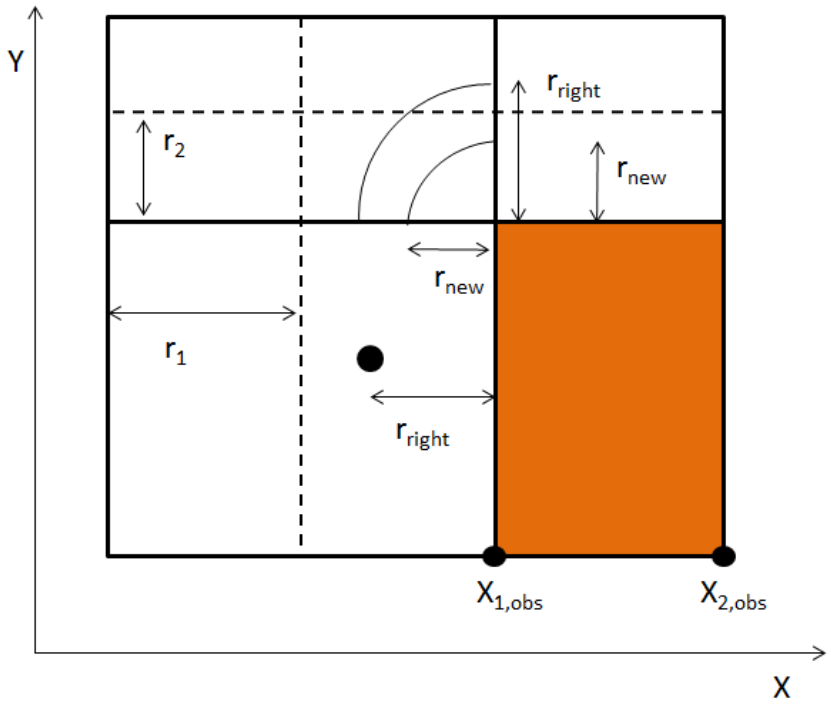


Figure 3.10: Example of a Required Lane Change

along the length of a road and not along the width, lane changes allow the particle cloud to more fully describe the possible motion of a target vehicle entering an intersection, without violating the simulated vehicle's physical constraints, namely ρ_{min}. The appropriate lane is determined by the particle's velocity and chosen turn direction.

Once a turn direction is randomly chosen and a lane adjustment is made, if necessary, the particle's turning parameters are computed. While turning, a particle's heading will change a total of $\pi/2$ and the particle's path during a turn will follow an arc of constant radius, ρ^i, which is the distance between the ith particle an the nearest obstacle in its chosen turning direction. The two choices for ρ^i are r_{right} and r_{left}, as illustrated by Fig. 3.9.

Because the dynamic system is a first order Markov process, the future state depends only on the current state. Therefore, only the state at time k is stored. In addition, the time step used in the simulation is 1 second. This presented a challenge during turning maneuvers, as the amount of time it takes to complete a turn is not always an integer or

constant. This required the use of the aforementioned variable q to determine when the appropriate amount of turning had occurred. The value of a particle's $q \in [1, 2, 3, 4]$ and it is related to a particle's heading by Eq. (3.18) and is illustrated by Fig. 3.11.

$$\theta_k^i = (q_k^i - 1)\frac{\pi}{2} \tag{3.18}$$

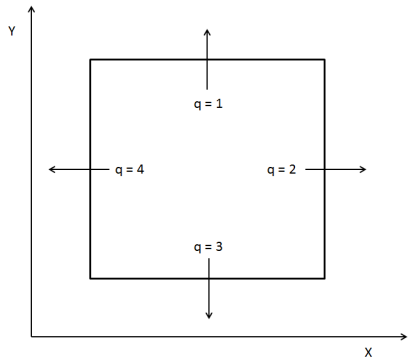


Figure 3.11: Definition of q Variable

If a particle chooses to turn right, its mode variable m_k^i becomes 1. Similarly, if a particle chooses to turn left, m_k^i is set to -1. When a particle makes a decision to turn, q_k^i is updated by use of the mode variable m_k^i, as indicated in Eq. (3.20), and is always restricted to the set $[1, 2, 3, 4]$. The value of q_k^i is used to update a particle's heading during a turn. While on a road, q_k^i remains constant as the heading is constant.

$$q_k^i = q_k^i + m_k^i \tag{3.19}$$

$$q_k^i = \text{mod}(q_k^i, 4) \tag{3.20}$$

Given the selected turn direction and turning radius, the number of time steps to complete a turn, N_s, is then computed based on the arc angle to be traversed per time step, α. Because the number of time steps to complete a turn depends on the velocity, no noise is introduced

into the velocity during a turn, thus keeping it constant.

$$\tan \phi_k^i = \frac{L}{\rho_k^i} \tag{3.21}$$

$$\alpha^i = \frac{mV_k^i}{L} \tan \phi_k^i \tag{3.22}$$

$$N_s^i = \left\lfloor \frac{\pi/2}{\alpha^i} \right\rfloor + 1 \tag{3.23}$$

Figure 3.12 illustrates the parameter α that is utilized during a sample turn where $N_s = 5$, $m = 1$, and the value of q changes from 1 to 2. The position of the ith particle along the arc during a turn at each time step k is illustrated by the black dot.

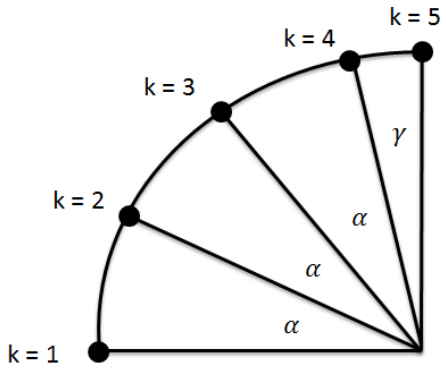


Figure 3.12: Geometry During a Sample Turn

The angle γ^i is always less than or equal to α^i as it is the remainder of the arc to be traversed at the final time step of a turn. This simulates steering to stay in the same lane.

$$\gamma^i = \frac{\pi}{2} - m\alpha^i(N_s - 1) \tag{3.24}$$

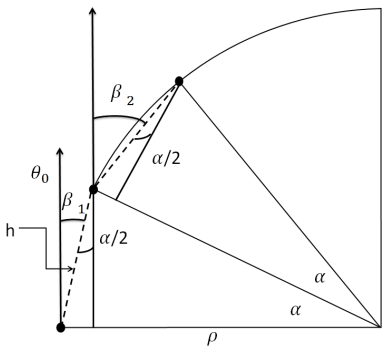


Figure 3.13: Geometric Turning Parameters

Once the above parameters are determined, the time update parameters illustrated by Fig. 3.13 are determined. The time dependant variable β_k is the angle between the heading at which the particle entered the intersection, θ_{k0}, and the next way point along the arc, $[x_{k+1}, y_{k+1}]$. The constant h is the distance between $[x_k, y_k]$ and $[x_{k+1}, y_{k+1}]$. These parameters are computed by the following set of equations, where θ_{k0} is obtained by determining the previous value of q:

$$\theta_{k0} = [(q-m)-1]\frac{\pi}{2} \tag{3.25}$$

$$\beta_k = \theta_{k0} + [(k-1)+1/2]\alpha \tag{3.26}$$

$$h = \rho \mathrm{m} \sin(\alpha)/\cos(\frac{\alpha}{2}) \tag{3.27}$$

Finally, the state variables for the ith particle are updated by the algorithm below for the sample turn in Fig. 3.12. The variable k is stored in each particle's state vector in order to keep track of how far into a turn the ith particle is.

if $k^i < N_s$ **then**

$\beta_k^i = \theta_{k0}^i + [(k-1)+1/2]\alpha^i$

$x_{k+1}^i = x_k^i + h^i \sin(\beta_k^i)$

$y_{k+1}^i = y_k^i + h^i \cos(\beta_k^i)$

$\theta_{k+1}^i = \beta_k^i$

$k^i = k^i + 1$

else

$x^i_{k+1} = x^i_k + \rho^i_k \sin\gamma^i$

$y^i_{k+1} = y^i_k + (x^i_{k+1} - x^i_k)\tan(\gamma^i/2)$

$theta^i_{k+1} = (q-1)\pi/2$

$k^i = 0$

end if

The updated x and y position during a turn is calculated in a Markov process by trigonometric relationships that include quadrant adjustments. The same update equation is used for all steps before the final, and the final step ensures the vehicle completes the in-lane turn and enters the road with proper heading. Once a particle finishes turning, it's mode variable, m, is set to zero until it exits the intersection and enters a road at a constant heading.

3.4 Measurement Update

It is assumed that there is a sensor that may be polled in each 200 m $\times$ 200 m region and that each sensor can view its entire region, as illustrated by Fig. 3.14. Only one sensor may be polled per measurement, therefore a metric of sensor selection must be established. The region of highest importance must be determined at each measurement interval in order to determine which sensor to poll. In this work, it was decided that the region with the highest particle weight sum was the most important and should be observed. A measurement was taken every three seconds.

At each measurement, the sector with the highest particle weight sum is polled by a sensor. The results presented in this chapter reflect a true sensor report at each measurement update. The effect of introducing false sensor reports is discussed in Chapter 4.

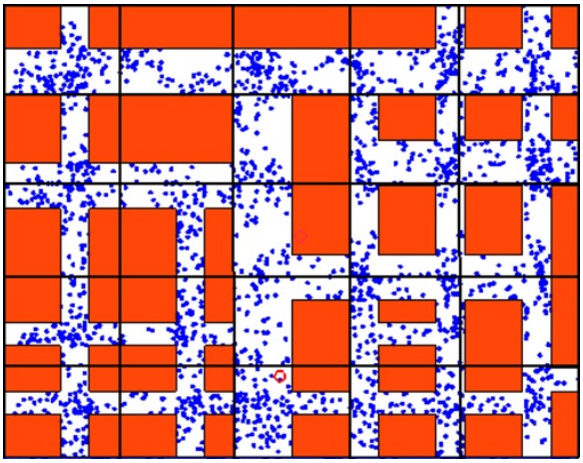


Figure 3.14: Determine Region of Highest Particle Weight

3.4.1 Particle Weight Update

In general, the measurement update step will adjust each particle's weight according to the information the measurement provides and the confidence level the particle filter has in the measurement source. As mentioned in Section 2.2, when a measurement, $\mathbf{z_k}$, is received, the state *pdf* is updated through the following likelihood update:

$$p\left(\mathbf{x}_k \mid \mathbf{z}_k\right) = \frac{p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)}{p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)} \tag{3.28}$$

The values for each term are determined based on the sensor's report and whether a particle is in the measured region or not. A sample measurement update of a smaller environment will be detailed for completeness. Suppose the measurement regions are divided as illustrated by Fig. 3.15, where the eight circles are particles of equal weight (1/8), and the star is the target. The region with the highest particle weight is the region in the bottom right corner, so it will be polled by the sensor. The sensor finds the target within the region. The likelihood function for each particle is computed based on its location. There are $N = 8$ particles and the array of particle weights is $\mathbf{w}_k^{i,-} = 1/8, i = 1, ..., N$.

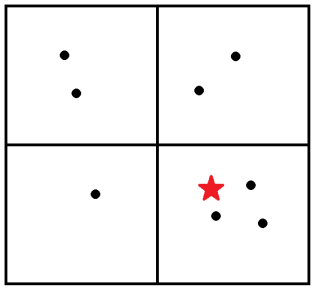


Figure 3.15: Measurement Update Example Scenario

In general, when the sensor reports a detection, the particles in the measured region (the bottom right corner) are updated as follows, where - corresponds to before a measurement is taken and + corresponds to after a measurement is taken.

$$p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) = 1 - R_{false} \tag{3.29}$$

$$p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right) = \sum_{i=1}^{N_{in}} w_{in}^{i,-} \tag{3.30}$$

$$p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right) = (1 - R_{false}) \sum_{i=1}^{N_{in}} w_{in}^{i,-} + \tag{3.31}$$

$$(R_{false})(1 - \sum_{i=1}^{N_{in}} w_{in}^{i,-}) \tag{3.32}$$

$$p\left(\mathbf{x}_k \mid \mathbf{z}_k\right) = \frac{p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)}{p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)} \tag{3.33}$$

$$w_k^{i,+} = \frac{w_k^{i,-} p\left(\mathbf{x}_k \mid \mathbf{z}_k\right)}{\sum \mathbf{w}_{in}^{-}}$$

In the equations above, $\mathbf{z}_k$ is the measurement report at time step k, $\mathbf{x}_k$ is the state of the *pdf* at time k, $\mathbf{Z}_{k-1}$ is the set of all measurements up to time k-1, $\mathbf{w}_{in}$ is the array of weights of the particles in the measured region, and R_{false} is the is the percentage of times the particle filter believes that the sensor reports it has found the target when it is actually not present. In this example, $\sum \mathbf{w}_{in}^{-} = 3/8$ and the false positive rate is zero. This results in the new weights for the particles in the lower right corner to each be $w_{in}^{i,+} = 1/3, i = 1, 2, 3$.

Similarly, when the sensor reports a detection, the particles outside of the measured region (the two top sections and the bottom left corner) are updated as follows:

$$
\begin{aligned}
p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) &= R_{false} && (3.34)\\
p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right) &= \sum_{i=1}^{N_{out}} w_{out}^{i,-} && (3.35)\\
p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right) &= \left(1 - R_{false}\right) \sum \mathbf{w}_{in}^{-} + && (3.36)\\
& \quad \left(R_{false}\right)\left(1 - \sum \mathbf{w}_{in}^{-}\right) && (3.37)\\
p\left(\mathbf{x}_k \mid \mathbf{z}_k\right) &= \frac{p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)}{p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)} && (3.38)\\
w_k^{i,+} &= \frac{w_k^{i,-} p\left(\mathbf{x}_k \mid \mathbf{z}_k\right)}{\sum_{i=1}^{N_{out}} w_{out}^{i,-}}
\end{aligned}
$$

where w_{out} is the array of weights of the particles outside of the measured region. This results in the weights of the particles not located in the measured region to each be set to zero. A similar formulation including a false negative rate is used when a sensor reports a non-detection.

In the work discussed in Section 3.5, it is assumed that the sensor always reports the truth. Therefore, if a sensor reports that the target is present within a region, the weights of all the particles located outside of that region are set to zero. Similarly, if a sensor reports that the target is not present within a region, the weights of all the particles located inside of that region are set to zero.

The resulting particle distribution is illustrated by Fig. 3.16, where larger particles hold more weight than smaller particles and the numbers indicate the ith position of the particle in the array $\mathbf{w}_k = w_k^i, i = 1, ..., N$. The effect of including a false report rate belief in the particle filter's measurement update step is introduced in Section 3.6.2 and discussed in detail in Chapter 4.

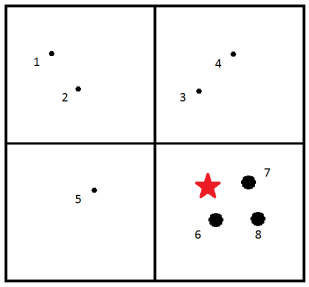


Figure 3.16: Post-measurement Update for Example Scenario

3.4.2 Resample Step

Because the sensor model used in this chapter is believed to always report the truth, resampling is done after each measurement update, resulting in $N_{\text{eff}} = N$. The resulting weight array in the sample used discussed in the previous section is $\mathbf{w}_k = [0, 0, 0, 0, 0, 1/3, 1/3, 1/3]$, where the last three elements are the weights of the three particles in the bottom right corner. The resampling algorithm given in Section 2.2 is used to break up the large particles in to smaller ones and to eliminate particle of insignificant weight. Figure 3.17 illustrates the particle distribution following resampling. The three particles in the bottom right corner were duplicated and the particles with zero weight outside of the measured region were eliminated. After resampling, duplicate particles will have the same state, and their weights become equal following normalization.

for $i = 1 : N$ **do**

$w_k^i = \frac{w_k^i}{\sum_{j=1}^{N} w_k^j}$

end for

The particles are separated in Fig. 3.17 for visualization purposes. In the urban tracking simulation, duplicate particles obtain separation over time through the noise terms in the particle-motion model.

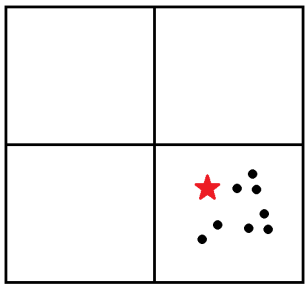


Figure 3.17: Resampled Distribution for Example Scenario

3.5 Motion Model Comparison Results

Monte Carlo simulations were conducted to study the effect of particle motion on the time to detect and tracking performance when the target path was varied between a straight path (Fig. 3.3 (a)) and a meandering one (Fig. 3.3 (b)). The time to detect is defined as the amount of simulated time that elapses before a measurement is received indicating that the target has been found for the first time. The system is observed for 60 seconds following the aforementioned initial target detection time. The tracking performance is measured by the mean square error. The mean square error is computed after each time step by summing the product of the weight of each particle with the square of its distance from the target.

$$e_{MSE}(t) = \sum_{i=1}^{N} \left[\left(x_T - x_k^i\right)^2 + \left(y_T - y_k^i\right)^2 \right] w^i \tag{3.39}$$

Following the last run of a trial consisting of N_{runs} total runs, the mean square error is averaged at time t after detection according to:

$$e_{ave}(t) = \frac{\sum_{j=1}^{N_{runs}} e_{MSE,j}(t)}{N_{runs}} \tag{3.40}$$

It is important to note that the mean square error establishes an approximation to the area that the particles cover. The area covered by a regional sensor, 4 x 10^4 m^2, can be used as a metric of comparison.

Two hundred Monte Carlo runs were completed for each configuration, each with identical initial target position, velocity, and heading. The four configurations detailed below consist of the two aforementioned motion models with two possible values of the total number of particles. All simulations were done in MATLAB on a standard desktop computer with a 2 GHz processor, and values for time to detect, tracking performance, and simulation time are presented as the average of the Monte Carlo runs. It is presented in Table 3.1 that

Table 3.1: Time to Detect Results for Straight Path, Perfect Sensor

Particle Model	**Number of Particles**	**Time to Detect**
Dispersion	500	51.09 sec
Traffic	500	36.13 sec
Dispersion	1000	53.59 sec
Traffic	1000	29.31 sec

Table 3.2: Time to Detect Results for Winding Path, Perfect Sensor

Particle Model	**Number of Particles**	**Time to Detect**
Dispersion	500	71.82 sec
Traffic	500	56.38 sec
Dispersion	1000	66.13 sec
Traffic	1000	69.54 sec

the Traffic Motion Model provides for faster detection times the case of a straight path and in the case of a winding path with 500 particles. Whereas, the Dispersion Motion Model provides for faster detection time in the case of a winding path with 1000 particles according to the results in Table 3.2. For comparison, searching randomly through the 25 sensor regions would provide an expected time-to-detect of 75 seconds. The benefit of the Traffic Motion Model is clear after observing the tracking performance results in Figs. 3.18 and 3.19. In the case of a straight path, after approximately, 20 to 25 seconds, the model difference is evident by high error divergence in the Dispersion Model results and reduced divergence in

the Traffic Model results for both both $N = 500$ and $N = 1000$. This divergent behavior was investigated further and is discussed in Section 3.6.1. The Dispersion Model yields better tracking performance in the case of a winding path, as the particles in the Dispersion Model tend to not traverse in a straight path. As expected, increasing the number of particles generally helps to improve the time to detect and tracking performance, with the exception of the Traffic Motion Model in the case of a winding path. This will be further investigated in Chapter 4.

Table 3.3: Computational Expense Comparison

Particle Model	**Number of Particles**	**Time to Simulate One Second**
Dispersion	500	0.0375 sec
Traffic	500	0.0847 sec
Dispersion	1000	0.0755 sec
Traffic	1000	0.1698 sec

Although the Dispersion Model is less computationally expensive, the values in Table 3.3 indicate that run time of the Traffic Motion Model is manageable. The simulation time includes the time to simulate the target, sensor, and particle filter. Also, an important observation of these results is that the difference in simulation time scales linearly with the number of particles. Further investigation to decrease detection time for the Traffic Motion Model when the target is on a winding path is necessary and will be discussed in Section 3.6.2.

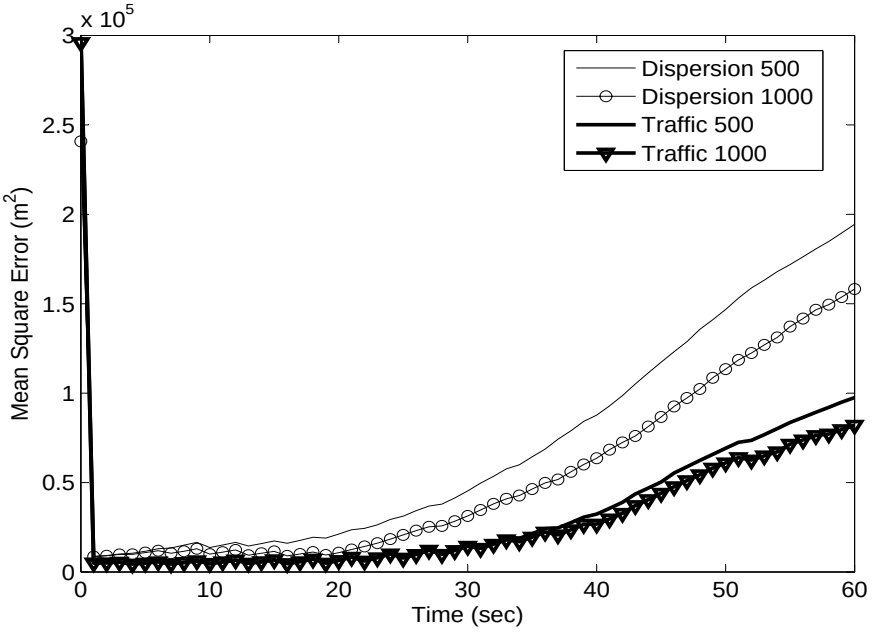


Figure 3.18: Average Mean Square Error After Target Found, Straight Path

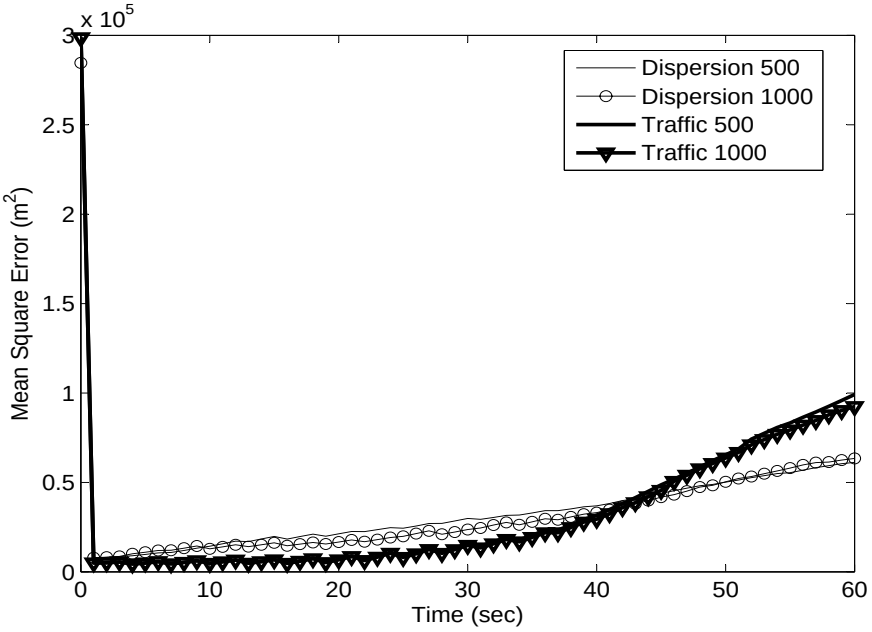
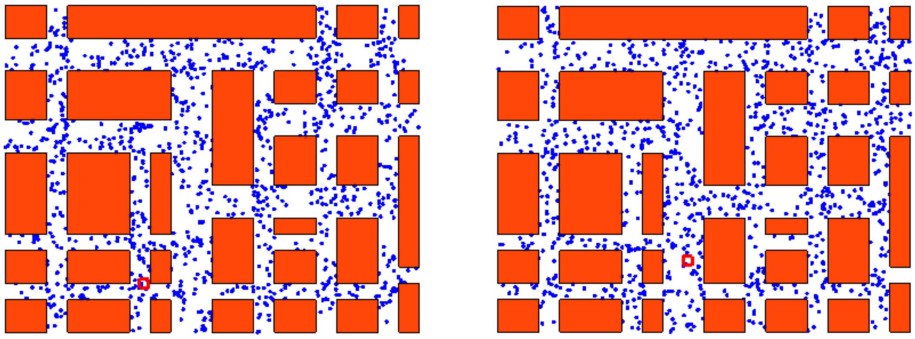


Figure 3.19: Average Mean Square Error After Target Found, Winding Path

3.6 Importance of Spatial Resolution

The disparity in the average time to detect values between the two candidate paths warranted a thorough investigation into the stability and behavior of the simulation. The stability of the Traffic Motion Model was studied by observing the spatial resolution of the particles at the start of the simulation and as time progressed, without any measurements. Figure 3.20 illustrates that the Traffic Motion Model is stable in that no clusters of particles form and an even distribution of particles is present at both the start of the simulation and as propagation progresses.



(a) (b)

Figure 3.20: (a) Time = 0 sec. (b) Time = 200 sec.

The sensor model used in this work is non-differentiable, non-smooth, and results in a challenging estimation problem. In addition, because the physical size of the measurement area is large, additional interest must be taken in the spatial resolution of the particle cloud. For instance, when the sensor reports that the target is found, no additional information is available concerning where in the measured region the target is located. If the particles within the measured region are not near the target, poor tracking performance may result. Therefore, the effect of redistributing the particles to fill the measured region following a "target found" report is described in Section 3.6.1.

Additionally, if the target is located just outside of the measured region, it is not found but many of the particles near it are eliminated. This increases the time to detect. The idea

of not eliminating all particles within a region of non detection was then studied. This was accomplished by allowing the particle filter to no longer assume the sensor always reports the truth, and instead has a false report rate. This is detailed in Section 3.6.2.

3.6.1 Particle Redistribution

The divergent nature of the mean square error presented in Section 3.5 was studied in an effort to provide for improved target tracking for the Traffic Motion Model. The statistical properties of the sensor were considered and the idea of particle redistribution was invented and investigated. Because it was assumed that the sensor was perfect, all particles outside of the region with reported target presence were being deleted. In addition, the location of the particles within the region of detection were not necessarily the best estimate of target location, as the sensed region is large (200 m $\times$ 200 m) and no indication is given to the target's location within the region. Therefore, when a positive target detection is reported, all particles are then redistributed throughout a 300 m square region about the center of the detection region, as illustrated by Fig. 3.21.

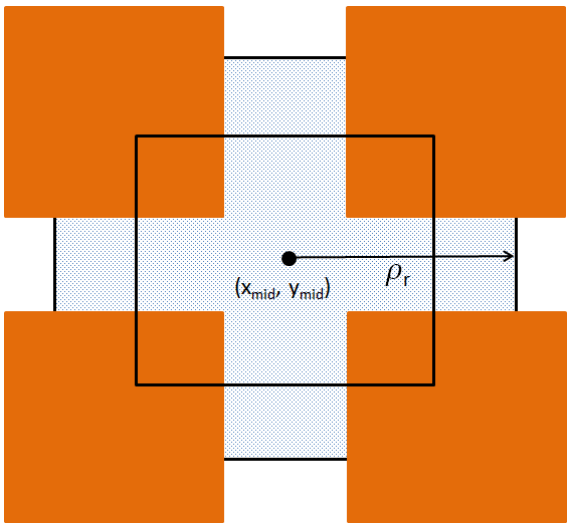


Figure 3.21: Redistribution Region

The center of the square measured region is located at the point $[x_{mid}, y_{mid}]$, and the particles are scattered through the roads within a 300 m square region about that point, excluding regions that are defined as obstacles, which are shown in orange in Fig. 3.21. The variable ρ_r is used to characterize the size of the region. The size of the redistribution region was chosen to extend slightly beyond the edge of measured region to improve tracking if the target is found near a the edge of the region. Redistribution about a certain point is done in a manner similar to creating the initial particle distribution, and is outlined by the algorithm below.

$\rho_r = 150$

for $i = 1 : N$ **do**

$rand \sim \mathcal{U}(0, 1)$

$x_k^i = (x_{mid} - \rho_r) + 2\rho_r rand$

$rand \sim \mathcal{U}(0, 1)$

$y_k^i = (y_{mid} - \rho_r) + 2\rho_r rand$

$point = [x_k^i, y_k^i]$

while $point \in$ Obstacle **do**

$rand \sim \mathcal{U}(0, 1)$

$x_k^i = (x_{mid} - \rho_r) + 2\rho_r rand$

$rand \sim \mathcal{U}(0, 1)$

$y_k^i = (y_{mid} - \rho_r) + 2\rho_r rand$

$point = [x_k^i, y_k^i]$

end while

$w_k^i = 1/N$

end for

The process of redistribution disregards the a priori *pdf* to reflect the new information from the sensor. If the target was reported found in a region, there was no guarantee that there was a sufficient particle distribution on the same road as the target or heading in

the same direction. The sensor report only indicated that the target was present within a region, similar to the beginning of the simulation where it is only known that a target is located within a 1000 m $\times$ 1000 m region of a city. Redistributing the particles to fill the sensed region was necessary given the presence of such a large regional sensor that was trusted completely. This provides for complete spatial particle resolution within the region of interest and resulted in improved tracking performance and prevented error divergence. The redistribution routine was implemented in two different methods. Method 1 redistributed particles at the first detection with $\rho_r = 150$ m, and at the first loss of detection following the initial detection, with $\rho_r = 175$ m. Following the first loss of detection, the redistribution area was increased to a 350 m square around the center of the region in which the target was last detected. Method 2 redistributed particles at the first detection only with $\rho_r = 150$ m.

The effect of redistribution on the tracking performance is illustrated by Figs. 3.23 and 3.24, where the redistribution methods are compared to the best performers without redistribution from Section 3.5. It is evident by Figs. 3.23 and 3.24 that Redistribution Method 2 significantly improves performance in the case of both straight and winding paths. The addition of particle redistribution did not significantly affect the computational run time, as it is only done once in the chosen method. The time to detect is not affected by the practice of redistribution as it only occurs after the target is found.

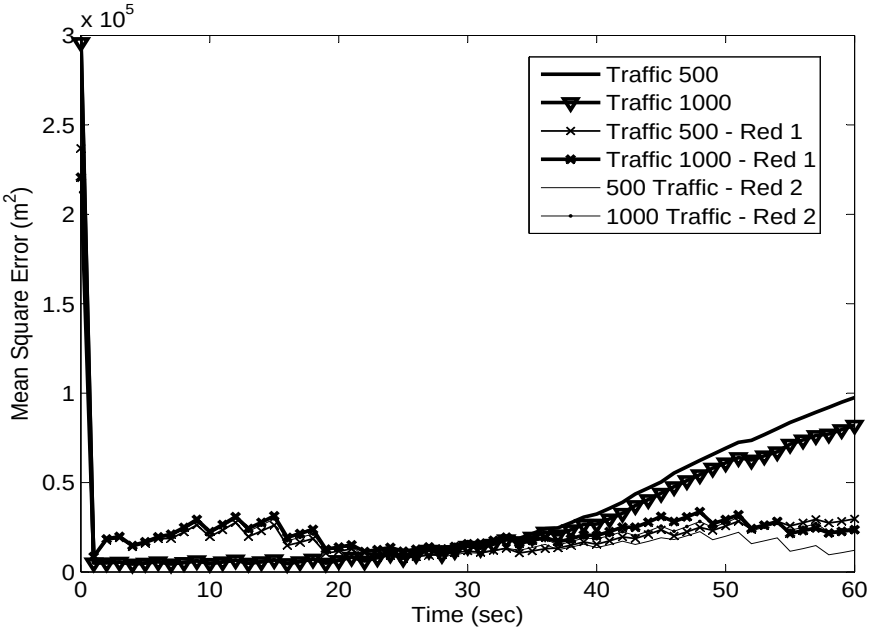


Figure 3.22: Average Mean Square Error After Target Found, Straight Path, Traffic Motion Model

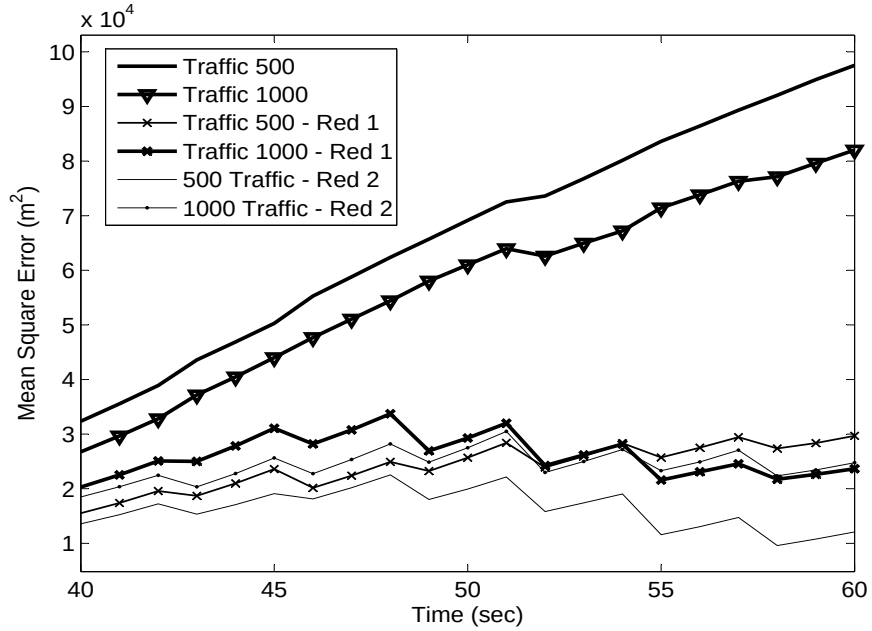


Figure 3.23: Average Mean Square Error After Target Found, Straight Path, Traffic Motion Model, Final 30 seconds

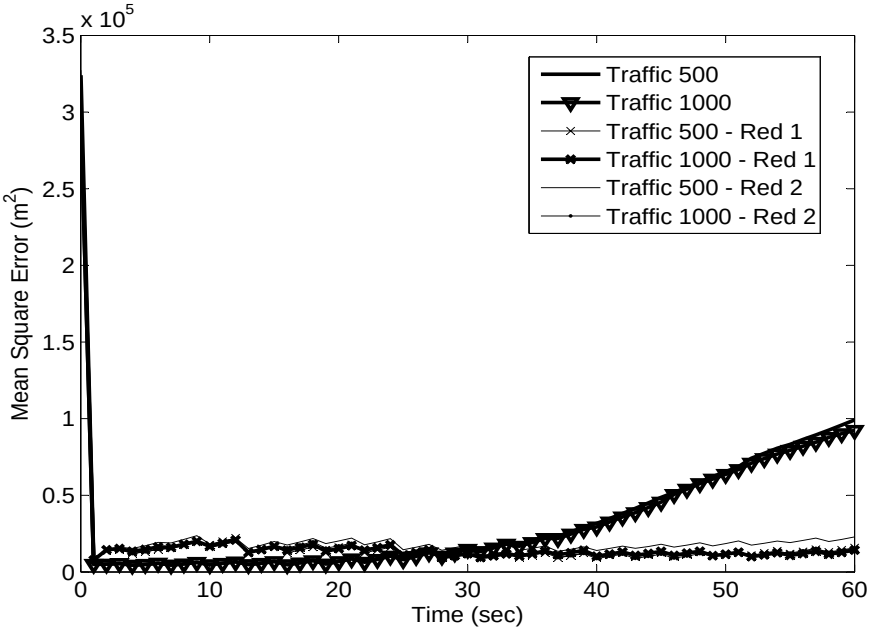


Figure 3.24: Average Mean Square Error After Target Found, Winding Path, Traffic Motion Model

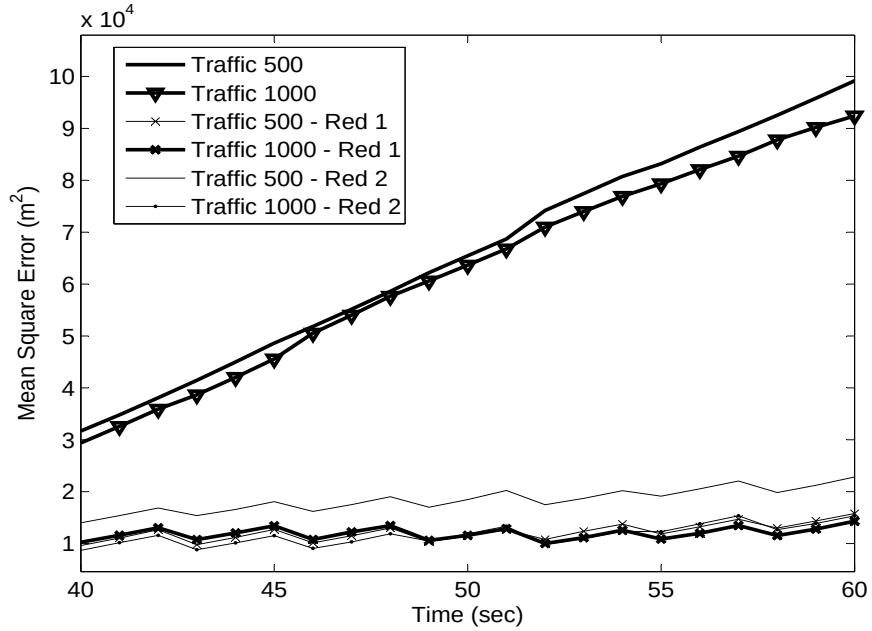


Figure 3.25: Average Mean Square Error After Target Found, Winding Path, Traffic Motion Model, Final 30 seconds

The practice of redistribution provides vital improvement to the tracking performance in the case of a target on a straight or a winding path. Because the true path of the target is not known a priori to the particle filter, redistribution should be included in the presence of a sensor model that covers such a large region. It should be noted that the area of each sensor region is $40,000\text{m}^2$, and by the inclusion of Redistribution Method 2 with 1000 particles, the mean square error is kept below that value in both path cases.

3.6.2 False Rate Inclusion

In an effort to decrease the time to detect the target, the idea of a false report rate was introduced into the particle filter's measurement update. The belief in a false report rate would decrease the particle filter's confidence in a measurement's report. This detuning of the measurement update would allow for the weight of particles in regions of non-detection to be decreased, but not set to zero. This practice would decrease the rate at which particles in regions of non-detection were eliminated due to resampling, and preserve some of the spatial resolution of the particle cloud. This was motivated by the observation that in some cases, if the target is located just outside of the measured region, it is not found but many of the particles near it are eliminated, which could lead to increases in the time to detect.

Figure 3.26 illustrates the effect that the particle filter's belief of the false report rate has on the particles in a region of non-detection. Each image is taken after the one measurement update and resample step, and the measurement region is the same in each scenario. It is observed that as the particle filter's confidence in the sensor's performance decreases, the resolution of particles in the region of non-detection is maintained. As expected, all particles in the region of non-detection are eliminated when it is believed that the sensor has no false report rate. In addition, if it is believed that the sensor has a 50% false report rate, no weight is shifted into or out of the region of non-detection, as the particle filter associates no benefit to any report from the sensor.

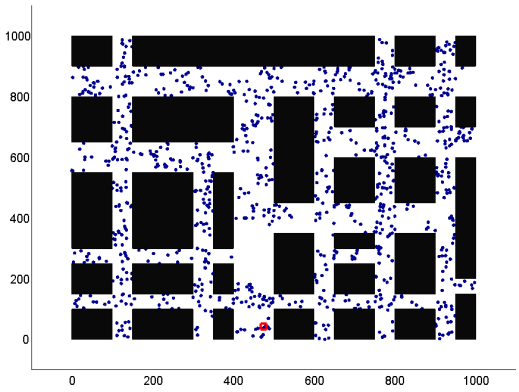
In addition, the effect of the false report belief was studied in the case of a target detection. Three beliefs are compared in Fig. 3.27 over the course of two measurements, one taken every three seconds. The particle cloud contracts to the measurement region much more quickly as the false report rate belief decreases. This is an important factor to consider when tuning a particle filter to estimate a state that is measured by a sensor with a known false report rate. The spatial resolution of the particle distribution is preserved for a longer period as the particle filter's confidence in the sensor is decreased. In the case of the report of a false detection, if the false report belief is properly tuned, the particle filter will maintain its spatial resolution following subsequent measurements that could correct the false report.

The tracking performance and time to detect where also investigated under the belief of a false report rate. In this study, the particle filter believed the sensor had false positive and false negative rates of 10%, however in reality, the sensor always reports the truth. This caused particle weights in regions of non-detection to be significantly reduced, but not always eliminated in the resampling process. Conversely, particles outside of a region of detection were no longer automatically eliminated until after repeated detection. The results listed below were obtained using 1000 particles, a sensor that always reported the truth, and a particle filter that believed the sensor had a 10% false report rate.

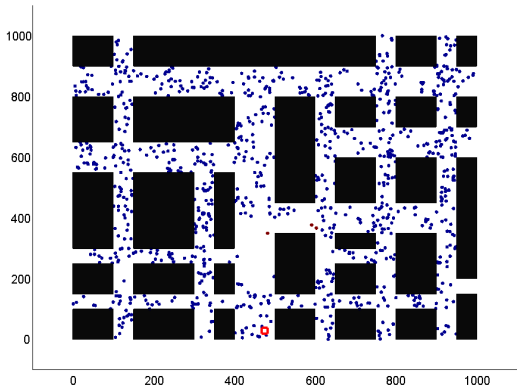
Table 3.4: Time to Detect Results for Straight Path, N = 1000

Particle Model	**Sensor Belief**	**Time to Detect**
Dispersion	Perfect	53.59 sec
Traffic	Perfect	29.31 sec
Dispersion	10% False Report	57.9 sec
Traffic	10% False Report	28.8 sec

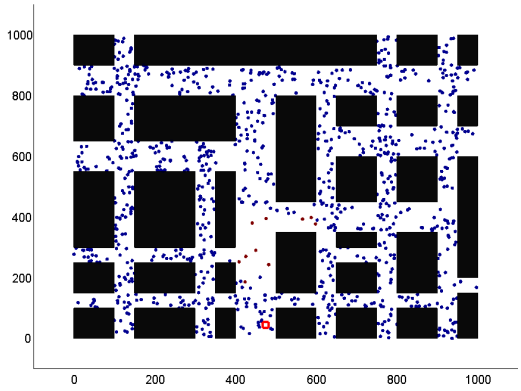
The times to detect results obtained in Section 3.5 changed slightly for both models in the case of a straight path, but there was a significant more difference in the case of the Traffic Motion Model on a winding path. The results in Table 3.4 indicate that the inclusion of the assumption of a false report rate in the presence of a perfect sensor decreased the



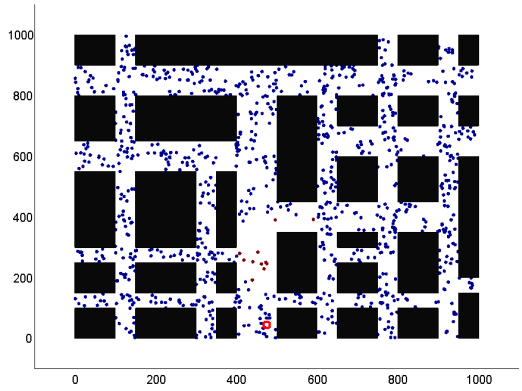
(a) 0% False Report Belief



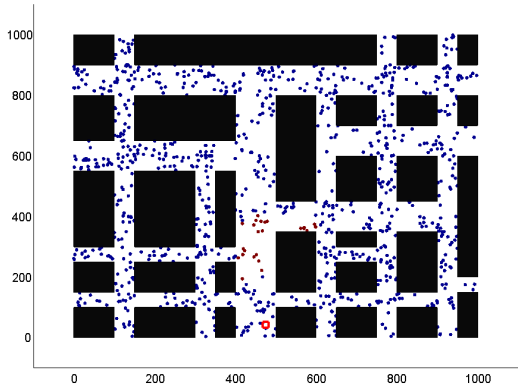
(b) 10% False Report Belief



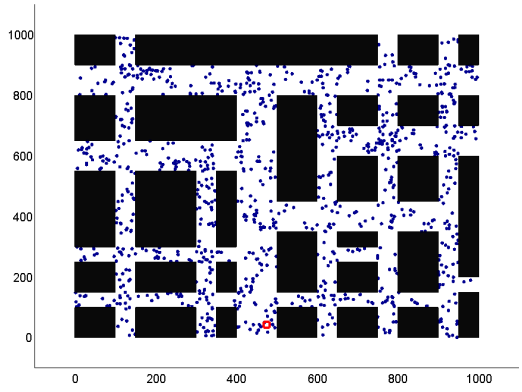
(c) 20% False Report Belief



(d) 30% False Report Belief

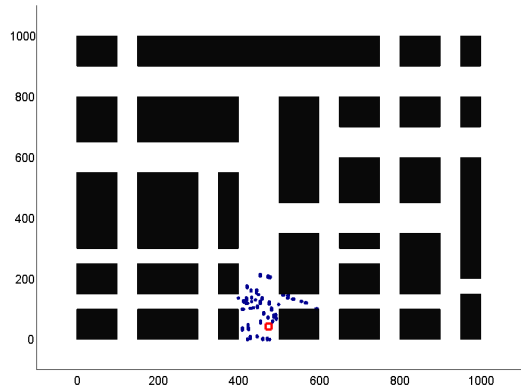


(e) 40% False Report Belief

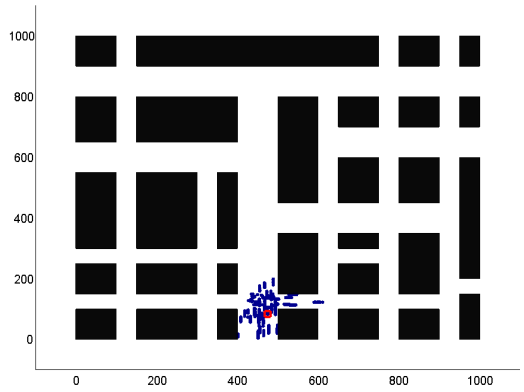


(f) 50% False Report Belief

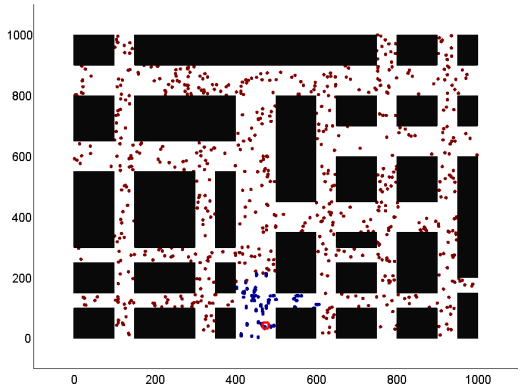
Figure 3.26: Effect of false report rate belief on particle spatial resolution and weight distribution



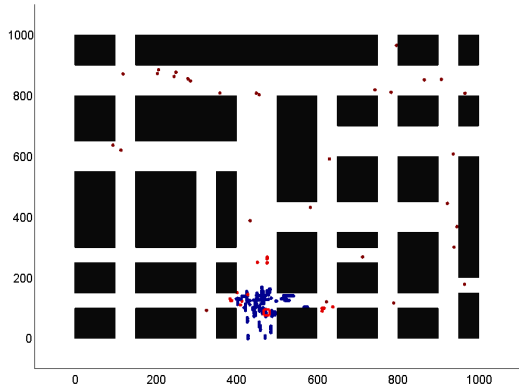
(a) 0% False Report Belief, after 1 measurement



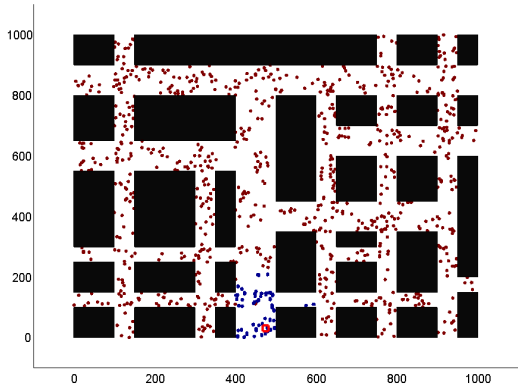
(b) 0% False Report Belief, after 2 measurements



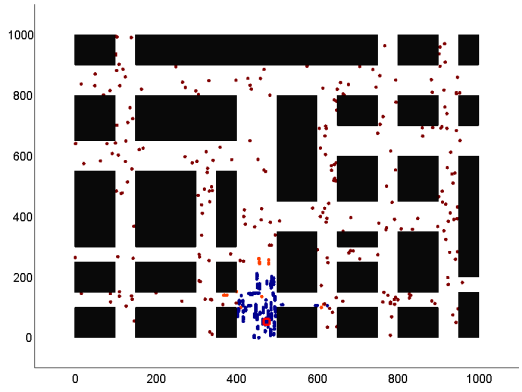
(c) 10% False Report Belief, after 1 measurement



(d) 10% False Report Belief, after 2 measurements



(e) 20% False Report Belief, after 1 measurement



(f) 20% False Report Belief, after 2 measurements

Figure 3.27: Effect of false report rate belief on particle spatial resolution and weight distribution

Table 3.5: Time to Detect Results for Winding Path, N = 1000

Particle Model	**Sensor Belief**	**Time to Detect**
Dispersion	Perfect	66.13 sec
Traffic	Perfect	69.54 sec
Dispersion	10% False Report	67.7 sec
Traffic	10% False Report	32.2 sec

time to detect for the Traffic Motion Model. This trend is also observed in the results in Table 3.5 with a much more significant improvement to the detection time for the Traffic Motion Model. In both cases, the Dispersion Model did not benefit from the inclusion of a false report rate belief. No significant change was observed in the time to detect for the Dispersion Model, further indicating the importance of a high fidelity model in the presence of an accurate sensor.

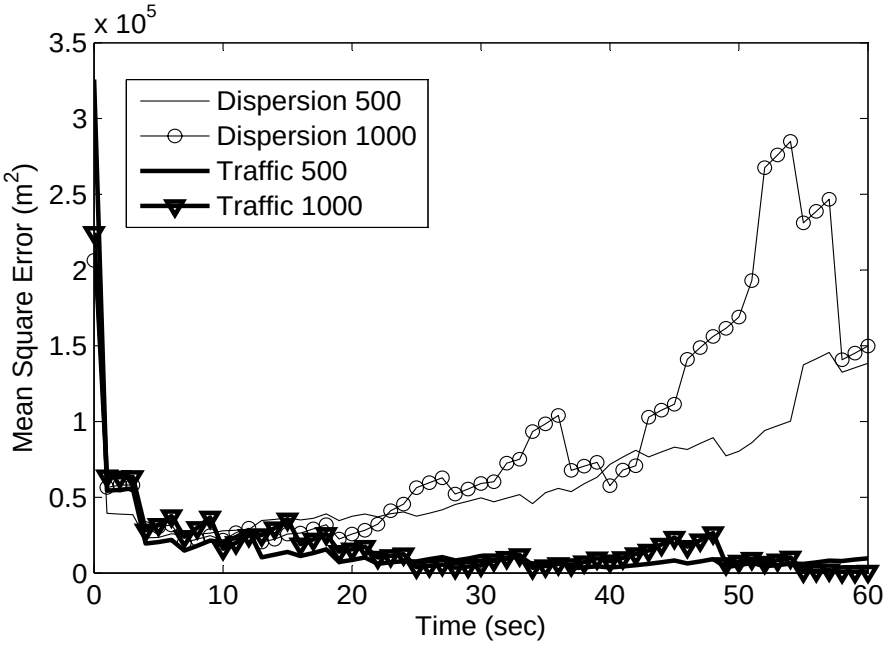


Figure 3.28: Average Mean Square Error After Target Found, Straight Path, 90% Confidence

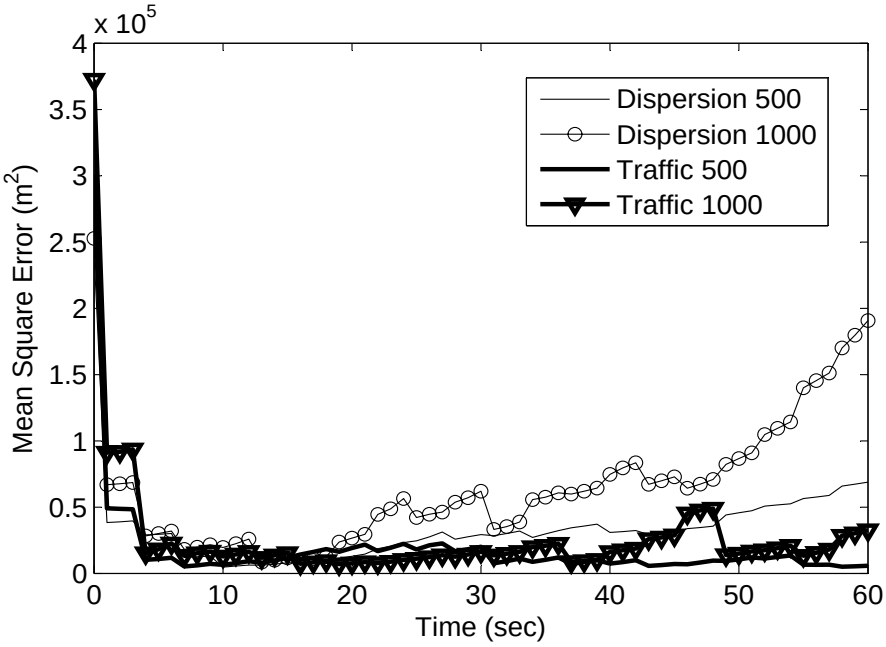
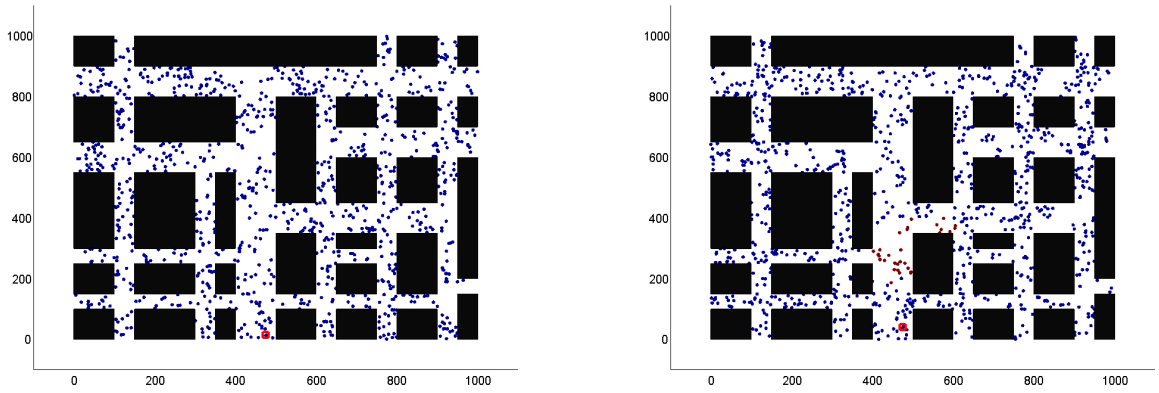


Figure 3.29: Average Mean Square Error After Target Found, Winding Path, 90% Confidence

The tracking performance results obtained after including a false report rate are illustrated by Figs. 3.28 and 3.29. In both cases, the Traffic Motion Model out performed the Dispersion Model. As expected, the Dispersion Model's performance in the case of a straight path was poorer than its performance in the case of a winding path, as the heading variable in the Dispersion Model is altered at each time step. The Traffic Motion Model's tracking performance improved with the inclusion of a false report belief, and consequently eliminated the need for particle redistribution, significantly reduced the time to detect in the case of winding target, and showed the importance of spatial resolution in the particle filter framework.

3.6.3 Reduced Number of Effective Particles

The resampling step is one of the more computationally expensive parts of a particle filter routine, as it is not parallelizable. Therefore, the frequency at which a probability density function is resampled is a design point. When the number of effective particles,



(a) Measurement 1: target found, resample

(b) Measurement 2: target found, no resample

Figure 3.30: Effect of the number of effective particles on particle spatial resolution, $N_{\text{eff}} > 90\%$N

N_{eff}, falls below a desired threshold,N_{thr}, the particle cloud is resampled. Equation 2.31 is repeated below for convenience.

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \tag{3.41}$$

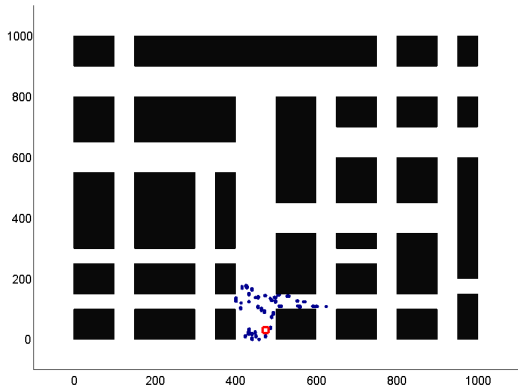
If the required number of effective particles is set less than the total number of particles (N), the distribution may not be resampled after each measurement. This could provide for reduced detection time by preserving some particles This could be important in the presence of a sensor with a false report rate. By not resampling after each measurement, the spatial resolution of the particle cloud is not immediately lost after a measurement, reducing the effect of a false report. For example, consider the initial particle distribution is illustrated by Fig. 3.30 (a) where the required number of effective particles is set to 90%N and a non-detection is reported in the region where $400 \leq x \leq 600$ and $200 \leq y \leq 400$. Figure 3.30 (b) illustrates that the particles in the region of non-detection are not automatically eliminated because the distribution was not resampled.

An investigation was done into the effect of reducing the required number of effective particles on the spatial resolution of the particle distribution. Two thresholds for the required number of effective particles were used: 90%N and 80% N. The sensor always reported the

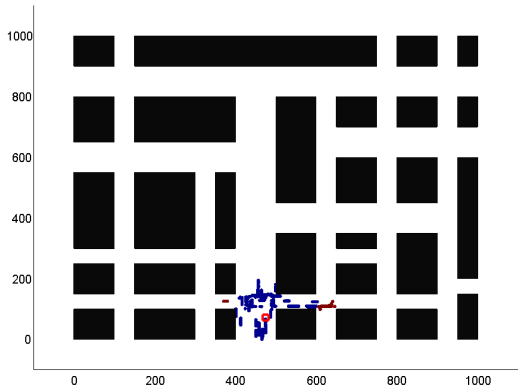
truth and the false report rate belief was set to 0%. Six measurements are taken, one every three seconds, and the particle distribution is shown after each measurement. The target is found in the first measurement, and the initial contraction and then gradual expansion of the particle cloud over time is shown. Particles of significant weight are blue and particles of lower weight are red. The rectangular region where $400 \leq x \leq 600$ and $0 \leq y \leq 200$ is measured at each measurement.

Both thresholds cause the particle distribution to contract to within the measured region after the first measurement. In addition, both thresholds cause the distribution to not be resampled after the second measurement. After the fourth measurement, the distribution is not resampled in the case of the 80% threshold, while it is resampled under a 90%N threshold. In both cases, when the target leaves the measured area and is not found, particles are not preserved in the region of non detection. Because the target is near the edge of the measured region but not detected after the fifth measurement, it would be beneficial to preserve some particles in the region of non-detection, as this could improve tracking performance.

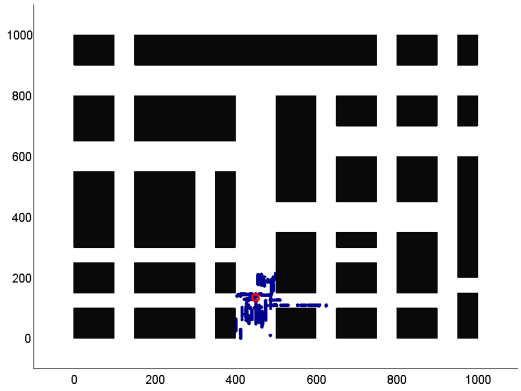
In addition to spatial resolution, tracking performance and time to detect data was collected. The sensor always reported the truth, and the particle distribution was only resampled when the number of effective particles fell below the specified threshold. Thresholds for the number of effective particles of 80%N and 90%N were used, and the performance was measured in the case of both a winding and a straight target path.



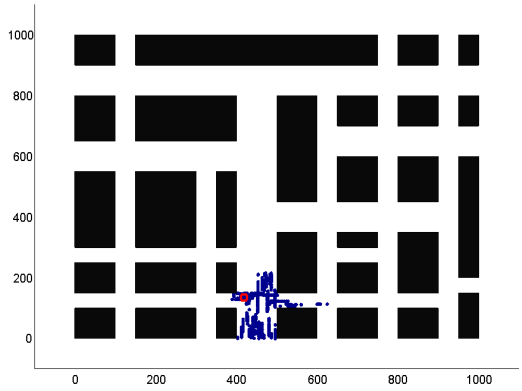
(a) Measurement 1: target found, resample



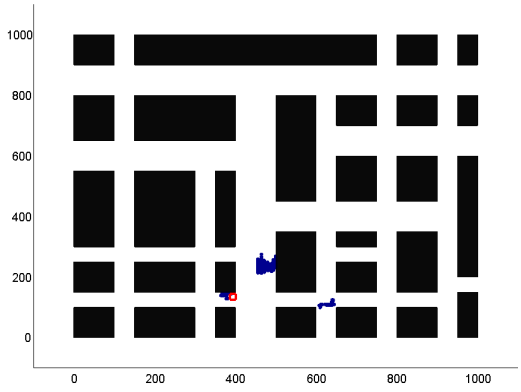
(b) Measurement 2: target found, no resample



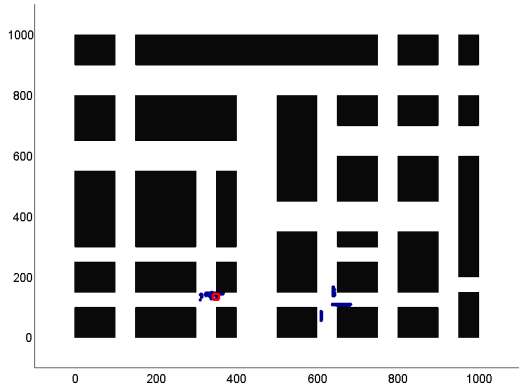
(c) Measurement 3: target found, resample



(d) Measurement 4: target found, resample

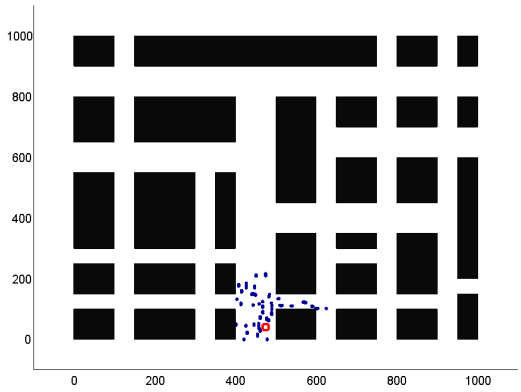


(e) Measurement 5: target not found, resample

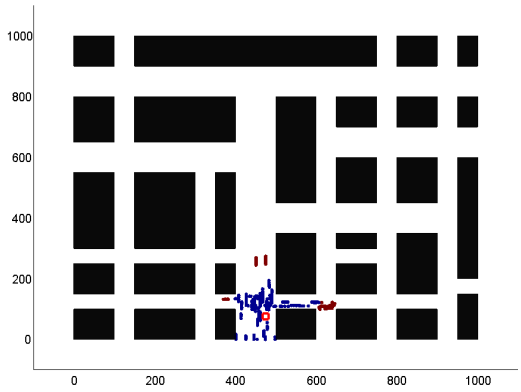


(f) Measurement 6: target not found, resample

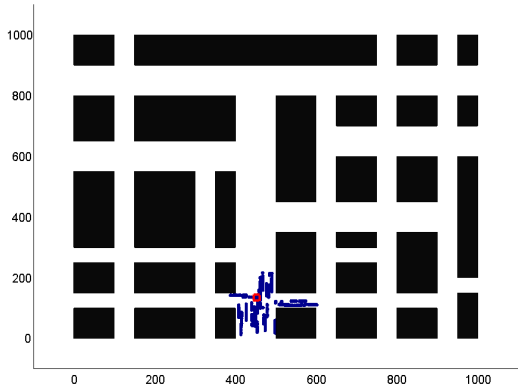
Figure 3.31: Effect of the number of effective particles on particle spatial resolution, $N_{eff} > 90\%N$



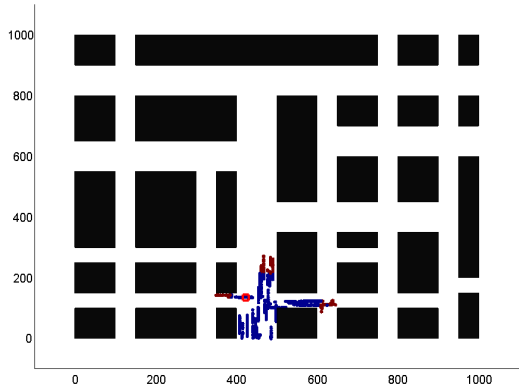
(a) Measurement 1: target found, resample



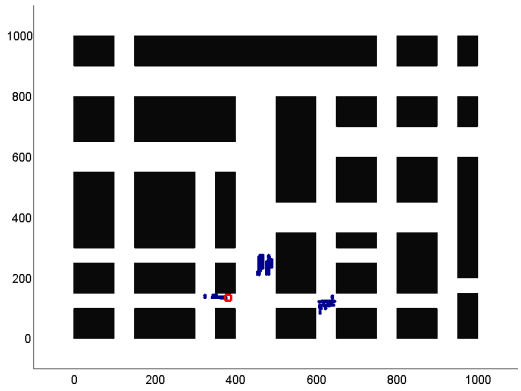
(b) Measurement 2: target found, no resample



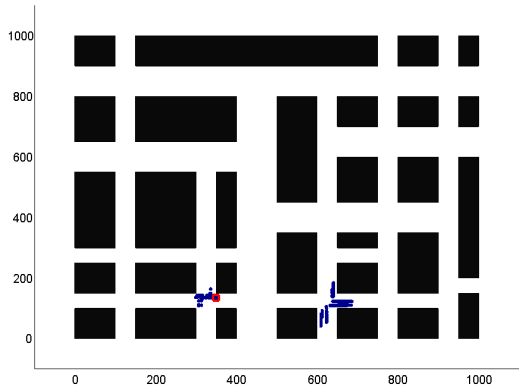
(c) Measurement 3: target found, resample



(d) Measurement 4: target found, no resample



(e) Measurement 1: target not found, resample



(f) Measurement 6: target not found, resample

Figure 3.32: Effect of the number of effective particles on particle spatial resolution, $N_{eff} > 80\%N$

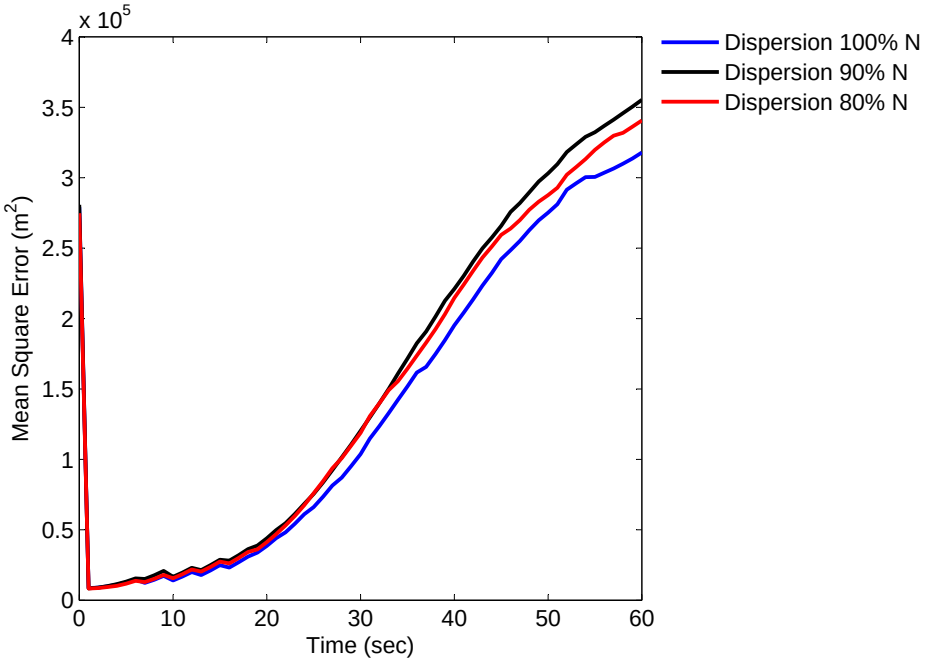


Figure 3.33: Dispersion Model, Average Mean Square Error After Target Found, Straight Path, 80%N and 90%N Effective Particle Thresholds

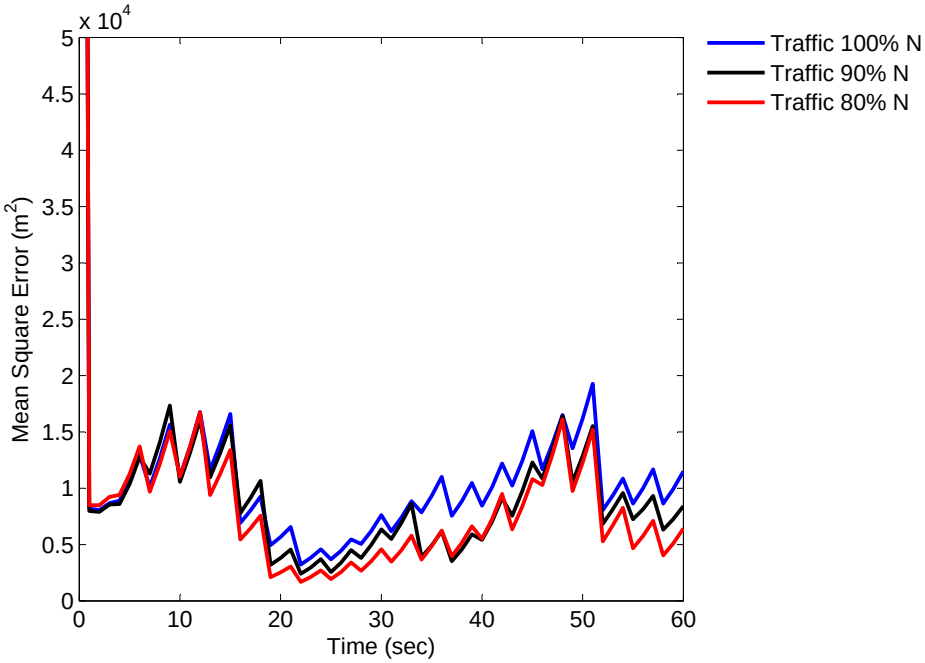


Figure 3.34: Traffic Motion Model, Average Mean Square Error After Target Found, Straight Path, 80%N and 90%N Effective Particle Thresholds

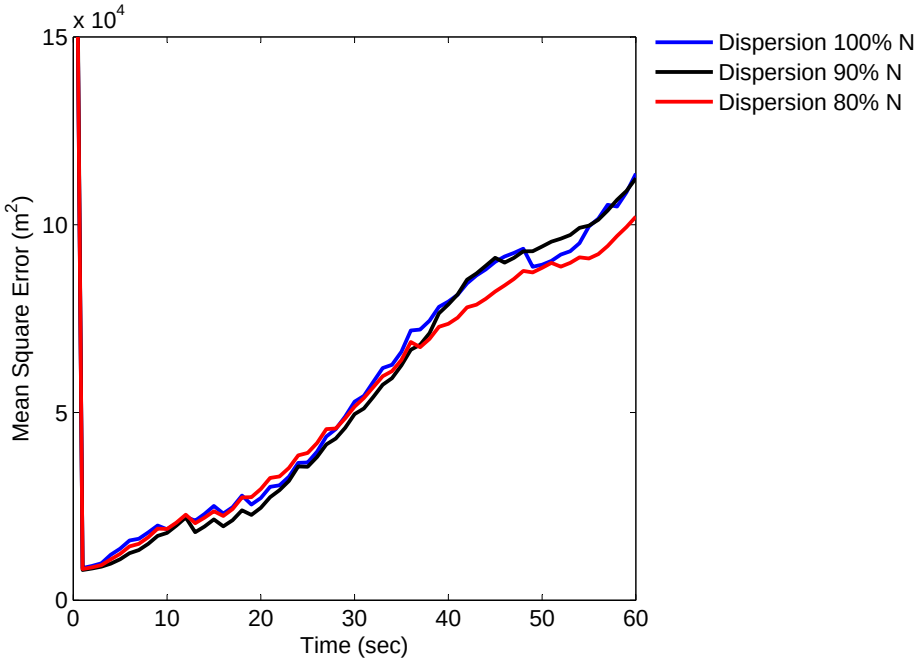


Figure 3.35: Dispersion Model, Average Mean Square Error After Target Found, Winding Path, 80%N and 90%N Effective Particle Thresholds

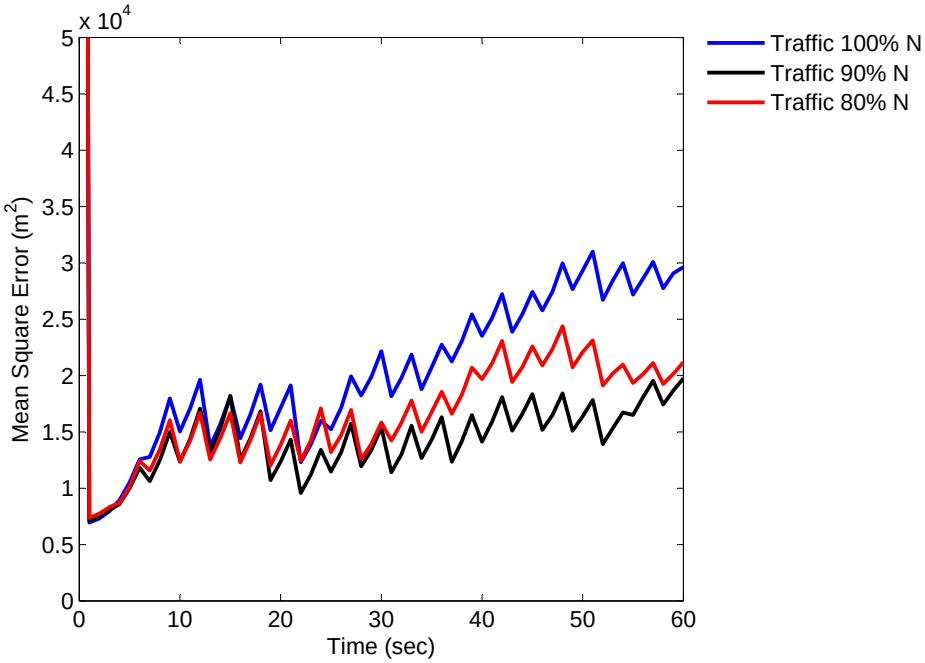


Figure 3.36: Traffic Motion Model, Average Mean Square Error After Target Found, Winding Path, 80%N and 90%N Effective Particle Thresholds

Table 3.6: Time to Detect Results for Straight Path, N = 1000

Particle Model	**N_{eff} Threshold**	**Time to Detect**
Dispersion	100% N	53.59 sec
Traffic	100% N	29.31 sec
Dispersion	90% N	56.61 sec
Traffic	90% N	35.38 sec
Dispersion	80% N	60.93 sec
Traffic	80% N	29.89 sec

Table 3.7: Time to Detect Results for Winding Path, N = 1000

Particle Model	**N_{eff} Threshold**	**Time to Detect**
Dispersion	100% N	66.13 sec
Traffic	100% N	69.54 sec
Dispersion	90% N	73.02 sec
Traffic	90% N	44.83 sec
Dispersion	80% N	65.64 sec
Traffic	80% N	50.95 sec

3.7 Summary of Results

A high fidelity traffic model was developed to propagate a particle filter in time in an urban environment to track a ground target vehicle. The sensor model used to locate and track the target vehicle was non-differentiable and provided only a binary response of target presence within a large region. In all cases, it is observed that the Traffic Motion Model provides for superior target tracking, and computational time may be appropriately decreased without significant loss of fidelity by adjusting the number of particles used, keeping spatial resolution in mind.

These results are encouraging, given the ease of introducing a false report belief into the particle filter framework and when paired with the previously obtained tracking performance

results. By decreasing the particle filter's confidence in the sensor, a softer *pdf* is obtained. This forces a broader spatial resolution of particle cloud and decreases the probability of losing the target after it is detected. It is nonintuitive to detune a filter to improve performance, but because the sensor model in this problem requires an added focus on spatial resolution, it is necessary.

This result also brings to light the coupling between the performance due to the dynamics model, sensor model, and particle management. Chapter 4 will explore the effect of an imperfect sensor on the performance of each model. The false report rate belief may be tuned to maintain the spatial resolution of the particle distribution after a false detection is reported. In addition, decreasing the required number of effective particles could provide for sufficient spatial resolution in the presence of a missed-detection. Chapter 4 discusses this in a structured and systematic approach to tuning the spatial resolution to achieve improved performance in the presence of various sensor models. The importance of the dynamic model has been solidified in the presence of a perfect sensor.

Chapter 4

Accounting for an Imperfect Sensor

Following the validation of the simulation's stability and the effectiveness of the Traffic Motion Model, a more realistic sensor model was introduced. Because the regional sensor is modeled after a human operator, the possibility of a false report must be addressed. The idea of adjusting the particle filter's belief in the sensor's false report rate was introduced in Section 3.6.2. By increasing the particle filter's belief in the false report rate without actually simulating false measurements, the spatial resolution of the particle cloud was preserved such that the target was not always lost when it was just outside of a measured region. This decreased the time to detect the target. The benefit provided by adjusting the believed false report rate provided insight into how to improve performance in the presence of actual false reports.

The previous chapter indicates the vital role of the motion model in tracking performance and the flexibility of the particle filter framework when sensor data is true. The results given in this chapter indicate the importance of the measurement update in the presence of untrustworthy measurements. A thorough investigation into the tuning of the existing parameters in the measurement update was done in order to establish the effect of each parameter. The detrimental impact of accepting a false report to be true is presented, in addition to the idea of establishing a risk metric that could be used to determine if a measurement is likely false.

The sensor model was modified to provide false reports at some measurements. For example, if a sensor is said to have a 10% false report rate (R_{false}), each measurement has a 10% chance of reporting the opposite of the truth. This results in identical values for missed detection and false detection rates. False measurements are achieved through the

following steps: measure a region, determine the truth, take a random draw from a uniform distribution between 0 and 1, if the random value is greater than 1 - R_{false}, the opposite of the truth is reported.

Figure 4.1 illustrates the diminished tracking performance that results from the introduction of a sensor with a 10% false report rate. It should be noted that in each case, the particle filter believed the sensor had a false report rate of 10%. The particle filter's belief in the false report rate affects how much a particle's weight is changed based on a measurement. It does not affect if the particle filter accepts or rejects a measurement. Each measurement is accepted to be true. As mentioned in Section 3.5, the tracking performance is measured for 60 seconds following the first time a detection is reported. By introducing a 10% false report rate into the sensor model, the tracking performance could begin being measured before the target is actually found. In the 10% false report example in Fig. 4.1, 60% of the initial "detection" reports were false reports. This resulted in seemingly diminished tracking performance when compared to a perfect sensor.

The results in Table 4.1 indicate that the average values of detection time decreased, as would be expected with the possibility of a reported detection actually being a false positive. For example, if measurements are taken every 3 seconds by a sensor with a false positive rate of 10%, one would expect a maximum time to detect around 30 seconds. This measure is less than the time to detect for the Dispersion and Traffic Motion Models presented in Section 3.5, which indicates a high probability of a false positive occurring before the target is actually found.

Trusting a false positive is a significant cause of decreased tracking performance. Sections 4.1 and 4.2 examine the effect of tuning parameters that exist within the standard particle filter framework in an effort to improve tracking performance in the presence of false reports. Section 4.1 investigates the measurement update step and the effect of the particle filter's belief of the sensor's false report rate. Section 4.2 studies the effect of adjusting the resampling step by tuning the effective number of particles. Both parameters play into the

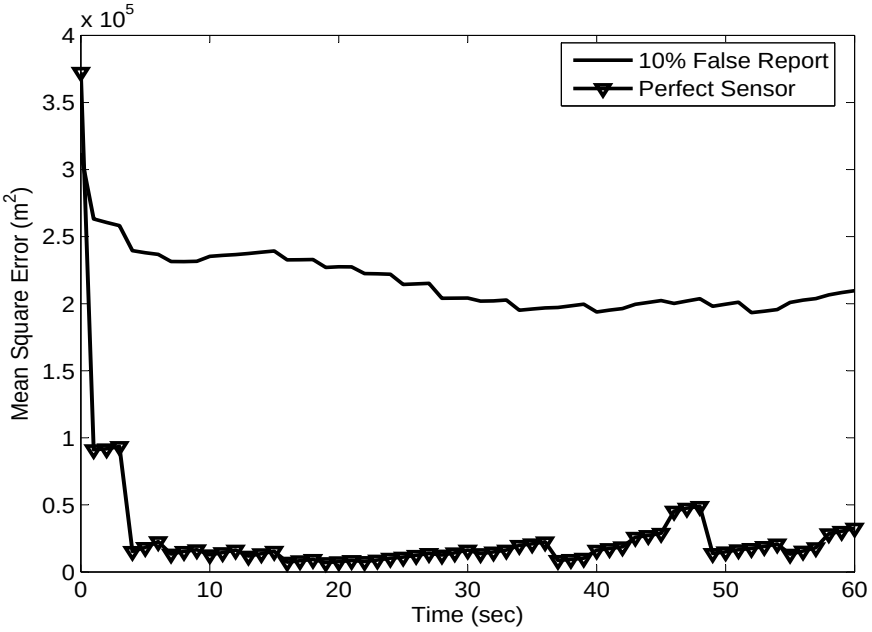


Figure 4.1: Impact of a Sensor with 10% False Report Rate, Traffic Model, Winding Path, N = 1000, False Report Belief 10%

importance of maintaining proper spatial resolution of the particle cloud based on a sensor's perceived accuracy. After the most beneficial values for the sensor belief and number of effective particles is determined for each sensor model, a risk assessment is introduced. The practice of assessing the cost of resampling after a measurement and accepting a positive measurement through a Bayesian risk analysis is discussed in Section 4.3.

4.1 Sensor Belief Study

As previously discussed in Section 3.6.2, the inclusion of a believed false report rate into the particle filter's measurement update caused particle weights in regions of non-detection to be significantly reduced, but not always eliminated in the resampling process. Conversely, some particles outside of a region of detection were no longer immediately eliminated until after a repeated detection. Recall, the particle update process directly involves the particle

Table 4.1: Time to Detect Results for Winding Path, N = 1000, False Report Belief 10%

Particle Model	**Target Path**	**Sensor Model**	**Time to Detect**
Dispersion	Straight	Perfect	57.9 sec
Dispersion	Winding	Perfect	67.7 sec
Traffic	Straight	Perfect	28.8 sec
Traffic	Winding	Perfect	32.2 sec
Dispersion	Straight	10% False Report	17.43 sec
Dispersion	Winding	10% False Report	20.9 sec
Traffic	Straight	10% False Report	16.5 sec
Traffic	Winding	10% False Report	22.35 sec

filter's belief in the sensor's false report rate, R_{false}.

$$p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) = 1 - R_{false} \tag{4.1}$$

$$p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right) = \sum_{i=1}^{N_{in}} w_{in}^i \tag{4.2}$$

$$p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right) = (1 - R_{false}) \sum_{i=1}^{N_{in}} w_{in}^i + (R_{false})(1 - \sum_{i=1}^{N_{in}} w_{in}^i) \tag{4.3}$$

$$p\left(\mathbf{x}_k \mid \mathbf{z}_k\right) = \frac{p\left(\mathbf{z}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{Z}_{k-1}\right)}{p\left(\mathbf{z}_k \mid \mathbf{Z}_{k-1}\right)} \tag{4.4}$$

$$w_k^i = \frac{w_k^i p\left(\mathbf{x}_k \mid \mathbf{z}_k\right)}{\sum w_{in}} \tag{4.5}$$

An investigation into the effects of the sensor accuracy on the amount of time required to detect the target and the tracking performance was included in this study. Three sensor models were used for comparison: a perfect sensor, a sensor with 10% false report rate, and a sensor with 20% false report rate. Figures are shown for the converged average error over 200 Monte Carlo runs. The target traversed both a straight and a winding path, and was initialized the same way at the start of each run. Particles were not redistributed in this study.

The following figures illustrate that regardless of the sensor model, the Traffic Motion Model tracks the target better than the Dispersion Motion Model. Some additional consideration will be given to the "optimal" value for the particle filter's belief of the false alarm rate as the target's path is never truly known.

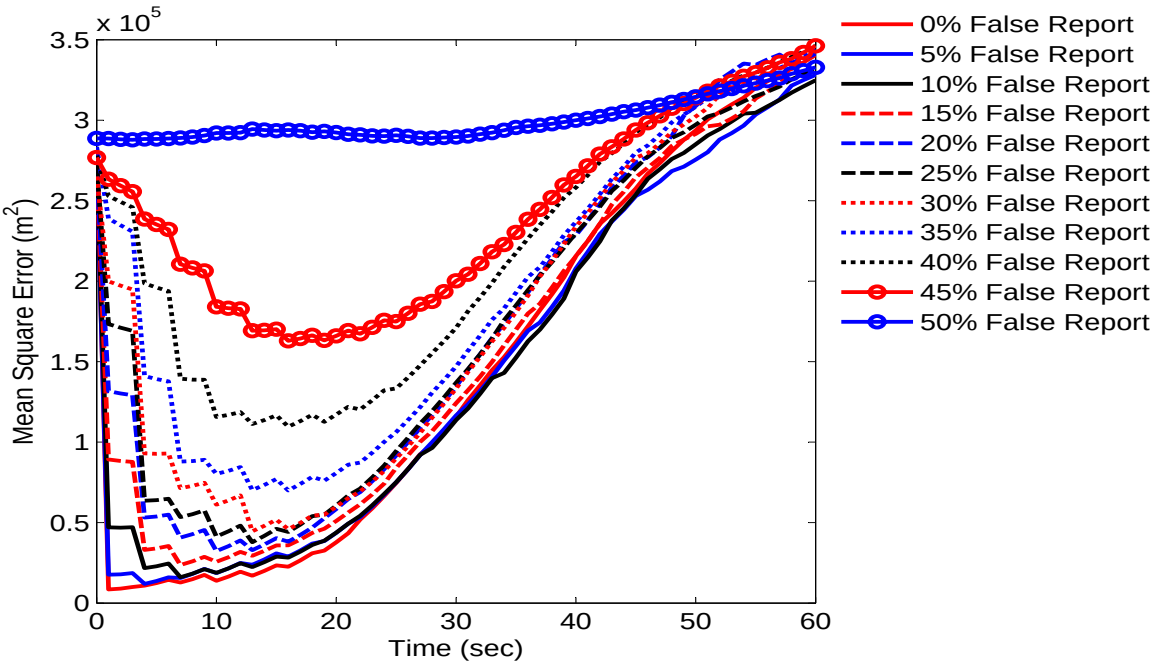


Figure 4.2: Dispersion Model, Straight Path, False Report Belief Study, Perfect Sensor

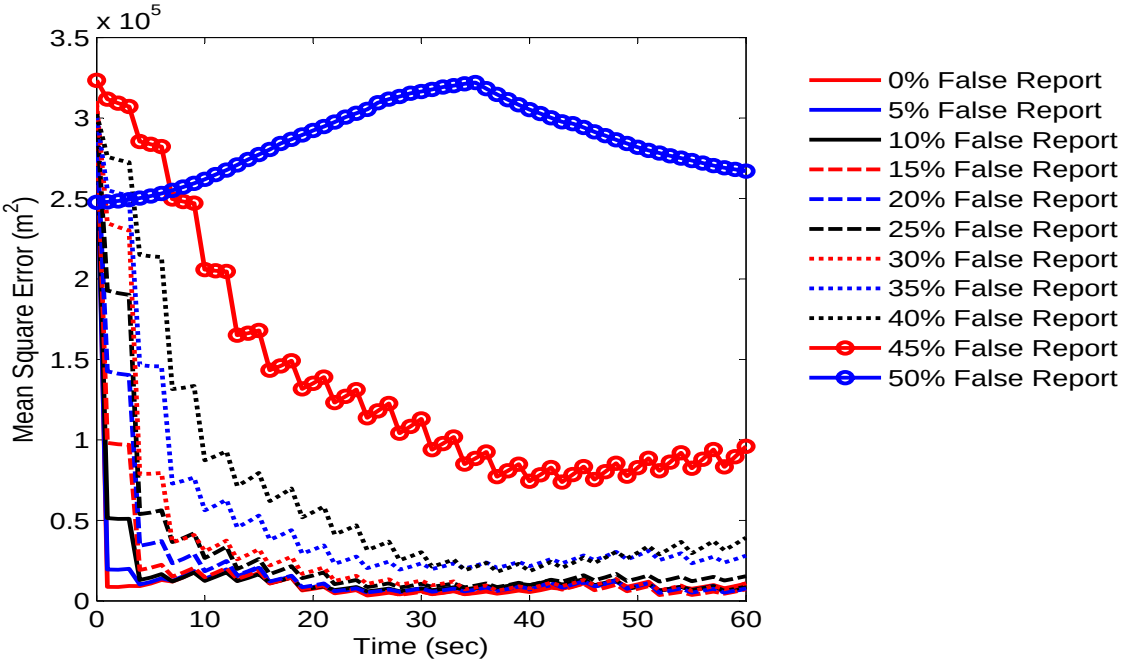


Figure 4.3: Traffic Model, Straight Path, False Report Belief Study, Perfect Sensor

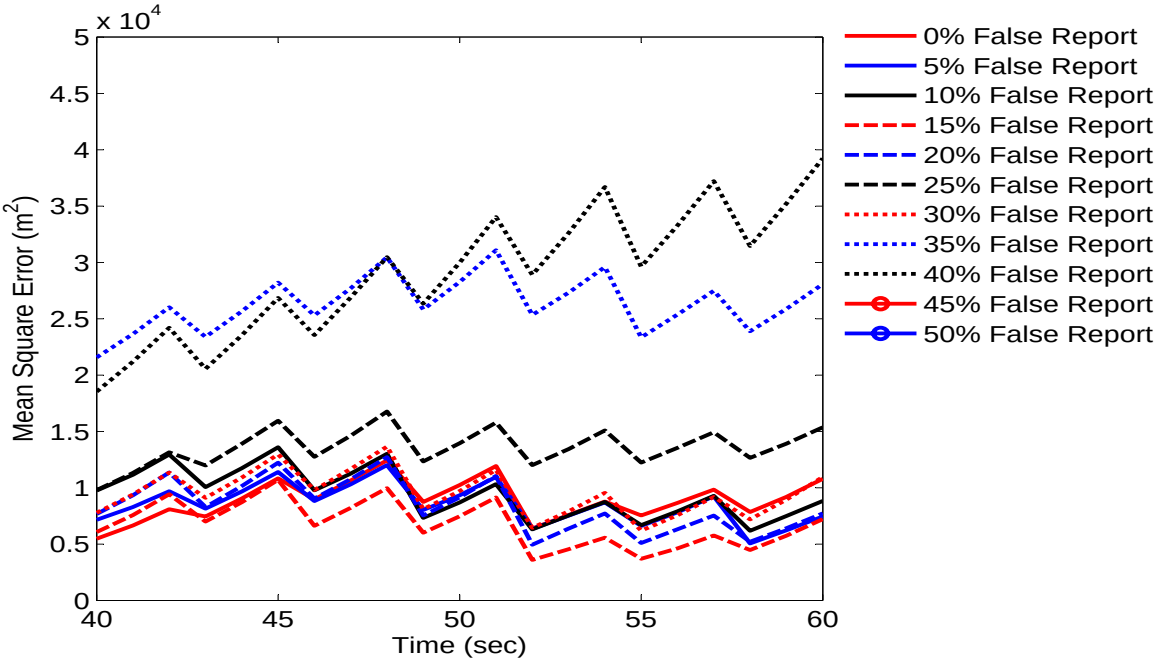


Figure 4.4: Traffic Model, Straight Path, False Report Belief Study, Perfect Sensor, final 20 seconds

The figures above indicate that in the presence of a perfect sensor, improved tracking performance is obtained when the particle filter assumes a false report rate exists, regardless of motion model. The Dispersion Model yields slightly improved performance with a false report rate belief of 5% and 10%. As expected, the Traffic Motion Model provides improved tracking performance when compared to the Dispersion Model, and with the inclusion of a false report rate belief of 15%. All of the results follow the same trend, except for the 50% false report belief cases. In those cases, no benefit is given to region of detection, therefore additional measurements do not improve tracking performance.

Table 4.2: Time to Detect Results, Straight Path, Perfect Sensor, False Report Belief 0%-50%

False Report Belief	**Time to Detect Dispersion**	**Time to Detect Traffic**
0%	57.93 sec	34.41 sec
5%	51.42 sec	30.91 sec
10%	49.23 sec	32.64 sec
15%	53.41 sec	36.19 sec
20%	51.91 sec	31.41 sec
25%	44.86 sec	33.21 sec
30%	50.86 sec	33.55 sec
35%	51.33 sec	33.45 sec
40%	47.53 sec	33.65 sec
45%	52.87 sec	34.66 sec
50%	284.58 sec	208.36 sec

The results in Table 4.2 indicate that some benefit is obtained by including a false report belief in the measurement updated in the case of both motion models. The improved tracking capabilities yielded by the Traffic Motion Model over the Dispersion Model consistently aids in reducing the time to detect the target. In addition, the 50% false report belief results indicate a significantly increased time to detect, as the particle weights are not altered significantly based on any measurement.

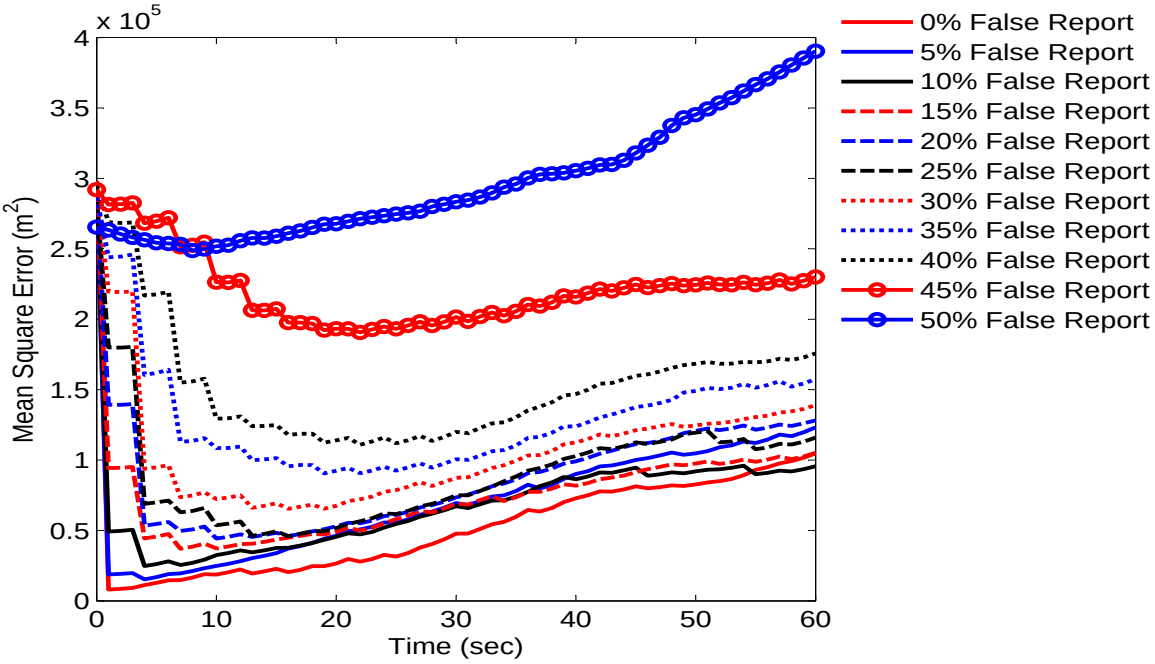


Figure 4.5: Dispersion Model, Winding Path, False Report Belief Study, Perfect Sensor

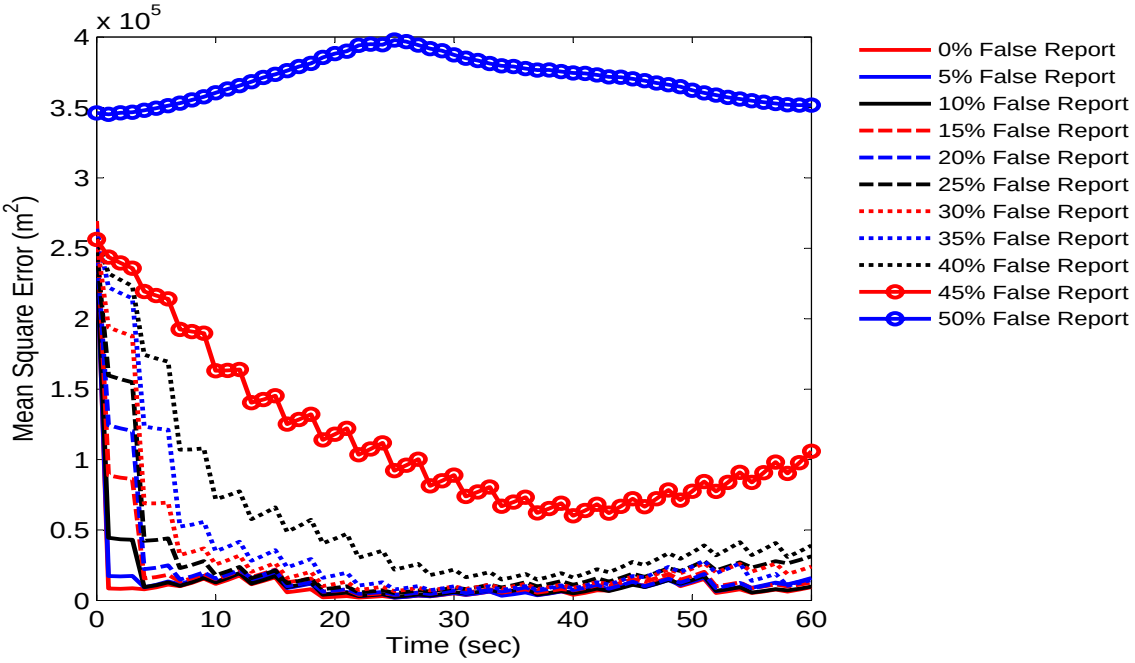


Figure 4.6: Traffic Model, Winding Path, False Report Belief Study, Perfect Sensor

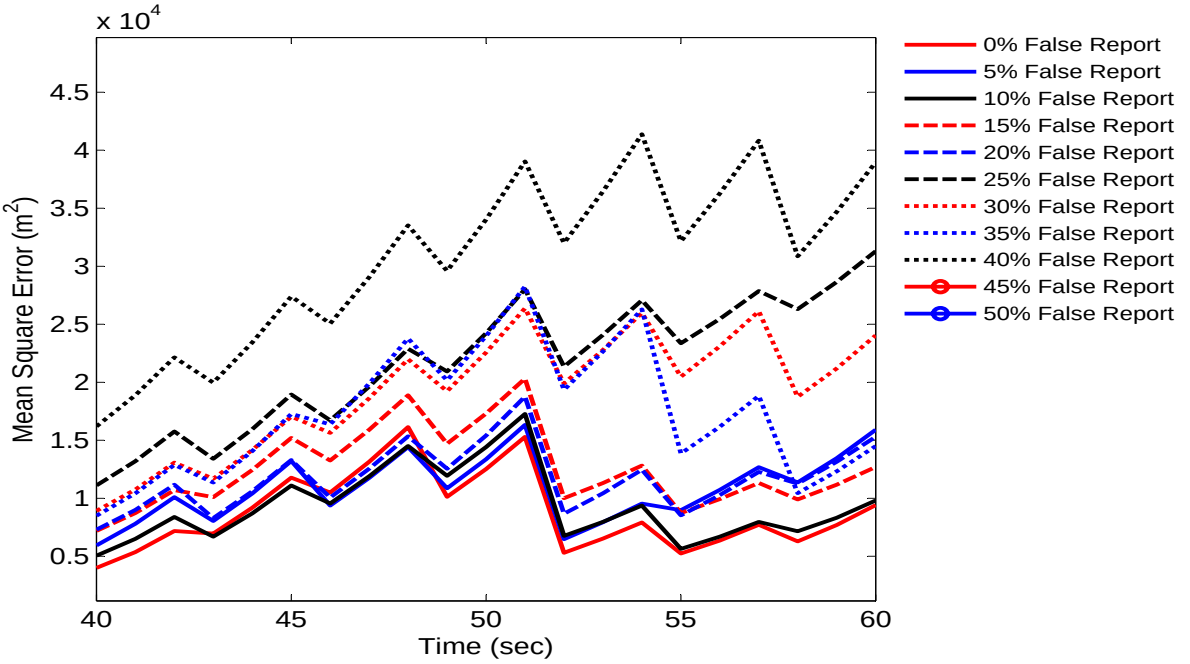


Figure 4.7: Traffic Model, Winding Path, False Report Belief Study, Perfect Sensor, Final 20 seconds

In agreement with the results in the case of a straight path, the figures above indicate that the Traffic Model outperforms the Dispersion Model's tracking performance. In addition, tuning the sensor's belief to 0% false report rate to match the sensor's actual false report rate provides improved tracking performance for both motion models. Tuning the false report rate belief to 10% also provides improved tracking in the case of the Traffic Motion Model, as consistent with the results in Chapter 3. The time to detect results are consistent with that of a straight path, in that the Dispersion Model provides for faster localization. However, both models benefit from the inclusion of a false report rate in the measurement update. This occurs because the winding path traverses through the same quadrant of the map for a significant amount of time. Therefore, if the target is located just outside of a measured region, it would benefit from some of the particles in that measured region being preserved. This would provide for improved particle spatial resolution in the proximity of the target and therefore decrease the time to detect the target.

Table 4.3: Time to Detect Results, Winding Path, Perfect Sensor, False Report Belief 0%-50%

False Report Belief	**Time to Detect Dispersion**	**Time to Detect Traffic**
0%	72.13 sec	38.85 sec
5%	73.43 sec	42.53 sec
10%	70.50 sec	39.61 sec
15%	63.06 sec	38.03 sec
20%	67.44 sec	40.96 sec
25%	62.20 sec	39.03 sec
30%	73.71 sec	38.40 sec
35%	62.28 sec	37.43 sec
40%	66.85 sec	40.96 sec
45%	74.14 sec	42.53 sec
50%	349.67 sec	265.15 sec

The following figures and tables illustrate the effect of including a sensor with a false report rate of 10% and 20%, respectively. Tracking performance is greatly diminished, and the mean square error is greater than the area of a regional sensor in all cases. False start rates are included for comparison. The false-start rate is defined as the percentage of times the first reported detection of the target is a false report, thus causing the tracking performance to begin being measured when the target was not actually found.

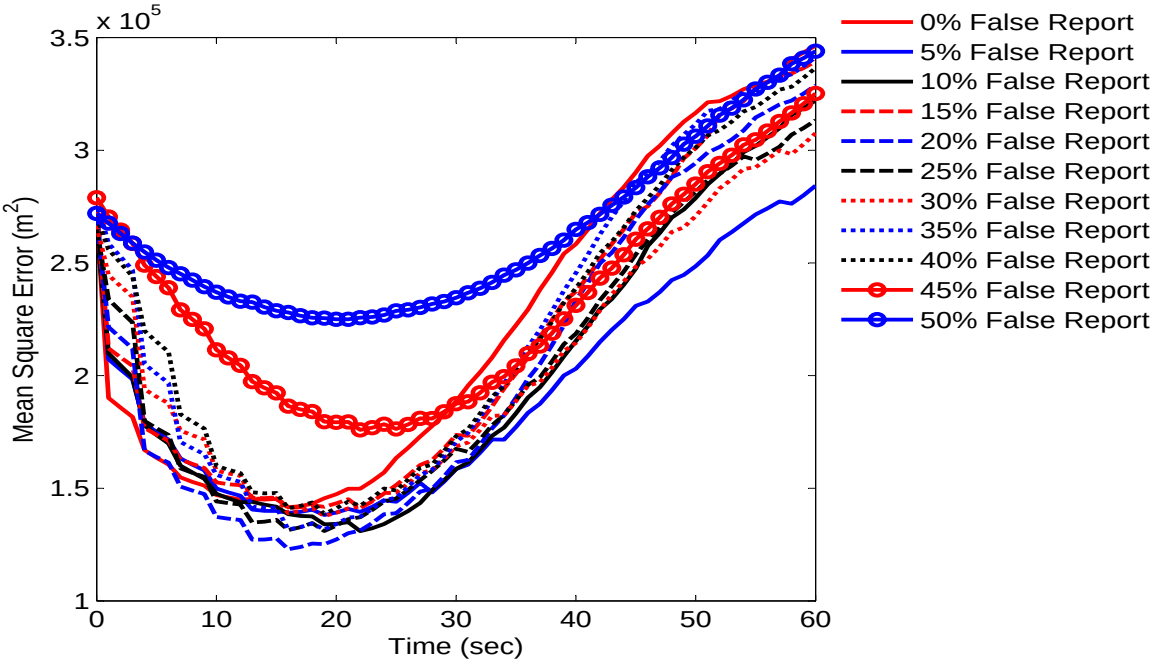


Figure 4.8: Dispersion Model, Straight Path, False Report Belief Study, 10% False Report Sensor

Figure 4.8 illustrates that in the case of a target on a straight path and a 10% false report rate sensor, the inclusion of a 5% false report belief results in improved performance for the Dispersion Model. However, the best performers converge to an error that is close to an order of magnitude larger than that of the perfect sensor, and well outside the bounds of a sensor region.

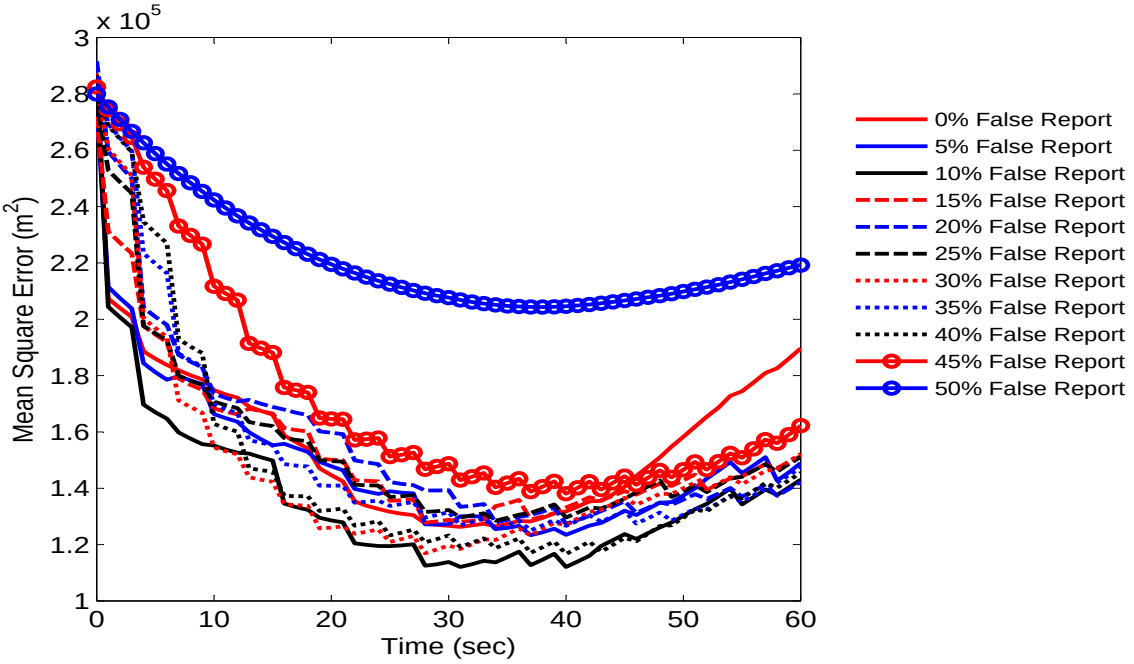


Figure 4.9: Traffic Model, Straight Path, False Report Belief Study, 10% False Report Sensor

Tuning the false report belief to that of the sensor's actual false report rate provided for optimal tracking performance when compared to the other false report belief values for the Traffic Motion Model. Although the performance is much worse than that with results from having perfect sensor data, the Traffic Motion Model provides for better target tracking than the Dispersion Model, despite a consistently higher false start rate, as given by Table 4.4. The higher false start rate was expected for the Traffic Motion Model in the presence of a sensor with a false report rate, as its time to detect with a perfect sensor was higher than that of the Dispersion Model for a perfect sensor. Therefore, because it would take longer to localize the target given a perfect sensor, the probability of receiving a false report before the target is actually found is higher than that of the Dispersion Model.

Table 4.4: Time to Detect Results, Straight Path, 10% False Report Sensor, False Report Belief 0%-50%

False Report Belief	Time to Detect Dispersion	% False Starts Dispersion	Time to Detect Traffic	%False Starts Traffic
0%	18.58 sec	61.50	19.84 sec	65.00
5%	18.57 sec	55.00	20.26 sec	64.50
10%	20.04 sec	55.00	19.56 sec	57.50
15%	18.78 sec	63.50	21.25 sec	63.50
20%	20.01 sec	56.00	17.56 sec	67.50
25%	18.61 sec	62.00	21.21 sec	62.50
30%	20.07 sec	60.00	19.08 sec	62.50
35%	19.05 sec	58.50	19.77 sec	68.50
40%	17.51 sec	69.50	19.47 sec	58.50
45%	20.35 sec	65.00	21.77 sec	64.50
50%	28.02 sec	85.00	27.45 sec	90.00

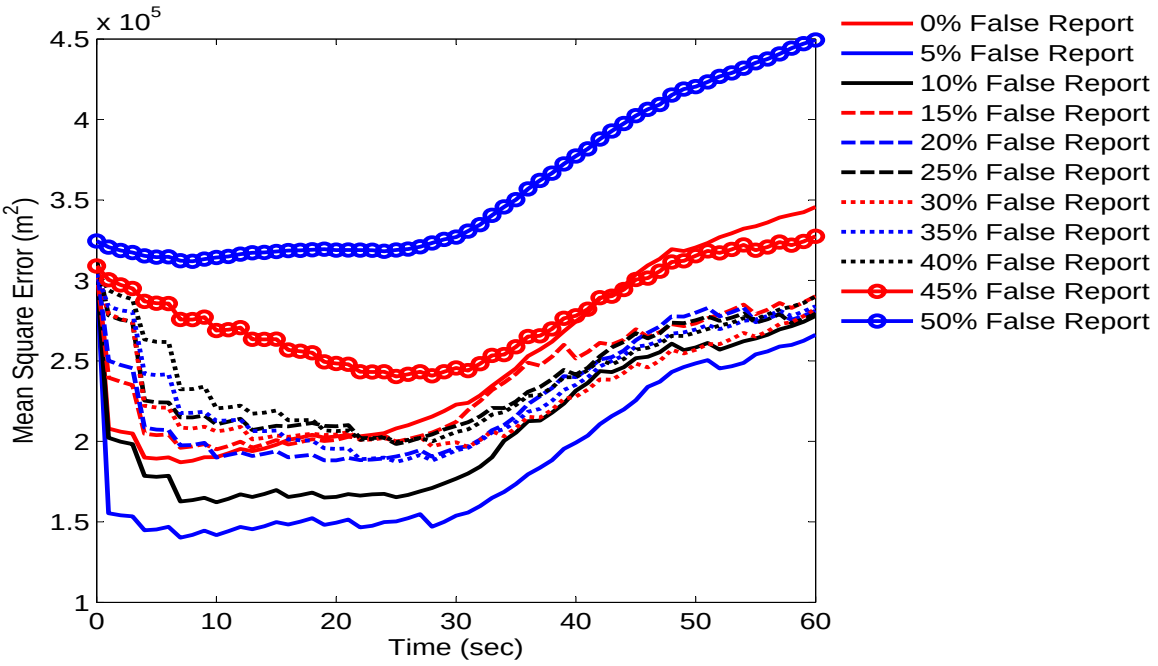


Figure 4.10: Dispersion Model, Winding Path, False Report Belief Study, 10% False Report Sensor

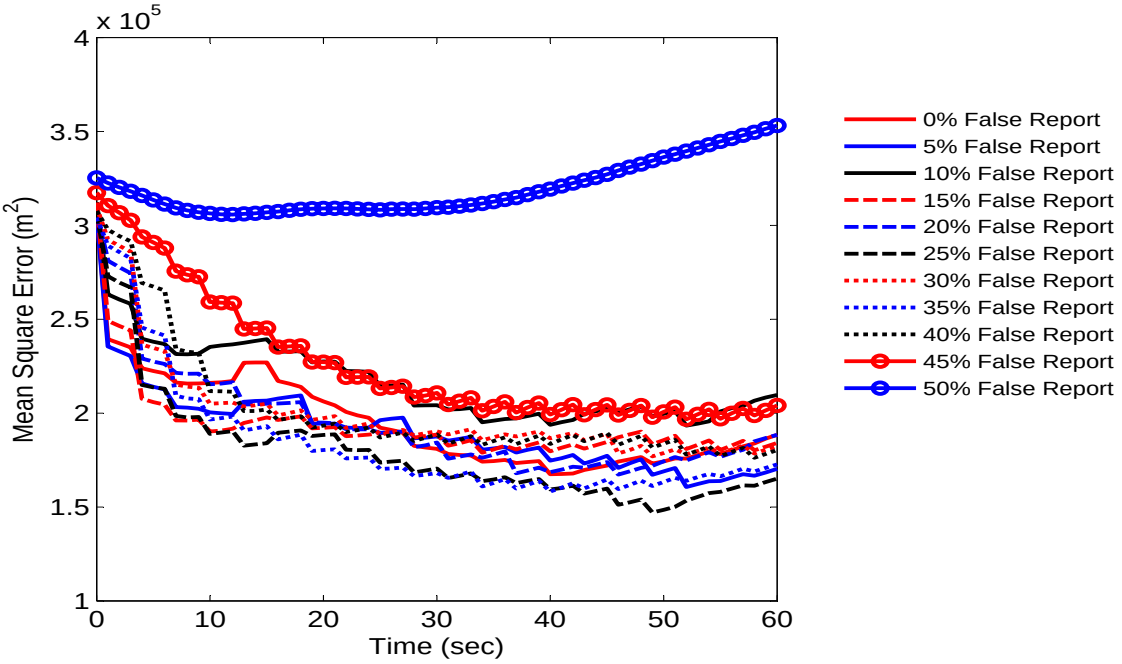


Figure 4.11: Traffic Model, Winding Path, False Report Belief Study, 10% False Report Sensor

The results for a winding path are similar to those of a straight path. The Traffic Motion Models shows improved performance when tuned to have a 25% false report belief, while the Dispersion Model benefits from a 5% false report belief. Again, the tracking performance is substantially worse than when the truth is always reported. Despite the higher false start rate, the Traffic Motion Model produces better tracking results than the Dispersion Model in the case of a winding path with a 10% false report rate sensor model.

Table 4.5: Time to Detect Results, Winding Path, 10% False Report Sensor, False Report Belief 0%-50%

False Report Belief	Time to Detect Dispersion	% False Starts Dispersion	Time to Detect Traffic	%False Starts Traffic
0%	20.78 sec	55.50	20.79 sec	62.50
5%	20.73 sec	57.00	21.37 sec	70.00
10%	17.76 sec	65.00	20.91 sec	73.50
15%	18.28 sec	61.00	19.80 sec	61.00
20%	20.91 sec	59.50	21.06 sec	71.50
25%	18.36 sec	65.00	20.61 sec	64.50
30%	19.96 sec	69.00	20.02 sec	64.50
35%	20.37 sec	65.00	22.00 sec	65.50
40%	20.52 sec	62.00	21.19 sec	65.00
45%	21.03 sec	58.00	18.49 sec	69.00
50%	29.05 sec	94.50	31.25 sec	92.50

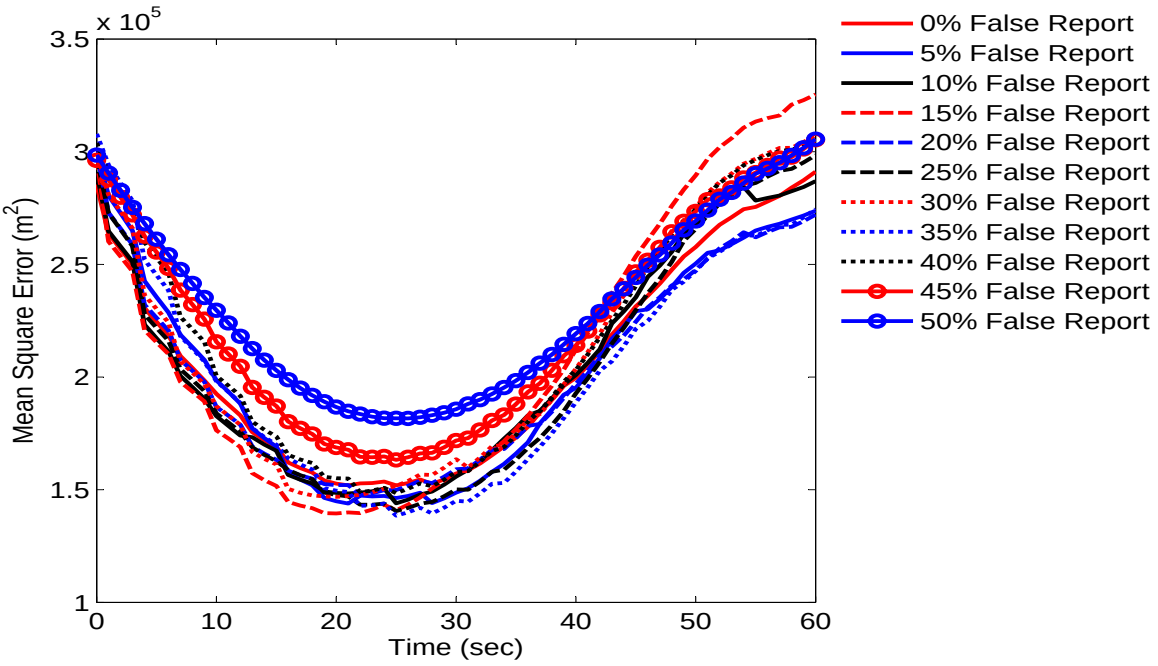


Figure 4.12: Dispersion Model, Straight Path, False Report Belief Study, 20% False Report Sensor

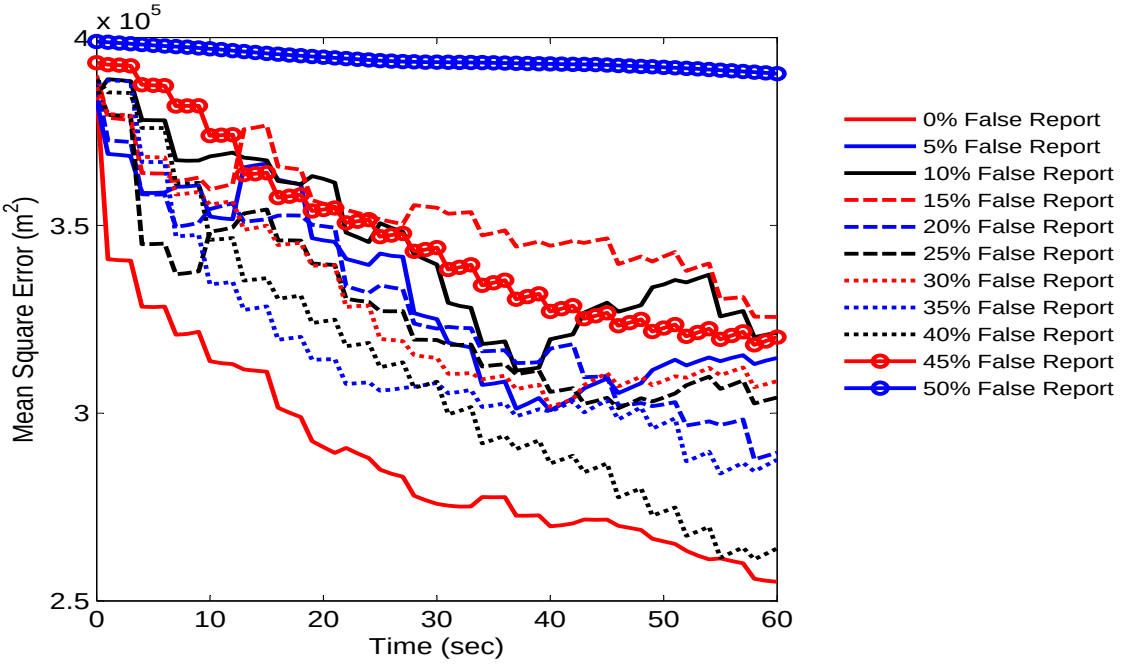


Figure 4.13: Traffic Model, Straight Path, False Report Belief Study, 20% False Report Sensor

The increase in the sensor's false report rate from 10% to 20% nearly doubled the tracking error for both motion models. The figures above indicate that the Dispersion Model achieves increased performance when tuned to believe the sensor's false report rate is 5% or 35%, but does not yield convergent tracking performance. Whereas, the Traffic Motion Model's performance is heavily diminished in the presence of an average false start of 83%, it is still able to produce tracking error that decreases over time, and the best performer is that of a false report belief of 0% and 40%.

Table 4.6: Time to Detect Results, Straight Path, 20% False Report Sensor, False Report Belief 0%-50%

False Report Belief	Time to Detect Dispersion	% False Starts Dispersion	Time to Detect Traffic	%False Starts Traffic
0%	13.06 sec	77.00	13.35 sec	82.00
5%	12.45 sec	80.00	14.79 sec	82.00
10%	12.36 sec	74.00	13.53 sec	84.00
15%	12.72 sec	77.50	12.15 sec	81.50
20%	13.26 sec	78.50	12.42 sec	83.50
25%	13.89 sec	76.00	14.01 sec	83.00
30%	11.24 sec	78.00	12.81 sec	87.50
35%	14.41 sec	76.00	13.72 sec	82.00
40%	11.53 sec	78.50	13.65 sec	84.50
45%	12.82 sec	76.50	13.68 sec	88.00
50%	13.72 sec	93.50	14.91 sec	90.50

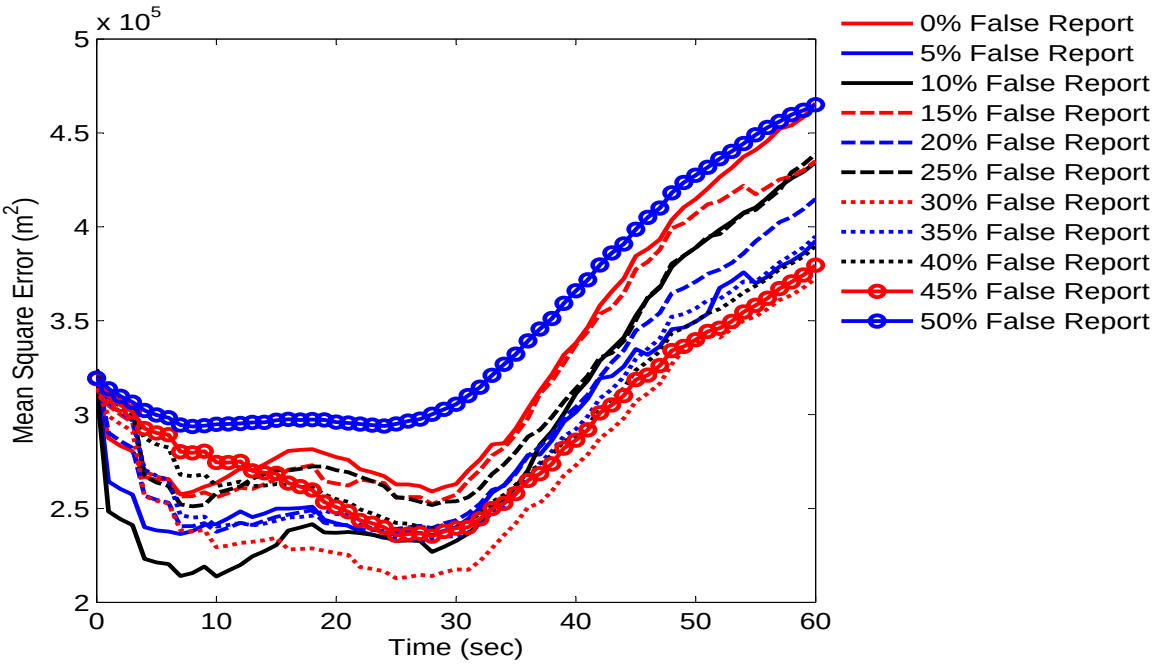


Figure 4.14: Dispersion Model, Winding Path, False Report Belief Study, 20% False Report Sensor

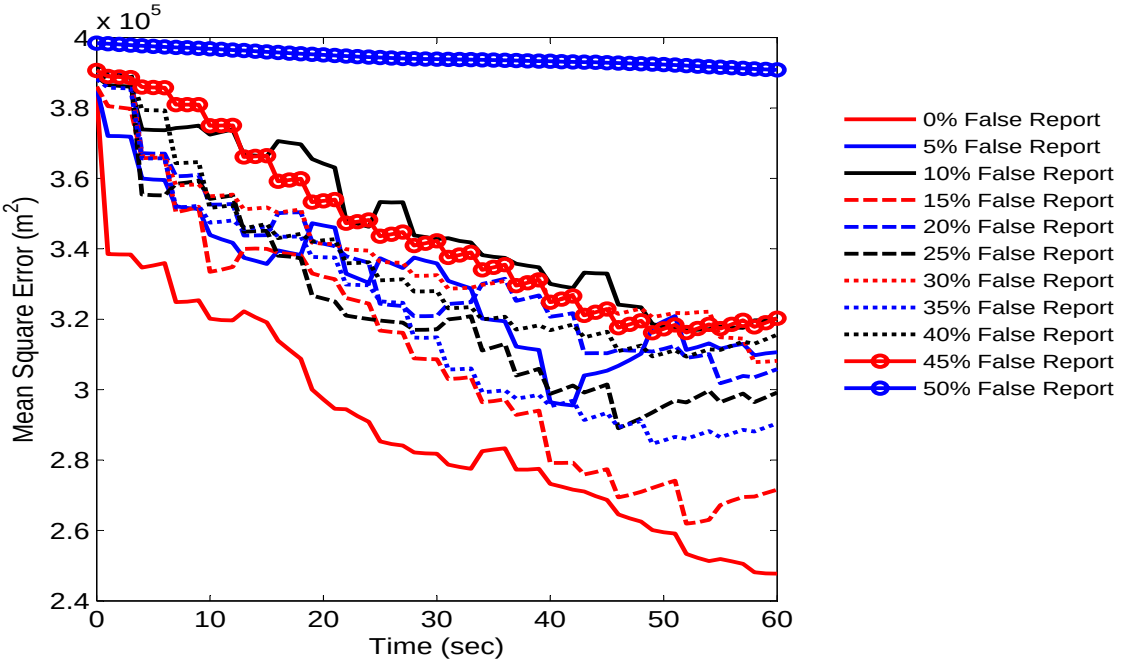


Figure 4.15: Traffic Model, Winding Path, False Report Belief Study, 20% False Report Sensor

Similar to results in the case of a straight path, the results obtained for a target on a winding path indicate that the Dispersion Model should be tuned to a 30% false report belief and the Traffic Model to 0% or 15% false report belief. Both motion models suffered from the elevated false start rate. The Traffic Motion model yielded convergent tracking error, while tracking error diverged in the case of the the Dispersion Model.

Detuning the particle filter's belief in the sensor's false report rate provided increased spatial resolution of the particle cloud. This improved the time to detect metric in the winding target scenario, as it allowed for all particles in a region of non-detection to not be eliminated immediately, providing for increased particle presence in the proximity of a target just outside of a measured region. In the presence of a sensor with a false report rate, the tracking performance is significantly diminished, regardless of the motion model that is used. But, the Traffic Motion Model out performed the Dispersion Model's tracking performance in the presence of each sensor model. Section 4.2 will explore the effect of decreasing the

Table 4.7: Time to Detect Results, Winding Path, 20% False Report Sensor, False Report Belief 0%-50%

False Report Belief	Time to Detect Dispersion	% False Starts Dispersion	Time to Detect Traffic	%False Starts Traffic
0%	13.05 sec	78.50	13.35 sec	87.50
5%	12.57 sec	74.50	14.40 sec	82.50
10%	13.62 sec	77.50	13.39 sec	83.00
15%	13.23 sec	70.50	12.75 sec	84.50
20%	12.34 sec	80.00	12.81 sec	86.50
25%	12.73 sec	75.50	13.57 sec	86.00
30%	11.92 sec	74.50	13.36 sec	83.00
35%	13.13 sec	73.00	12.36 sec	84.50
40%	12.34 sec	78.50	12.87 sec	86.50
45%	12.37 sec	75.50	13.29 sec	84.00
50%	14.13 sec	97.50	13.77 sec	95.50

required number of effective particles to further study the effect of spatial resolution on the tracking performance and false alarm rate in the presence of a sensor with a false report rate.

4.2 Number of Effective Particles Study

For example, if a detection is reported, weight is shifted to the particles within the measured region and the region will be measured again, as it continues to maintain the highest particle weight sum. However, if the number of effective particles remains below a desired threshold, the particle cloud is not resampled. If the particles are not resampled and the measurement was a false detection, the following measurement has a high probability of reporting the truth and the particle cloud will maintain its previous coverage of the simulation space. This will improve tracking performance, but will not provide for a means of rejecting false reports, as all measurements are still believed to be true.

All simulations were done with 1000 particles, and all curves reflect the average value of the mean square error for a case.

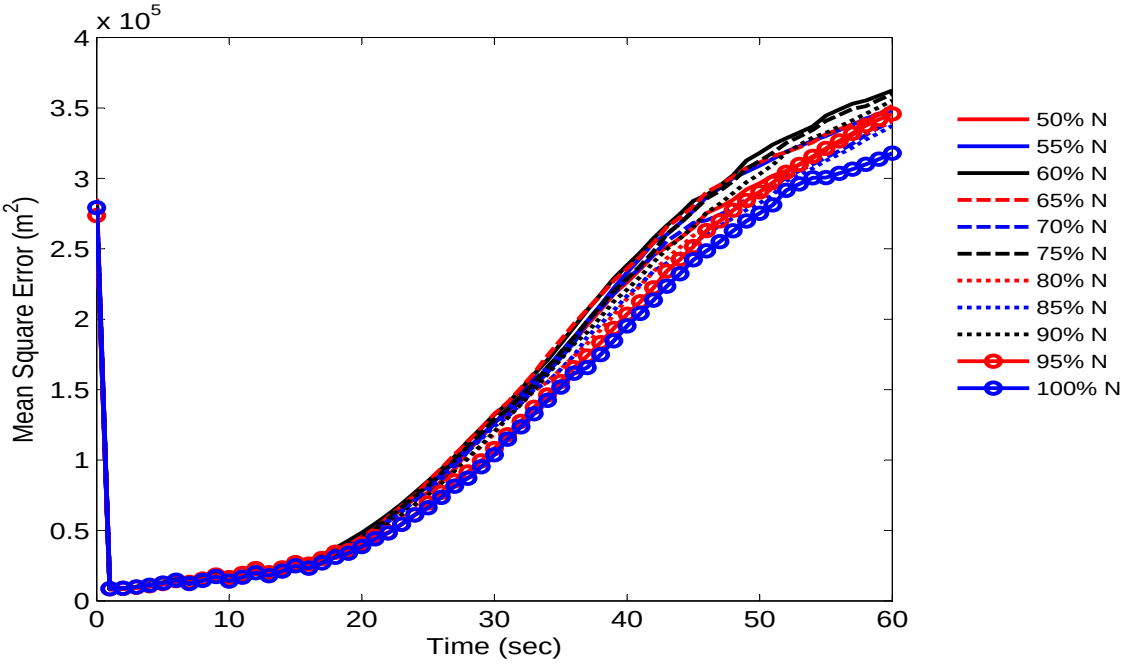


Figure 4.16: Dispersion Model, Straight Path, Number of Effective Particles Study, Perfect Sensor

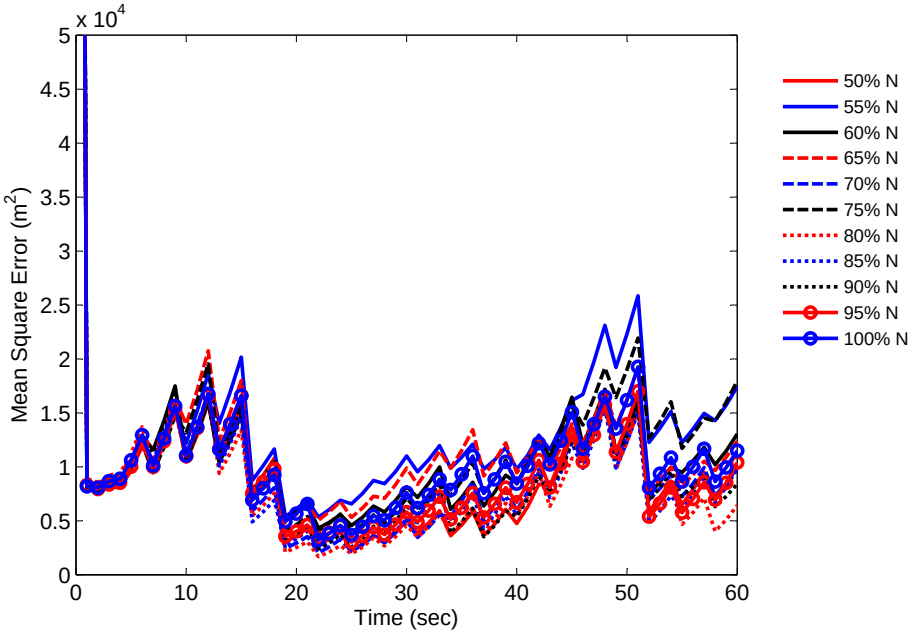


Figure 4.17: Traffic Model, Straight Path, Number of Effective Particles Study, Perfect Sensor

In the presence of a perfect sensor and a target on a straight path, improved tracking results are achieved for the Dispersion Model when the particle distribution is resampled after each measurement, as consistent with the $N_{thr} = N$. The Traffic Model displays its best performance when the threshold is set at 85%N. The detection times indicate that faster target localization results from resampling following each measurement and setting $N_{thr} = N$. This result could be used in future work to adapt the required number of effective particles to the phase of the application: localization or tracking.

Table 4.8: Time to Detect Results, Straight Path, Perfect Sensor, $N_{thr} = (100\% - 50\%)N$

Resample Threshold	**Time to Detect Dispersion**	**Time to Detect Traffic**
100% N	51.71 sec	33.33 sec
95% N	58.87 sec	32.86 sec
90% N	56.61 sec	35.38 sec
85% N	57.54 sec	35.97 sec
80% N	60.93 sec	29.89 sec
75% N	50.64 sec	36.54 sec
70% N	60.41 sec	34.14 sec
65% N	48.47 sec	36.91 sec
60% N	52.09 sec	38.07 sec
55% N	53.19 sec	30.39 sec
50% N	48.34 sec	27.73 sec

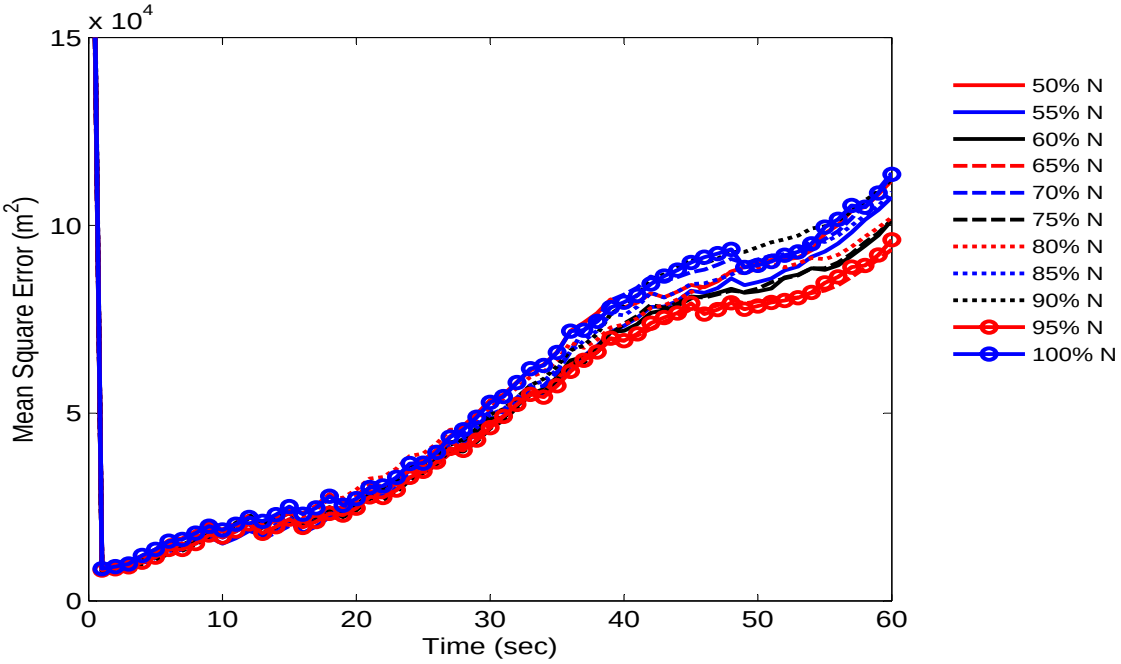


Figure 4.18: Dispersion Model, Winding Path, Number of Effective Particles Study, Perfect Sensor

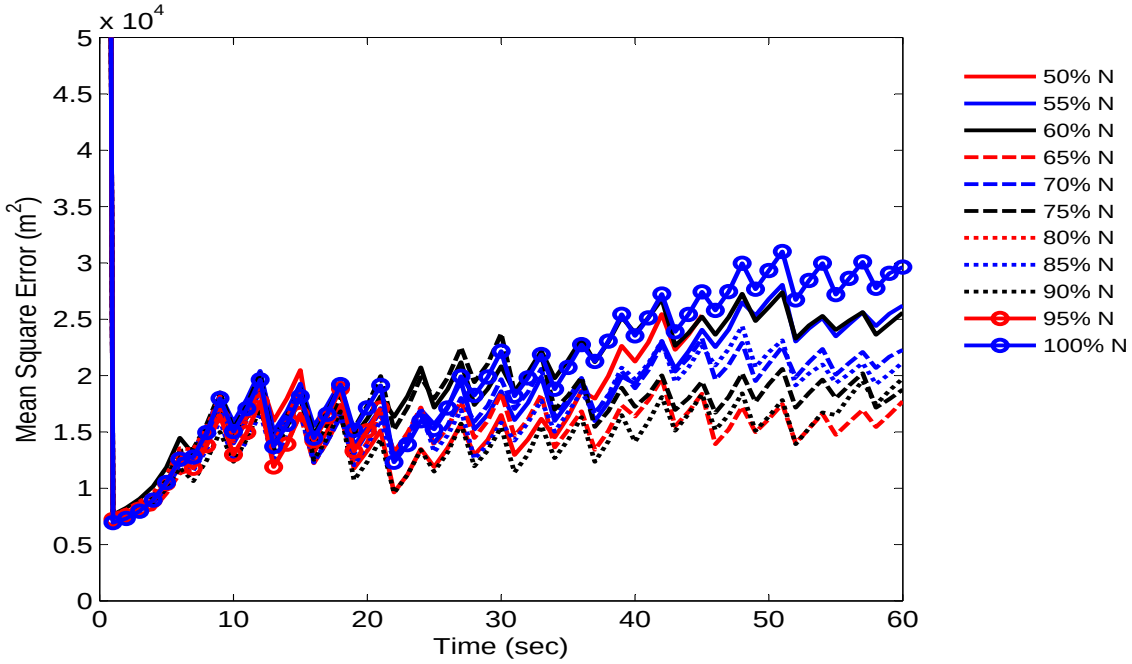


Figure 4.19: Traffic Model, Winding Path, Number of Effective Particles Study, Perfect Sensor

In the case of a winding path, the Dispersion Model benefits from tuning its required number of effective particles to 95%N while the Traffic Model benefits from a 90%N threshold. The times to detect at the tuned value of required number of effective particles are similar to the minimal value for each motion model in the case of a winding target path.

Table 4.9: Time to Detect Results, Winding Path, Perfect Sensor, $N_{thr} = (100\% - 50\%)N$, False Report Belief 0%

Resample Threshold	**Time to Detect Dispersion**	**Time to Detect Traffic**
100% N	59.37 sec	42.11 sec
95% N	60.99 sec	44.55 sec
90% N	73.02 sec	44.83 sec
85% N	53.10 sec	44.46 sec
80% N	65.64 sec	50.95 sec
75% N	54.11 sec	46.13 sec
70% N	64.29 sec	45.18 sec
65% N	61.65 sec	42.63 sec
60% N	61.88 sec	45.71 sec
55% N	68.51 sec	50.26 sec
50% N	58.06 sec	49.74 sec

The inclusion of a sensor with false report rate is expected to diminish tracking performance and reduce the detection time, as false positives are believed true. The effect of reducing the required number of effective particles is illustrated by the figures below. A sensor with a false report rate of 10% and 20% were studied, and in each case, the particle filter's belief of the sensor's false report rate was set to its actual false report rate.

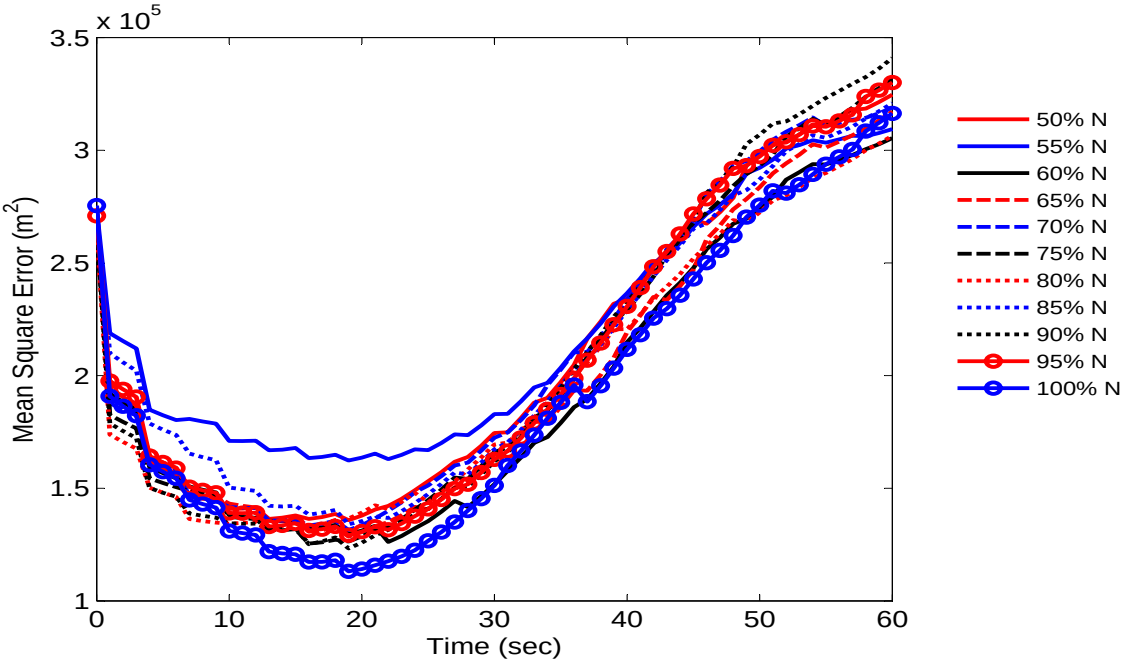


Figure 4.20: Dispersion Model, Straight Path, Number of Effective Particles Study, 10% False Report Sensor

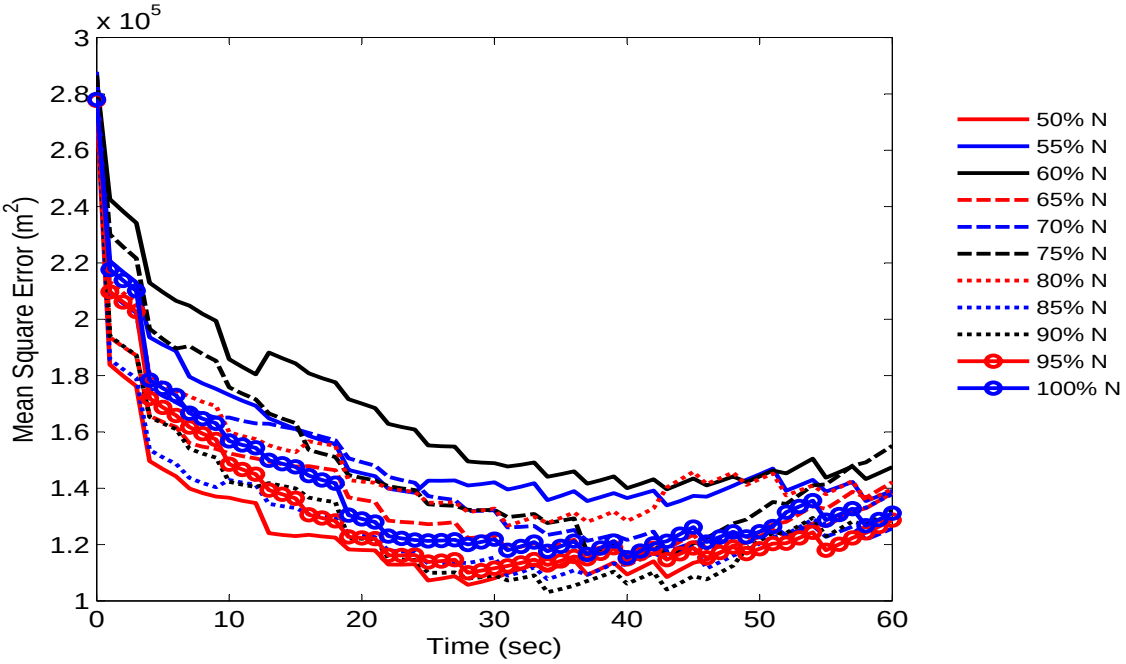


Figure 4.21: Traffic Model, Straight Path, Number of Effective Particles Study, 10% False Report Sensor

Although the tracking performance is much weaker than that resulting from the use of a perfect sensor, the number of effective particles play a role in the spatial resolution of the particle cloud and can increase tracking performance. The Dispersion Model does not benefit from the reduction in the amount of required effective particles, but the Traffic Motion Model achieves improved tracking performance with a required number of effective particles of 50%N.

Table 4.10: Time to Detect Results, Straight Path, Perfect Sensor, $N_{thr} = (100\% - 50\%)N$, False Report Belief 10%

Resample Threshold	**Time to Detect Starts**	**% False Dispersion**	**Time to Detect Traffic**	**% False Starts Traffic**
100% N	15.60 sec	55.50	21.37 sec	56.50
95% N	21.03 sec	57.00	20.45 sec	58.00
90% N	16.95 sec	56.50	18.06 sec	57.00
85% N	17.53 sec	63.50	20.23 sec	60.50
80% N	19.21 sec	58.00	23.34 sec	62.50
75% N	19.06 sec	50.00	20.33 sec	63.00
70% N	19.91 sec	58.00	20.59 sec	61.00
65% N	20.52 sec	59.00	19.56 sec	60.00
60% N	17.12 sec	53.50	19.59 sec	66.00
55% N	16.98 sec	54.50	18.73 sec	59.50
50% N	16.78 sec	55.50	19.98 sec	57.50

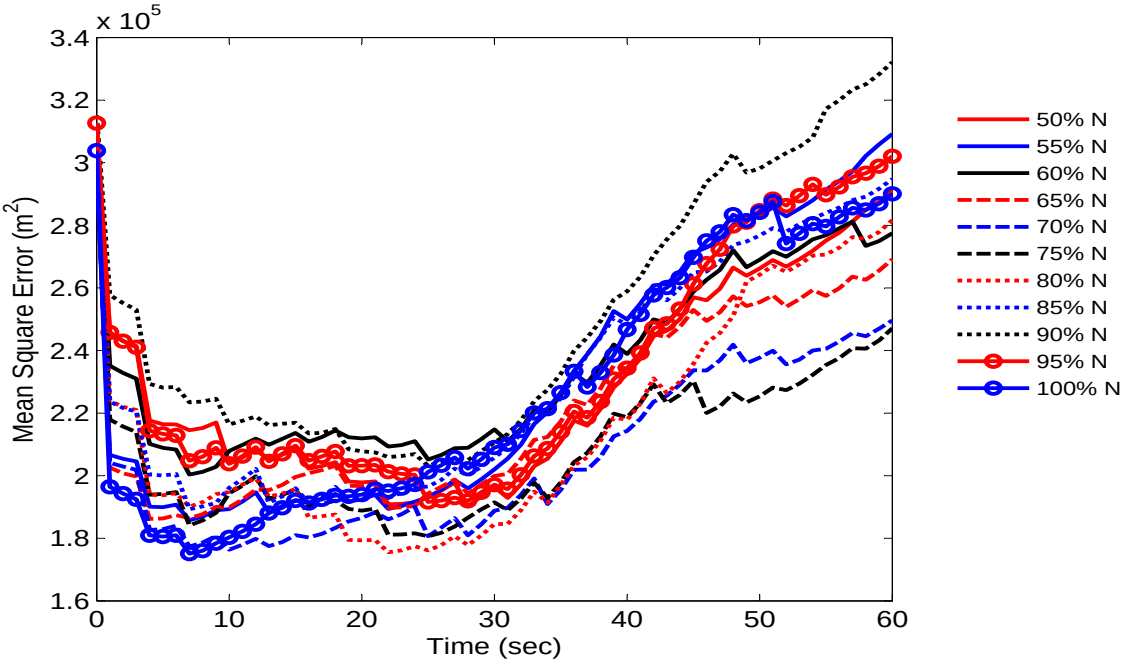


Figure 4.22: Dispersion Model, Winding Path, Number of Effective Particles Study, 10% False Report Sensor

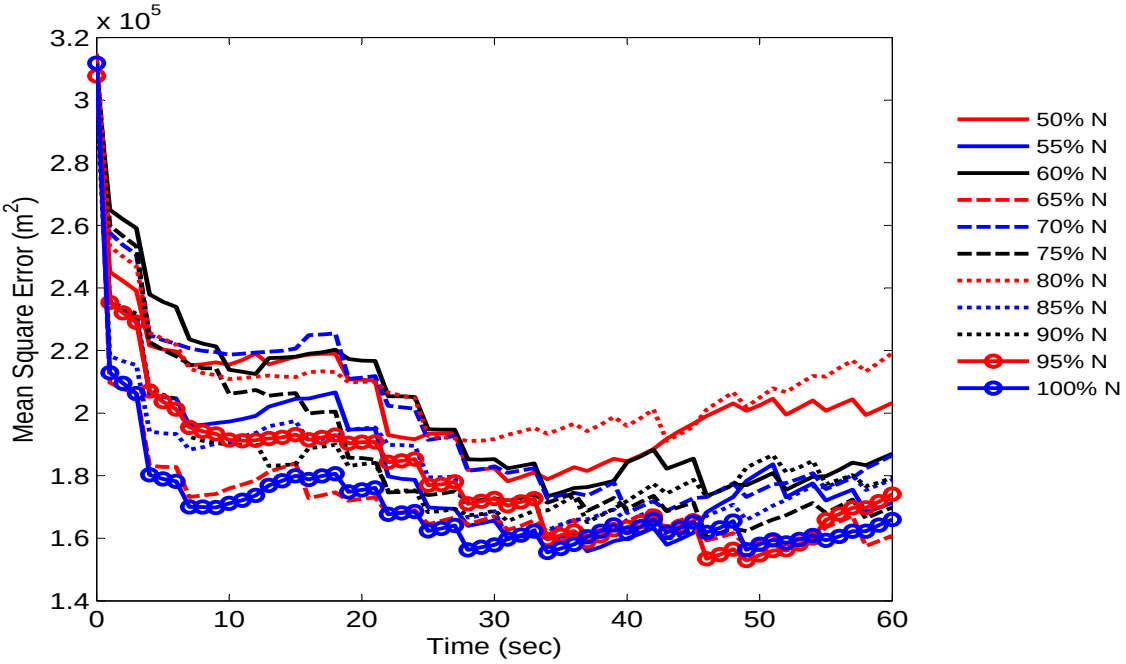


Figure 4.23: Traffic Model, Winding Path, Number of Effective Particles Study, 10% False Report Sensor

In the case of a target on a winding path, the Dispersion Model exhibits its best tracking results when the required number of particles is set to 70%N, whereas the Traffic Model does not benefit from a decreased number of effective particles.

Table 4.11: Time to Detect Results, Winding Path, 10% False Report Sensor, $N_{thr} = (100\% - 50\%)N$, False Report Belief 10%

Resample Threshold	**Time to Detect Starts**	**% False Dispersion**	**Time to Detect Traffic**	**% False Starts Traffic**
100% N	18.19 sec	59.00	18.06 sec	60.50
95% N	19.45 sec	55.00	21.49 sec	69.50
90% N	18.52 sec	59.00	22.58 sec	66.00
85% N	20.85 sec	55.50	21.99 sec	72.00
80% N	17.59 sec	54.50	18.93 sec	68.50
75% N	20.96 sec	61.50	18.47 sec	62.50
70% N	15.42 sec	57.00	19.85 sec	66.50
65% N	18.05 sec	56.50	22.83 sec	59.50
60% N	17.97 sec	55.50	22.75 sec	68.50
55% N	19.80 sec	52.50	22.38 sec	64.00
50% N	19.43 sec	56.50	19.14 sec	69.50

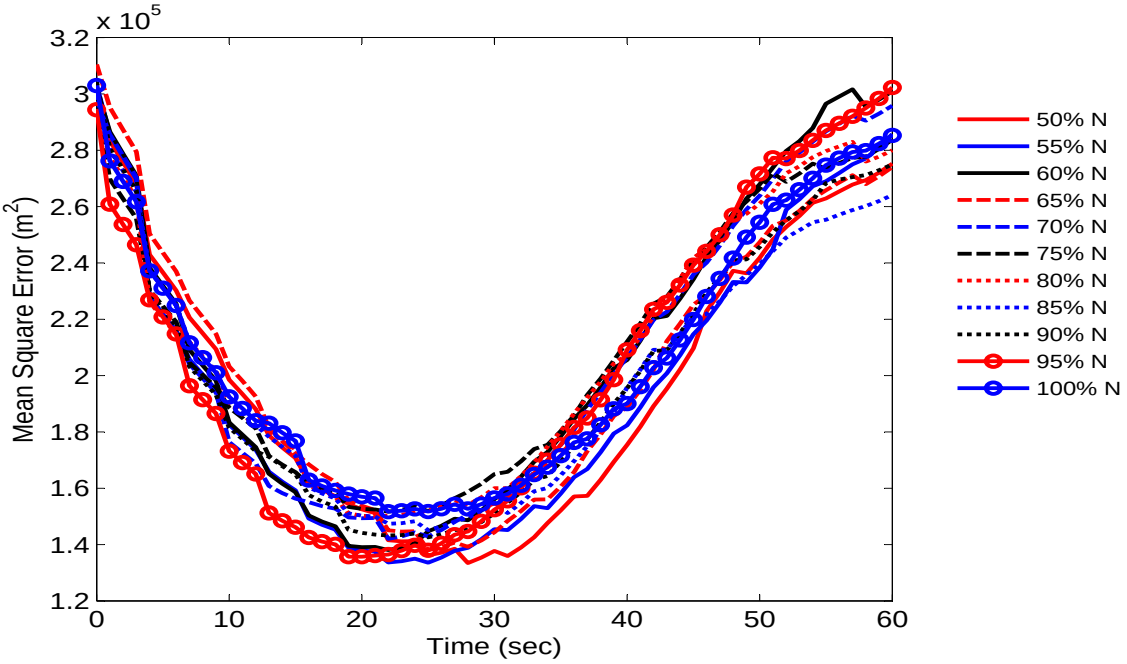


Figure 4.24: Dispersion Model, Straight Path, Number of Effective Particles Study, 20% False Report Sensor

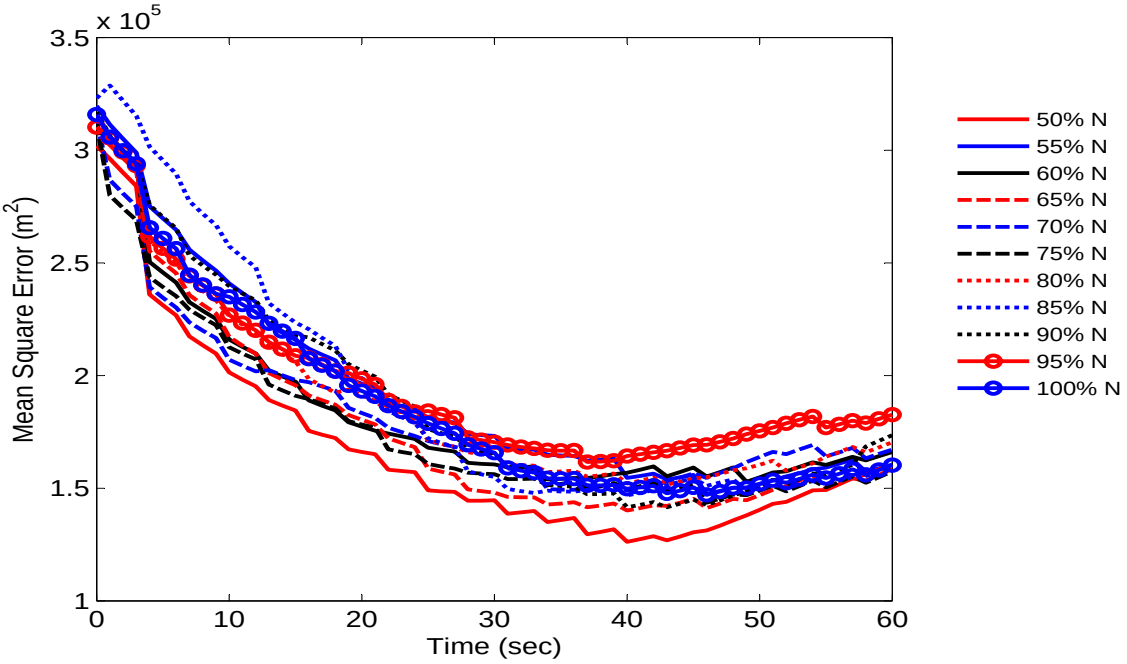


Figure 4.25: Traffic Model, Straight Path, Number of Effective Particles Study, 20% False Report Sensor

The figures above indicate that the Dispersion Model achieves improved tracking performance with a 70%N effective particle threshold, while the Traffic Motion Model performs best with a 50% effective particle threshold. The false alarm rates and diminished tracking performance are consistent with the results in Section 4.1. It remains true that believing a false report to be true is damaging to the tracking performance.

Table 4.12: Time to Detect Results, Straight Path, 20% False Report Sensor, $N_{thr} = (100\% - 50\%)N$, False Report Belief 20%

Resample Threshold	**Time to Detect Starts**	**% False Dispersion**	**Time to Detect Traffic**	**% False Starts Traffic**
100% N	12.36 sec	78.50	12.49 sec	77.50
95% N	12.55 sec	76.00	13.45 sec	78.50
90% N	12.07 sec	82.00	11.52 sec	83.00
85% N	12.16 sec	76.00	11.34 sec	85.50
80% N	12.60 sec	81.00	13.11 sec	79.00
75% N	12.54 sec	80.50	12.84 sec	78.00
70% N	12.81 sec	75.00	13.12 sec	76.00
65% N	11.43 sec	80.50	12.46 sec	77.50
60% N	11.73 sec	73.50	13.11 sec	82.50
55% N	11.05 sec	76.50	11.83 sec	82.50
50% N	12.30 sec	80.00	14.48 sec	79.50

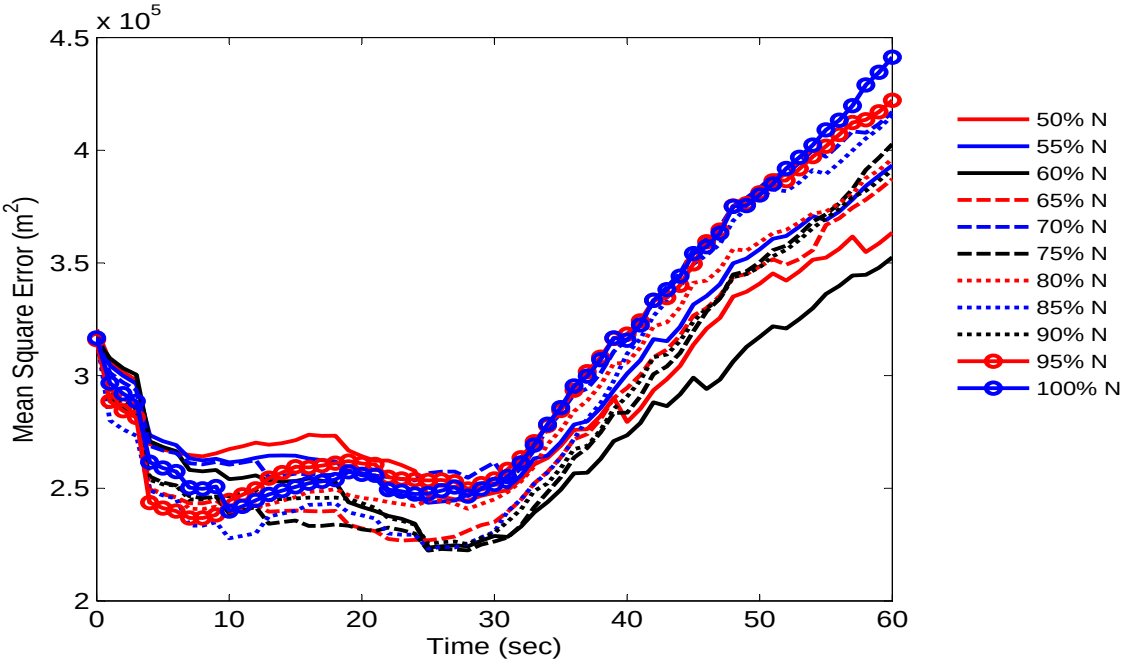


Figure 4.26: Dispersion Model, Winding Path, Number of Effective Particles Study, 20% False Report Sensor

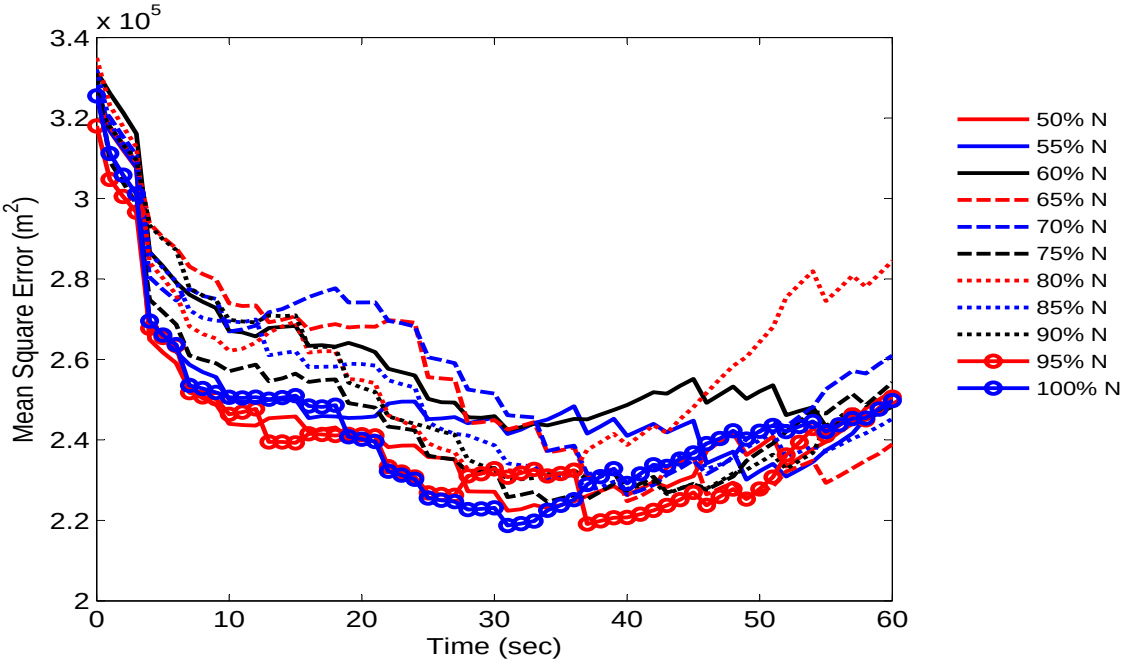


Figure 4.27: Traffic Model, Winding Path, Number of Effective Particles Study, 20% False Report Sensor

Similar to the results for a target on a straight path, the Dispersion Model benefits from a 75%N effective particle threshold and the Traffic Model benefits from a 50%N effective particle threshold.

Table 4.13: Time to Detect Results, Winding Path, 20% False Report Sensor, $N_{thr} = (100\% - 50\%)N$, False Report Belief 20%

Resample Threshold	**Time to Detect Starts**	**% False Dispersion**	**Time to Detect Traffic**	**% False Starts Traffic**
100% N	11.89 sec	79.50	12.79 sec	78.00
95% N	13.75 sec	79.00	13.90 sec	81.00
90% N	12.81 sec	80.00	12.90 sec	82.00
85% N	13.20 sec	75.50	12.63 sec	85.00
80% N	14.75 sec	76.00	11.20 sec	81.50
75% N	12.87 sec	79.50	11.70 sec	83.00
70% N	13.58 sec	78.00	13.87 sec	81.50
65% N	14.05 sec	77.00	13.74 sec	83.50
60% N	13.38 sec	81.00	12.15 sec	84.00
55% N	12.27 sec	83.50	12.70 sec	81.50
50% N	11.86 sec	78.50	14.22 sec	78.50

The number of effective particles may be tuned to improve performance, however false sensor reports continue to diminish tracking performance. Decreasing the required number of effective particles reduces the frequency of resampling and therefore forces a broader probability density function and widens the spatial resolution of the distribution. When a false detection is reported, significant weight is shifted to the measured region. That region will be measured at the next measurement as it contains the highest sum of particle weight. If a false detection is reported and the required number of effective particles is low (50-75%N), significant particle coverage outside of the measured region is maintained. Upon the next measurement of the detection region, it is likely that another false detection will not be reported, causing particle weight to be shifted from that region back to the particles outside the measured region that were preserved by not resampling. Then, the regional sensor polling problem continues and other areas of the map are searched for the target. In this situation, however, the tracking performance would have begun to be measured at the

report of the false detection. This results in perceived poor tracking performance, as the target was not actually found at the start of the 60 second observation period. After a false detection is reported, the 60 second observation period is often spent searching the rest of the map for the target, therefore preventing tracking performance similar to that of a perfect sensor. The frequency of this increases as the false report rate of the sensor increases.

Because of the significant effect of false reports on the tracking performance, an additional means of improving performance was investigated. A measure of risk was developed to determine the potential impact of accepting a measurement and resampling the particle distribution. By the inclusion of a means to reject a possibly false measurement, the importance of the number of effective particles and sensor belief will come to light. Without reducing the number of effective particles and sensor belief in the presence of a sensor with a false report rate, the first false detection would result in completely loss of particle resolution outside of the measured region, and therefore would result in a severely increased true detection time.

4.3 Bayesian Risk

A Bayesian Risk Assessment was used to evaluate the risk of accepting a measurement as true, based on an example where the expected change in the estimate variance was used to determine the cost of a accepting a measurement [33]. In this application, the expected flux of particle weight into or out of the measured region will be used as the cost of resampling after a measurement. In the formulation presented by O'Reilly in [33], two hypotheses are considered: the measurement is true (H_0) and the measurement has been altered (H_1). A ratio of the likelihoods of each hypothesis given the data, $\mathbf{d}$, is computed and denoted as LR, the likelihood ratio.

$$LR = \frac{p\left(\mathbf{d}|H_1\right)}{p\left(\mathbf{d}|H_0\right)} \tag{4.6}$$

In order to determine which hypothesis to accept, the likelihood ratio is compared to a threshold, denoted τ.

$$\text{if } \frac{p\left(\mathbf{d}|H_1\right)}{p\left(\mathbf{d}|H_0\right)} < \tau, \text{ then } H_1 \text{ is accepted} \tag{4.7}$$

$$\text{if } \frac{p\left(\mathbf{d}|H_1\right)}{p\left(\mathbf{d}|H_0\right)} \geq \tau, \text{ then } H_0 \text{ is accepted}$$

The threshold τ may be formulated in the Bayesian framework based on the cost of accepting a certain measurement and the probability associated with each hypothesis.

$$\tau = \frac{\left(C_{10} - C_{00}\right) P_{H0}}{\left(C_{01} - C_{11}\right) P_{H1}} \tag{4.8}$$

The cost variable C_{XY} is the cost of accepting hypothesis H_X if the hypothesis H_Y is actually true. There is usually no cost associated with accepting a true hypothesis, in which case the terms C_{00} and C_{11} are zero.

The framework outlined by the previous equations provide a means for assigning a value of risk to each measurement. Risk is based on the likelihood of a measurement being true, taking into account the amount of particle weight that would shift if a measurement is accepted. If a measurement is deemed to risky to accept, the particle filter accepts the measurement but does not resample the distribution. This preserves all particles outside of the measured region. The risk assessment was used to control when to resample instead of whether or not to accept a measurement. This was done because if a measurement is correct, a repeated measurement will confirm it and when the risk decreases to an acceptable level, the distribution is resampled. If a measurement was false, a repeated measurement will often bring this to light and the spatial resolution of the particle cloud will not have been compromised.

To examine the risk, the likelihood ratio must be established in terms associated with the particle filter. The likelihood ratio is computed based on the expressions below for the likelihoods that the target is actually present within the measured region (p(present)) or not

(p(not present)).

$$\begin{aligned} p\,(\text{present}) &= (1 - R_{false}) \sum_{i=1}^{N_{in}} w_{in}^{i} + (R_{false})(1 - \sum_{i=1}^{N_{in}} w_{in}^{i}) \\ p\,(\text{not present}) &= (R_{false}) \sum_{i=1}^{N_{in}} w_{in}^{i} + (1 - R_{false})(1 - \sum_{i=1}^{N_{in}} w_{in}^{i}) \end{aligned} \tag{4.9}$$

When a detection is reported, the likelihood ratio is given by:

$$LR = \frac{p\,(\text{not present})}{p\,(\text{present})} \tag{4.10}$$

When a non-detection is reported, the likelihood ratio is given by:

$$LR = \frac{p\,(\text{present})}{p\,(\text{not present})} \tag{4.11}$$

Expressions for the cost of accepting a measurement are derived below. The terms C_{00} and C_{11} in Eq. (4.8) are set to zero because there is no cost associated with accepting a true measurement. The terms P_{H0} and P_{H1} depend on the received measurement. If a detection is reported, P_{H0} is the sum of the weights of the particles inside the measured region, and P_{H1} is the sum of the weights of the particles outside of the measured region. If a non-detection is reported, P_{H0} and P_{H1} are the sum of the weights of the particles outside the measured region and the sum of the particles outside the measured region, respectively.

The cost associated with accepting a false measurement is quantified by the amount of particle weight that would shift if a measurement was accepted. In the case of a detection

being reported, the cost of accepting the measurement if it is false is computed by:

$$\Delta_{in} = \frac{\left|(\sum_{i=1}^{N_{in}} w_{in}^{i}) - p(+|\text{present})\right|}{N_{in}} \tag{4.12}$$

$$\Delta_{out} = \frac{\left|(1 - \sum_{i=1}^{N_{in}} w_{in}^{i} - p(-|\text{present})\right|}{N_{out}} \tag{4.13}$$

$$\tau = \frac{\Delta_{out} \sum w_{in}}{\Delta_{in}(1 - \sum_{i=1}^{N_{in}} w_{in}^{i})} \tag{4.14}$$

In the case of a non-detection being reported, the cost of accepting the measurement if it is false is determined by the following expressions:

$$\Delta_{in} = \frac{\left|(1 - \sum_{i=1}^{N_{in}} w_{in}^{i}) - p(-|\text{not present})\right|}{N_{in}} \tag{4.15}$$

$$\Delta_{out} = \frac{\left|\sum_{i=1}^{N_{in}} w_{in}^{i} - p(+|\text{not present})\right|}{N_{out}} \tag{4.16}$$

$$\tau = \frac{\Delta_{out}(1 - \sum_{i=1}^{N_{in}} w_{in}^{i})}{\Delta_{in} \sum_{i=1}^{N_{in}} w_{in}^{i}} \tag{4.17}$$

In a situation where all particle weight is located in a region and a non-detection is reported, it is obvious that the measurement is most likely false. Also, if a region of very low particle weight is polled and a detection is reported, it would be risky to believe such a measurement. However, in such instances where there is significant particle weight in each region, it is not immediately clear if a measurement should be accepted or rejected. An example of such a case is outlined for completion. The risk associated with a reported detection and a reported non-detection will be examined if a measurement is taken of the region in the bottom right corner in Fig. 4.28. In the example illustrated by Fig. 4.28, it is believed that the sensor has a false report rate of 10% and all particles have equal weight of 1/8.

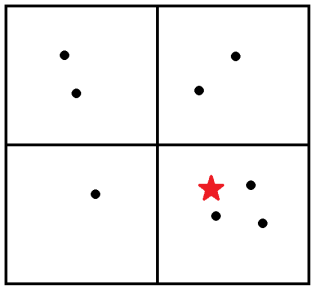


Figure 4.28: Measurement Update with Risk Assessment Example Scenario

If a detection is reported in the bottom right region of Fig. 4.28, the likelihood ratio is computed as follows:

$$\begin{aligned} p(\text{present}) &= (0.9)(3/8) + (0.1)(5/8) \\ p(\text{not present}) &= (0.1)(3/8) + (0.9)(5/8) \\ LR &= \frac{p\,(\text{not present})}{p\,(\text{present})} = 1.5 \end{aligned} \tag{4.18}$$

The cost ratio, τ, is then calculated.

$$p(+|\text{present}) = \frac{(1 - R_{false}) \sum w_{in}}{p(\text{present})} = 0.8438 \tag{4.19}$$

$$\begin{aligned} \Delta_{in} &= \frac{|(\sum w_{in}) - p(+|\text{present})|}{N_{in}} = 0.1563 \\ p(-|\text{present}) &= \frac{(1 - R_{false})(1 - \sum w_{in})}{p(\text{not present})} = 0.9375 \\ \Delta_{out} &= \frac{|(1 - \sum w_{in} - p(-|\text{not present})|}{N_{out}} = 0.0625 \\ \tau &= \frac{\Delta_{out} \sum w_{in}}{\Delta_{in}(1 - \sum w_{in})} = 0.2400 \end{aligned} \tag{4.20}$$

Because τ is less than the likelihood ratio, the particle cloud would not be resampled as the risk is too high.

If the sensor reports that the target is not detected in the bottom right corner region, that is a false report. However, there is significant particle presence in other sensor regions,

so it is not immediately clear to the particle filter that the measurement is not likely true. The likelihood ratio is the inverse of that of a detection, and is therefore 2/3.

The cost ratio, τ, is then calculated.

$$
\begin{aligned}
p(-|\text{not present}) &= \frac{R_{false} \sum w_{in}}{p(\text{not present})} = 0.0625 \qquad (4.21)\\
\Delta_{in} &= \frac{\left|(1 - \sum_{i=1}^{N_{in}} w_{in}^{i}) - p(-|\text{not present})\right|}{N_{out}} = 0.1125\\
p(+|\text{present}) &= \frac{R_{false}(1 - \sum_{i=1}^{N_{in}} w_{in}^{i})}{p(\text{not present})} = 0.0625 \qquad (4.22)\\
\Delta_{out} &= \frac{\left|(1 - \sum_{i=1}^{N_{in}} w_{in}^{i} - p(-|\text{present})\right|}{N_{in}} = 0.0729\\
\tau &= \frac{\Delta_{out}(1 - \sum_{i=1}^{N_{in}} w_{in}^{i})}{\Delta_{in} \sum_{i=1}^{N_{in}} w_{in}^{i}} = 1.0802
\end{aligned}
$$

Because the likelihood ratio is less than τ, the particle distribution would be resampled if the number of effective particles is below the desired threshold.

4.3.1 Risk Assessment Results

The Bayesian Risk Assessment discussed above was implemented in the cases of the two sensors with false report rates. The following values were used for the false report belief and the number of effective particles:

The values in Table 4.14 were determined in Chapter 4. It is evident that the number of effective particles should be reduced in the presence of an imperfect sensor and the false report belief should be set to a value greater than or equal to the true false report rate. Although the results in Section 4.1 indicate that the false report belief for the Traffic Motion Model in the presence of a sensor with a 20% false repot rate should be set to 0%, the false report belief for the Traffic Motion Model was set to 20% in this section. This was done because in Section 4.2, the false report belief was set to the true false report rate of each sensor, and superior tracking performance occurred in the case of the 20% false report sensor.

Table 4.14: Tuned Values for Sensor Belief and Number of Effective Particles

Motion Model	**Target Path**	**Sensor Model**	**False Report Belief**	**Number of Effective Particles**
Dispersion	Straight	10% False Report	5%	60%N
Dispersion	Winding	10% False Report	5%	70%N
Traffic	Straight	10% False Report	10%	50%N
Traffic	Winding	10% False Report	25%	65%N
Dispersion	Straight	20% False Report	35%	55%N
Dispersion	Winding	20% False Report	30%	60%N
Traffic	Straight	20% False Report	20%	50%N
Traffic	Winding	20% False Report	20%	50%N

In an effort to reduce the false alarm percentage, the time to detect was not established until two consecutive detection measurements were reported. Any time a false alarm is recorded, the second detection in the sequence is a false detection. Data regarding the time to detect the target, tracking performance, and the decrease in false start rates is presented to show the importance of the Bayesian Risk Assessment when a sensor with a false report rate is supplying information.

In each figure, the results are presented for the optimal values found for the sensor belief and number of effective particles found in Chapter 4 for each motion model. Results are given without the risk assessment, with the risk assessment, and the results for a perfect sensor are also plotted for comparison.

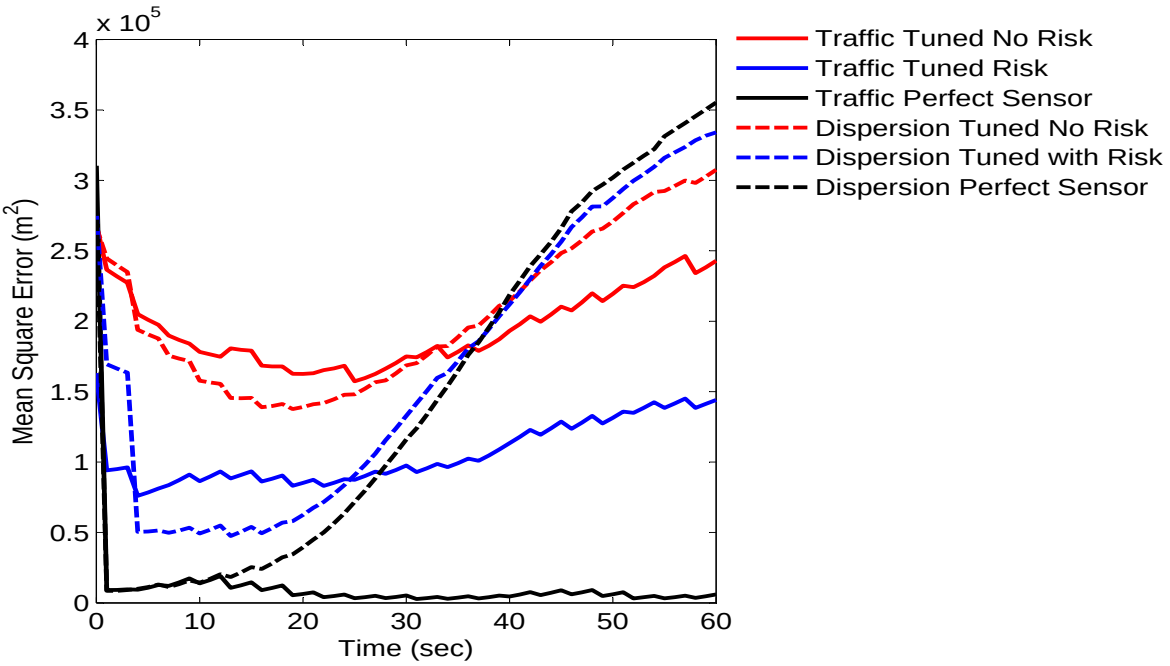


Figure 4.29: Effect of Risk Assessment, Straight Path, 10% False Report Sensor

Figure 4.29 illustrates the benefit to the tracking performance provided by the risk assessment in the case of the Traffic Motion Model. By not resampling after a possibly false measurement, the spatial resolution of the particle cloud is preserved, and the tracking performance greatly improves when compared to the error values when the risk assessment is not included. The Dispersion Model yields poor tracking results in the case of a straight path, regardless of the particle filter's measurement update and inclusion of risk assessment. The frequency of believing a false detection greatly decreased for both motion models. As expected, this increased the time to detect when compared to the case of a perfect sensor and the case of when all measurements are immediately accepted.

Table 4.15: Time to Detect and False Start Results, Straight Path, 10% False Report Sensor

Motion Model	Risk Assessment	Sensor Model	Time to Detect	Percent False Starts
Traffic	No	10% False Report	22.53 sec	70.50
Traffic	Yes	10% False Report	70.92sec	19.50
Traffic	No	Perfect	28.80 sec	0
Dispersion	No	10% False Report	16.47 sec	45.00
Dispersion	Yes	10% False Report	33.69 sec	10.00
Dispersion	No	Perfect	57.90 sec	0

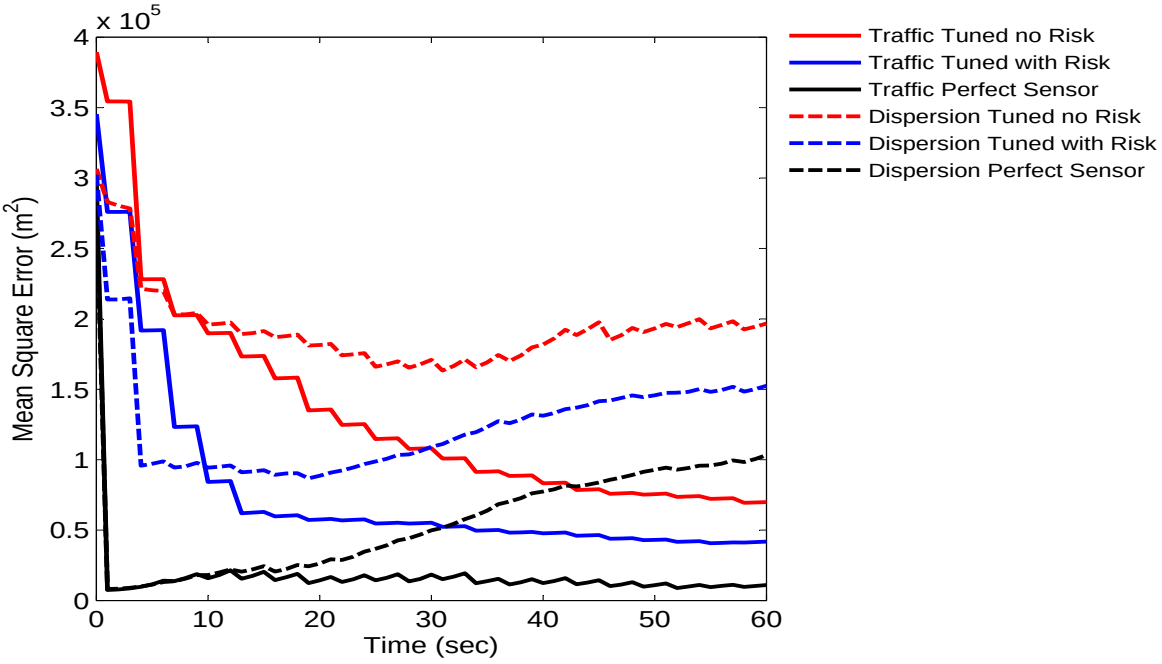


Figure 4.30: Effect of Risk Assessment, Winding Path, 10% False Report Sensor

Figure 4.32 illustrates the improved tracking performance that results from the implementation of the risk assessment in both motion models. The Traffic Motion Model continues to provide improved tracking performance.

Table 4.16: Time to Detect and False Start Results, Winding Path, 10% False Report Sensor

Motion Model	Risk Assessment	Sensor Model	Time to Detect	Percent False Starts
Traffic	No	10% False Report	19.68 sec	61.00
Traffic	Yes	10% False Report	71.62 sec	21.00
Traffic	No	Perfect	32.20 sec	0
Dispersion	No	10% False Report	17.29 sec	55.00
Dispersion	Yes	10% False Report	39.19 sec	14.00
Dispersion	No	Perfect	67.70 sec	0

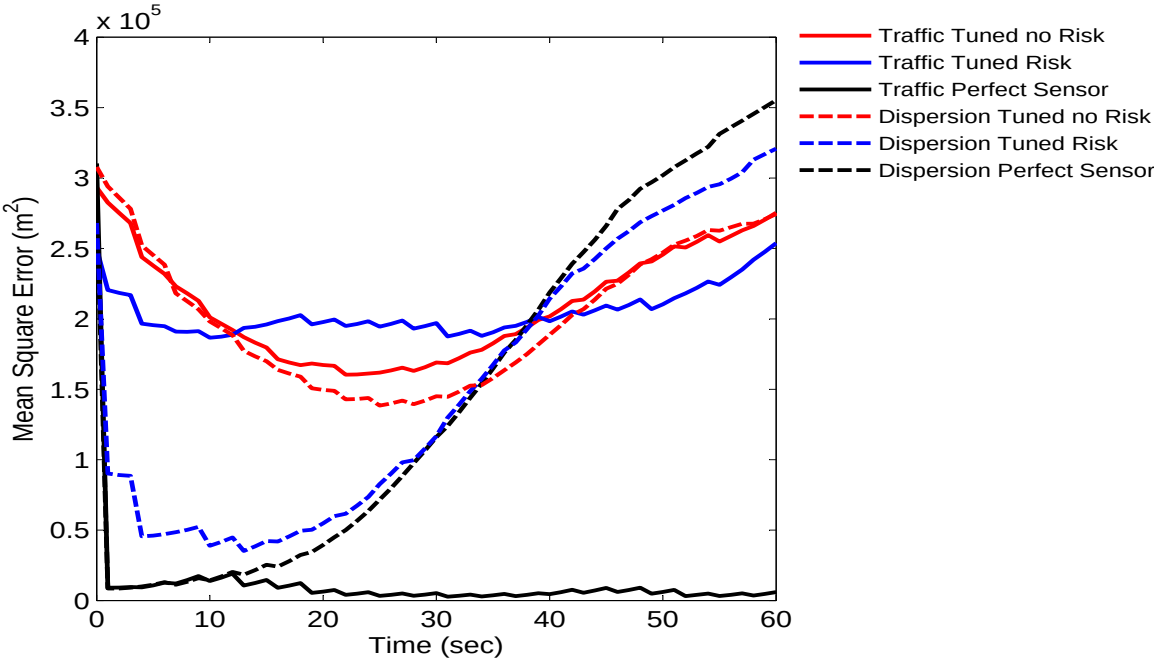


Figure 4.31: Effect of Risk Assessment, Straight Path, 20% False Report Sensor

The tracking performance in the case of a straight path improved with the addition of the risk assessment, and the requirement to only start observing tracking behavior after two successive reported detections. The Dispersion Model did not experience as great a decrease in the number of false starts as the Traffic Motion Model, but the tracking performance was improved. The Traffic Motion Model had a higher percentage of false starts, but the presence of the risk assessment aided in tracking performance for both motion models.

Table 4.17: Time to Detect and False Start Results, Straight Path, 20% False Report Sensor

Motion Model	Risk Assessment	Sensor Model	Time to Detect	Percent False Starts
Traffic	No	10% False Report	22.53 sec	70.50
Traffic	Yes	10% False Report	56.02sec	58.00
Traffic	No	Perfect	28.80 sec	0
Dispersion	No	10% False Report	16.47 sec	45.00
Dispersion	Yes	10% False Report	40.78 sec	42.00
Dispersion	No	Perfect	57.90 sec	0

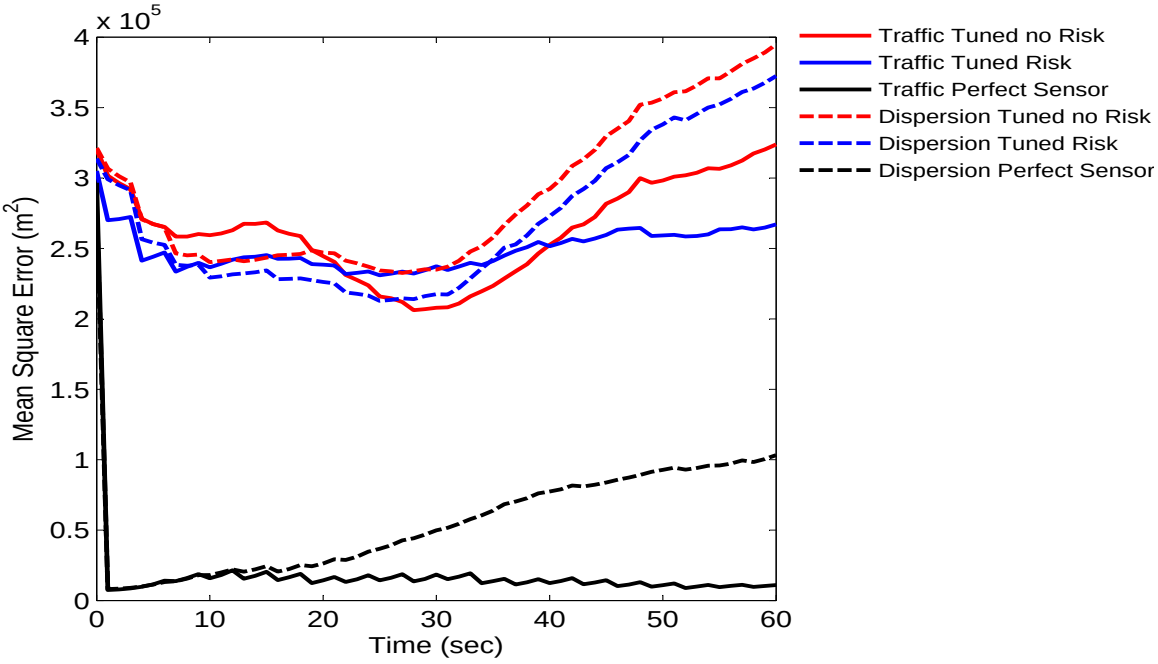


Figure 4.32: Effect of Risk Assessment, Winding Path, 20% False Report Sensor

In the case of a winding path, both motion models benefit from the inclusion of the risk benefit. The Traffic Motion Model experienced a higher percentage of false starts, but was able to produce better tracking results in all cases than the Dispersion Model.

In the presence of a sensor with a false report rate, the inclusion of the risk assessment presented above helped to improve tracking performance by preventing resampling after a possibly false measurement is received. As the false report rate increased, the risk

Table 4.18: Time to Detect and False Start Results, Winding Path, 20% False Report Sensor

Motion Model	**Risk Assessment**	**Sensor Model**	**Time to Detect**	**Percent False Starts**
Traffic	No	10% False Report	19.68 sec	67.50
Traffic	Yes	10% False Report	65.08 sec	61.00
Traffic	No	Perfect	32.20 sec	0
Dispersion	No	10% False Report	17.29 sec	55.00
Dispersion	Yes	10% False Report	35.19 sec	40.50
Dispersion	No	Perfect	67.70 sec	0

assessment's benefit to tracking performance decreased. A false report is the result of three possibilities: two false detections reported consecutively, a false detection followed by a true detection, or a true detection followed by a false detection. The first case is most detrimental, but is statistically less probable (1% chance in the case of a 10% false report sensor). The second and third cases are undesirable, but because a true detection is involved, the target is within the vicinity of the measured region and was likely located just on the edge of the measured region. In that instance, the required number of effective particles and the sensor belief play critical roles in the preservation of the particle spatial resolution outside of the measured region. Further study of filtering out false detections is necessary.

With the inclusion of the risk assessment, much improved tracking performance was obtained in the case of a sensor with a false report rate of 10%, and some improvement was obtained in the case of a 20% false report sensor. Only the tuned values of the number of effective particles and the sensor belief were used in the simulations in this section because in the presence of false measurements, they provided for superior tracking performance.

A longer observation period would give further insight into the convergence rate of the risk assessment method to the perfect sensor results. An additional investigation into the long term tracking performance of each particle motion model. This was done to observe the tracking performance throughout the entire simulation, not just after detection. This

was motivated by the order of magnitude increase in the tracking error resulting from the increasing the sensor's false report rate by 10% increments.

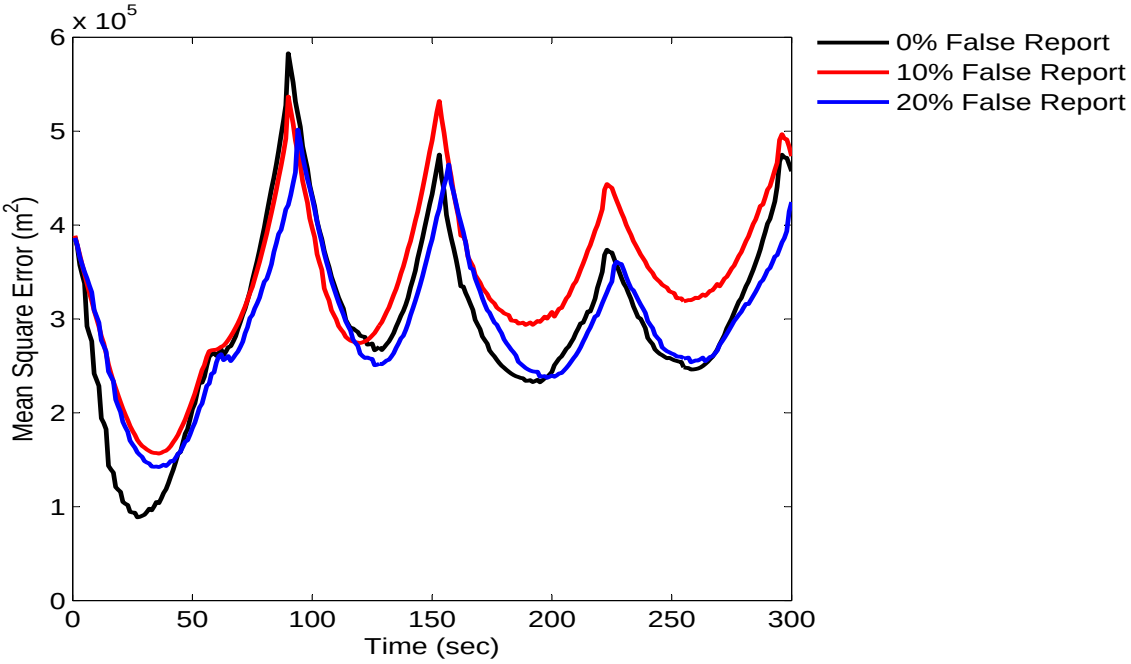


Figure 4.33: Dispersion Model Tracking Performance over 5 minutes, Straight Path

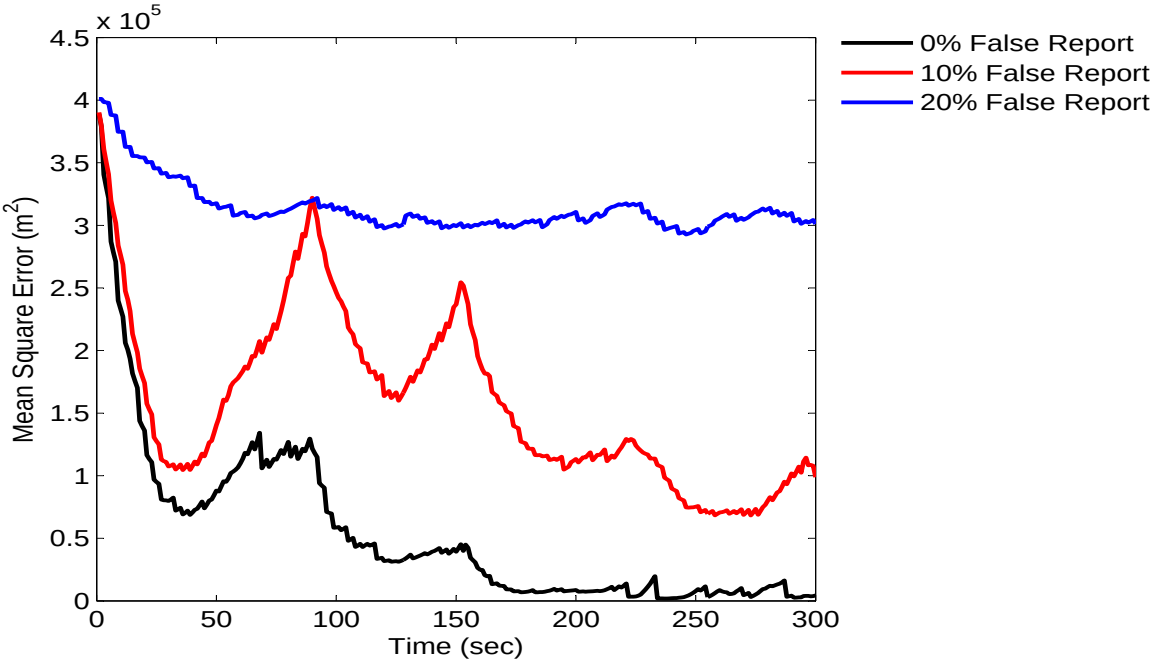


Figure 4.34: Traffic Motion Model Tracking Performance over 5 minutes, Straight Path

Table 4.19: Time to Actually Detect, Straight Path

Motion Model	**Sensor Model**	**Time to Detect**
Traffic	Perfect	50.10 sec
Traffic	10% False Report	79.65 sec
Traffic	20% False Report	101.97 sec
Dispersion	Perfect	48.59 sec
Dispersion	10% False Report	64.99 sec
Dispersion	20% False Report	75.78 sec

It is observed that regardless of the sensor model, the Dispersion Model yields a cyclical fluctuation in tracking performance for a target on a straight path, with peaks of decreasing amplitude as time progresses. The Traffic Motion Model's tracking performance is, as expected, diminished by increased sensor inaccuracy. The 10% false report rate sensor model yields tracking performance that decreases in error as time progresses toward the error values from the perfect sensor. The 20% false report rate sensor error values decrease much more

slowly than the results from more accurate sensor models. The peaks and valleys in the error values occur approximately every 20 to 30 seconds. That time period is approximately how long it would take the target vehicle to traverse through a sensor region. This behavior of the error value could relate to the effect of entering a new sensor region and potentially losing track on the target or regaining track on the target. For both motion models, the time to truly detect a target increases as the frequency of false measurements increases.

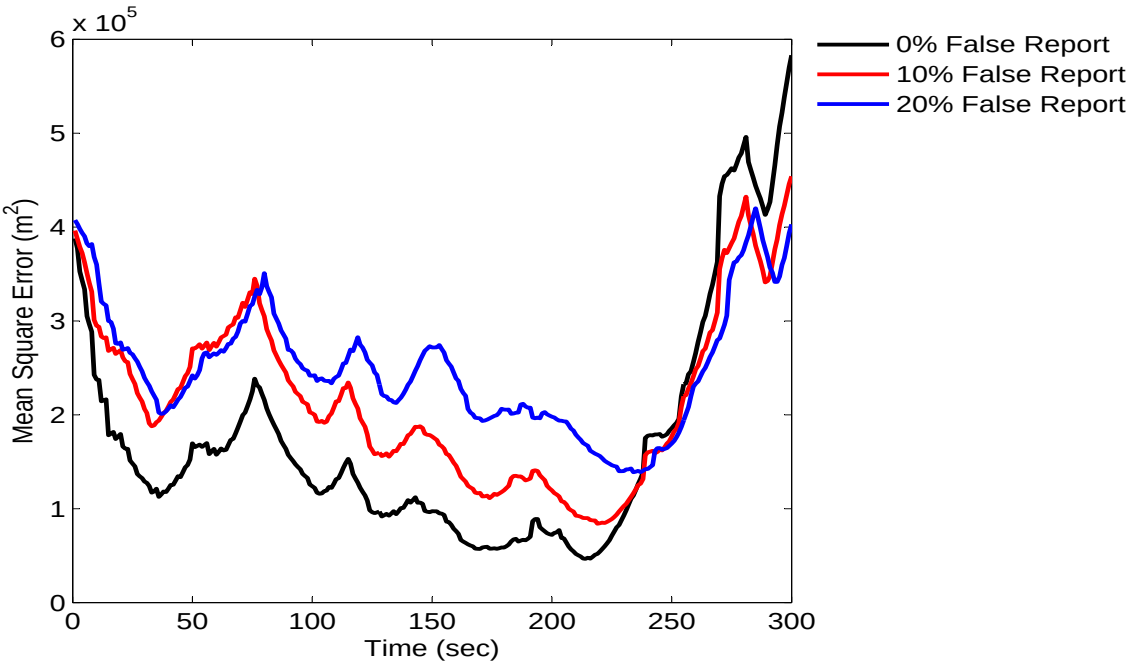


Figure 4.35: Dispersion Model Tracking Performance over 5 minutes, Winding Path

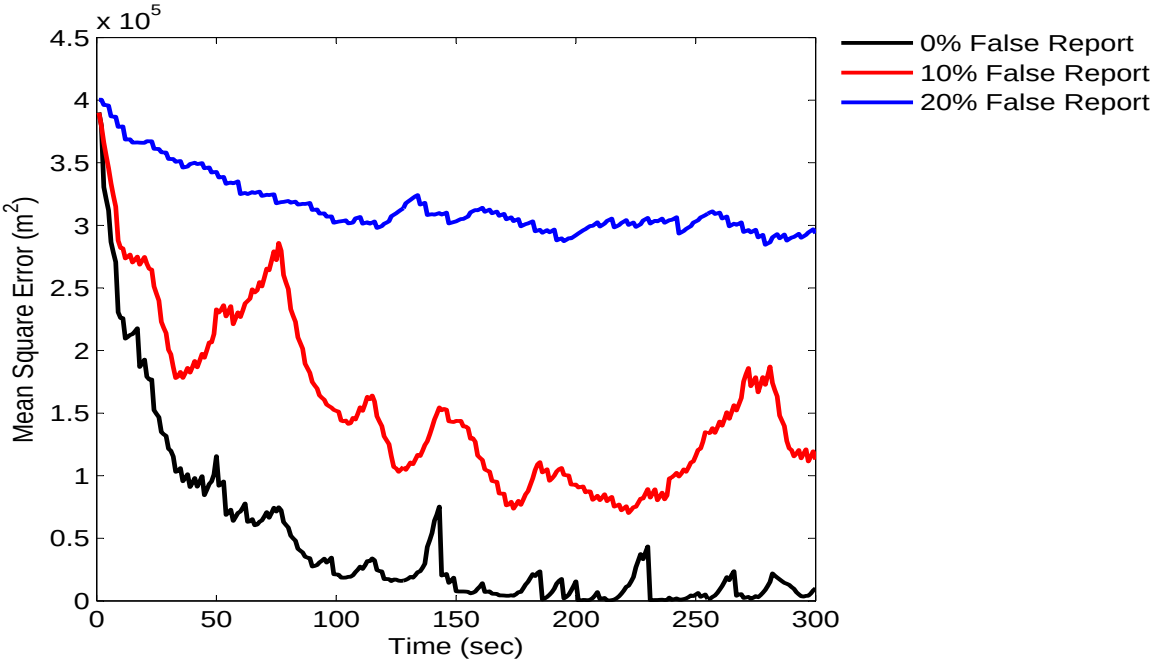


Figure 4.36: Traffic Motion Model Tracking Performance over 5 minutes, Winding Path

Table 4.20: Time to Actually Detect, Winding Path

Motion Model	**Sensor Model**	**Time to Detect**
Traffic	Perfect	36.57 sec
Traffic	10% False Report	66.78 sec
Traffic	20% False Report	116.70 sec
Dispersion	Perfect	61.50 sec
Dispersion	10% False Report	82.29 sec
Dispersion	20% False Report	80.88 sec

Similar to the case of a straight path, tracking performance is diminished for both motion models as sensor accuracy decreases. The tracking performance for the Dispersion Model is less cyclical in the case of a winding path, as the Dispersion Model is more suited to track a target on a winding path. However, the tracking performance diverges for each sensor model at approximately 225 seconds after the start of the simulation. The Traffic Motion Model exhibits similar behavior in the case of both target paths, and outperforms

the Dispersion Model in all cases. Again, the time to truly detect the target in the presence of false measurements increases as the sensor's accuracy decreases. The true times to detect give insight into why the false alarm rates in Sections 4.1 and 4.2 were so high.

The steady decrease in tracking error yielded by the Traffic Motion Model in the case of each sensor model motivated its use in the next phase of the urban tracking problem. A UAV is to intercept the ground vehicle that was being tracked in the aforementioned studies. The UAV will utilize the information provided by the particle filter to plan its path to intercept the target. The same sensor models will be used to study the effectiveness of the Traffic Motion Model's tracking performance in a target intercept problem.

Chapter 5

Target Intercept

A constant velocity UAV is to be introduced into the urban scenario. The mission of the UAV is to plan its path and intercept the target based only on the information provided to it by the particle filter. In this work, the mission will be considered complete when the UAV is within 100 m of and approaching the target, regardless of what road the UAV and target are on.

The dynamic motion model for the UAV is based on the vector field following method found in [34], and is discussed in Section 5.1. Initially, the UAV's path will depend on only measurements provided by the soft binary regional sensor. This reinforces the importance of the particle filter's tracking performance. The effect of adding a measurement source to the UAV will also be investigated. The measurement model that will be added to the UAV is discussed in Section 5.2.

The UAV is to plan its path to traverse above the roadways and must remain between the obstacles. The receding horizon control method for calculating the UAV's path is discussed in Section 5.4. The method for path selection developed in this work is compared to a previously implemented minimum entropy method to show similar performance and computational cost reduction.

5.1 UAV Path Following

A constant speed dynamic model for the UAV was desired for this work. The vector field following method described herein was chosen because it was developed for constant speed MAV's in the presence of unknown wind [34]. It has been experimentally validated in a similar urban environment. In addition, it is assumed that the UAV is aware of its

ground track position. Although the time step for the particle filter is 1.0 sec, a more refined integration routine was necessary for the UAV. A fourth order Runge-Kutta integration with a time step of 0.1 sec was used to integrate the UAV equations of motion. The state vector below is used to describe the UAV, where V is the speed and θ is the heading angle. The speed is held constant while on a road and during a turn.

$$\dot{x} = V \sin\theta \tag{5.1}$$

$$\dot{y} = V \cos\theta \tag{5.2}$$

It is assumed that the UAV is equipped with an autopilot that implements a course-hold loop and that the resulting dynamics are represented by the first order system:

$$\dot{\theta} = \alpha(\theta^c - \theta) \tag{5.3}$$

where θ^c is the commanded heading angle and α is a known positive constant that characterizes the response speed of the autopilot loop.

Figure 5.1 illustrates the vector field leading to a straight path with path heading angle of $\theta_{path} = 0$, as angles are measured clockwise from the positive y-axis. The time update equations to follow this path angle will be discussed below. Quadrant adjustments to the given equations are necessary for the three remaining path heading angles involved in the map in this study.

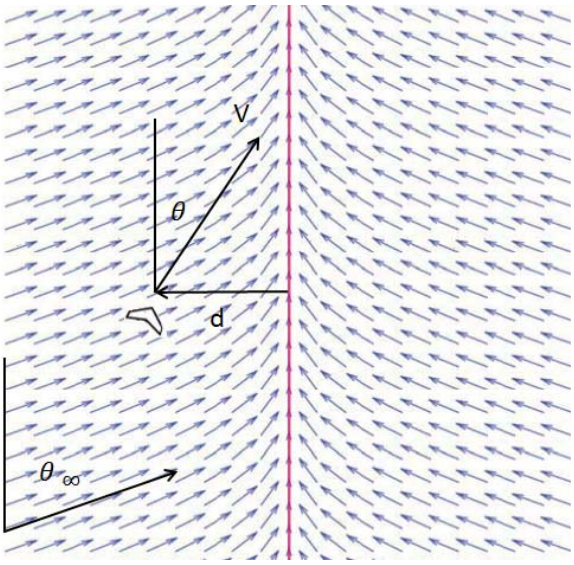


Figure 5.1: Vector field for straight-line following

The lateral difference between the UAV and the desired path is denoted d. The resulting vector field approach will direct the UAV to approach the path at angle θ_∞ when d is very large. As d approaches zero, the heading angle will approach the path angle. The desired heading angle, θ_d is computed as a function of the distance d by the following expression:

$$\theta_d(d) = -\theta_\infty \frac{2}{\pi} \tan^{-1}(kd) + \theta_{path} \tag{5.4}$$

Figure 5.2 illustrates the effect of the value of k on the difference in the sharpness of the approach to the path heading. Large values of k result in short, abrupt transitions, whereas small values of k result in long, smooth transitions in the desired heading angle.

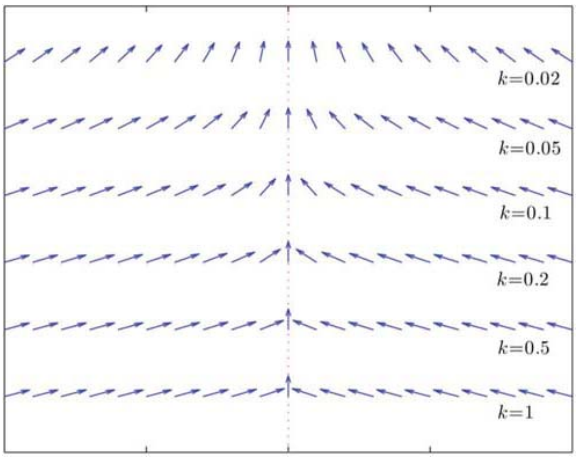


Figure 5.2: Vector fields for various values of k

By restricting θ_∞ to the range $\theta_\infty \in (0, \pi/2)$, it can be shown by using the Lyapunov function $W_1(d) = (1/2)d^2$ that $y \to 0$ asymptotically if $\theta = \theta_d(d)$ [34].

A sliding mode approach is used in [34] to ensure the set $\mathcal{S} = (d, \theta) : \theta = \theta_d(d)$ is positively invariant and the system trajectory reaches $\mathcal{S}$ in finite time. Defining $\tilde{\theta}_d \equiv \theta - \theta_d(d)$ and differentiating yields:

$$\dot{\tilde{\theta}}_d = \dot{\theta} - \dot{\theta}_d(d) \tag{5.5}$$

$$= \alpha(\theta^c - \theta) + \theta_\infty \frac{k}{1 + (ky)^2)} V \sin\theta \tag{5.6}$$

$$\tag{5.7}$$

By defining a second Lyapunov function, $W_2 = (1/2)\tilde{\theta}^2$ and differentiating:

$$\dot{W}_2 = \tilde{\theta}\dot{\tilde{\theta}} \tag{5.8}$$

$$= \tilde{\theta}(\dot{\theta} - \dot{\theta}_d(d)) \tag{5.9}$$

$$= \tilde{\theta}\left(\alpha(\theta^c - \theta) + \theta_\infty \frac{k}{1 + (ky)^2)} V \sin\theta\right) \tag{5.10}$$

If the control signal is chosen as:

$$\theta^c = \theta - \frac{1}{\alpha}\theta_\infty \frac{2}{\pi}\frac{k}{1+(ky)^2)}V\sin\theta - \frac{\kappa}{\alpha}\text{sign}\left(\theta - \theta_d\right) \tag{5.11}$$

where

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{5.12}$$

and the gain parameter $\kappa > 0$ and controls the shape of the trajectories onto the sliding surface. Large values of κ drive $\tilde{\theta}$ to zero quickly.

This results in $\dot{W}_2 \leq -\kappa|\tilde{\theta}|$, which yields that $\tilde{\theta} \to 0$ in finite time. To avoid the chattering associated with the sign function, the control signal is chosen as:

$$\theta^c = \theta - \frac{1}{\alpha}\theta_\infty \frac{2}{\pi}\frac{k}{1+(ky)^2)}V\sin\theta - \frac{\kappa}{\alpha}\text{sat}\left(\frac{\theta - \theta_d}{\epsilon}\right) \tag{5.13}$$

where ϵ defines the width of the boundary region around the sliding surface in radians and:

$$\text{sat}(x) = \begin{cases} x, & \text{if } |x| \leq 1 \\ \text{sign}(x), & \text{otherwise} \end{cases} \tag{5.14}$$

The values for the constants used throughout the path following routine listed in Table 5.1 were used in experimental tests in [34].

Table 5.1: Path Following Constants

Constant	**Value**
V	20 m/s
α	1.65 rad/sec^2
k	0.02/m
θ_∞	$\pi/2$ rad/sec
κ	$\pi/2$ rad^2/sec
ϵ	1.0 rad

Figure 5.3 illustrates a sample path that was implemented into the urban scenario used throughout this work. The UAV's starting point is the green square and stopping point is the red square. The UAV successfully traversed down the center of roadways, as it was commanded, and was able to complete smooth turns, and then re-center itself on a new roadway. The minimum turning radius encountered on this path was 20 m.

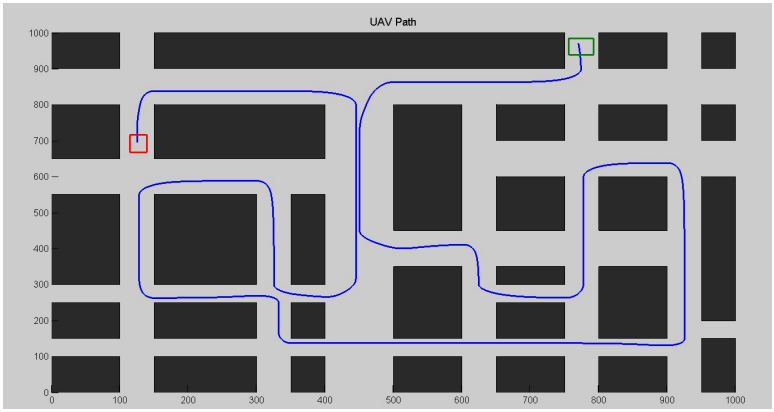


Figure 5.3: UAV Path Following via Vector Fields Example

5.2 UAV Measurement Update

The sensor on the UAV is modeled after the perfect binary sensor discussed in previous chapters. It is assumed that the UAV will take one measurement per second and that the field of regard spans the width of the road the UAV is traversing for one second, creating the sensor footprint illustrated below.

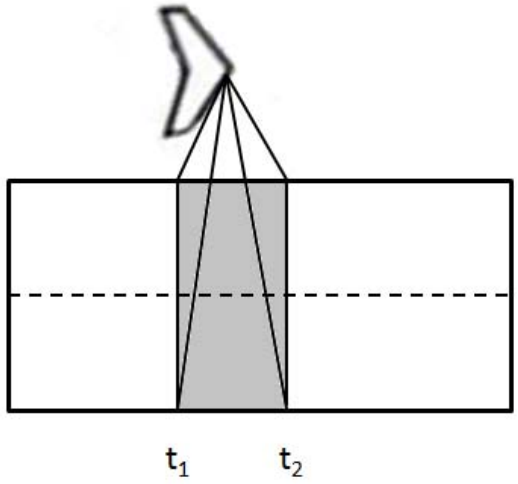


Figure 5.4: UAV Sensor Requirements

The UAV is commanded to travel down the center of the road at a constant altitude of 30 m. The maximum width of a road in the road network is 100 m, resulting in a maximum required pan angle of approximately 60° to the right and to the left of the center of the UAV. This is illustrated by Fig. 5.5, where the dark arrow in the center of the road indicates the direction of travel.

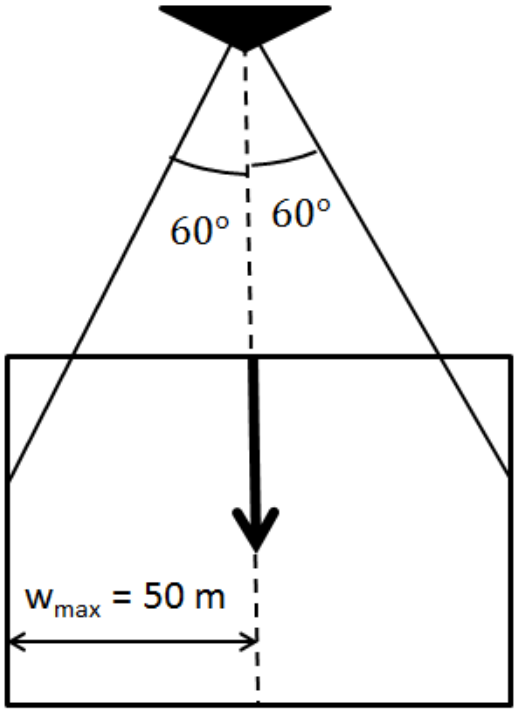


Figure 5.5: UAV sensor span while traveling down-road

It is assumed that the UAV travels faster than the target vehicle, therefore a constant speed of 20 m/s was chosen. In addition, to reinforce the importance of path planning, the UAV was constrained to the road network by a forced constant altitude of 30 m, which is less than the height of the obstacles. A constant commanded speed of 20 m/s, altitude of 30 m, and a measurement frequency of 1 Hz results in a maximum tilt angle of approximately 20°. Figure 5.6 illustrates the 20° angle required to cover the required ground.

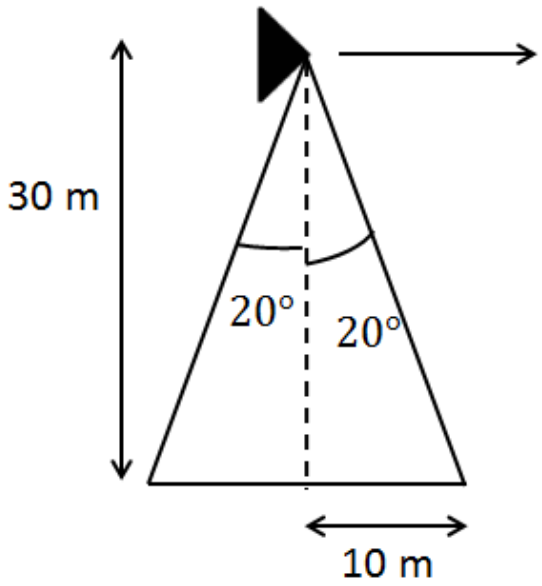


Figure 5.6: UAV sensor tilt while traveling down-road

A constant commanded altitude of 30 m results in a maximum sight distance requirement of 62 m for the UAV sensor. The UAV sensor will be assumed to have a 0% false report rate. The particle cloud will not be resampled after each UAV measurement to conserve computational expense as the UAV sensor footprint observes a relatively small subset of particles each second. This small number of particles will contribute to but not greatly affect the number of effective particles. This results in a sensor footprint that covers the width of the road and the length of the road traversed by the UAV each second.

5.3 Candidate Path Formation

As the UAV is required to remain within the road network of the urban environment used in this study, it must make a decision when it reaches an intersection as to which way to proceed. The UAV's path may contain up to three segments that must consist of one road between two intersections; the UAV may not traverse over an obstacle. This problem is similar to a directed graph where each intersection is a node and each road is an edge. The

direction of each edge is based on the UAV's heading at the time the path is planned. The UAV may only travel forwards and is forced to stay on the map, requiring a U-turn at the map edge. The node network used in this work is illustrated by Fig. 5.7.

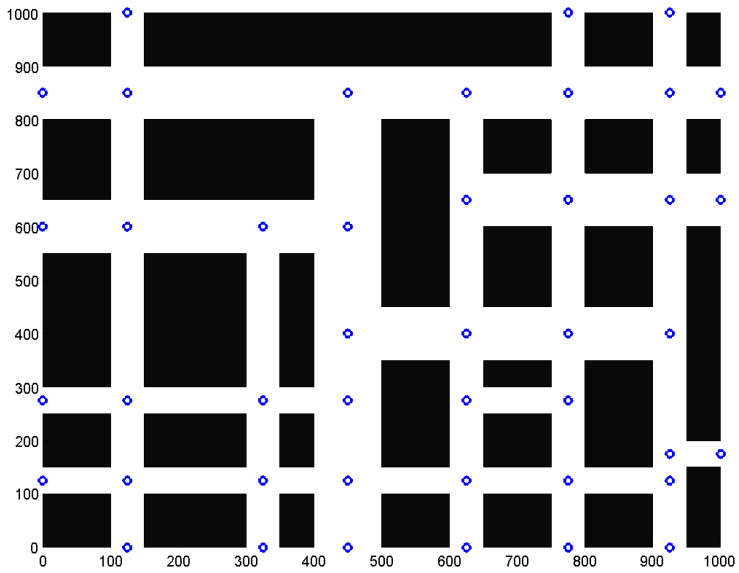


Figure 5.7: Node Network

Only a subset of the total set of nodes is used to plan the UAV's path as it approaches an intersection. The subset of nodes are chosen based on their proximity to and relative direction from the UAV at the time of path planning. This is done to limit the complexity of the path planning problem, as the UAV replans its path at each new intersection it encounters. Planning paths to nodes far beyond UAV's position would be a waste of computational expense, therefore a planning horizon is defined. The path planning routine collects all permissable paths within a specified semi-circular region, known as the planning horizon, in the direction of the UAV's current heading. The radius for the planning horizon used in this work is 350 m, and it is measured from the root node. The nearest intersection ahead of the UAV's current path at the time of planning is defined as the root node.

The aforementioned path planning parameters are illustrated by Fig. 5.8. The UAV, illustrated by the green circle, is located at the point [550,400], with a heading angle of

$3\pi/2$ as indicated by the arrow, and is approaching an intersection located at [450,400]. In Fig. 5.8, the red circle is the target, the black circle is the root node. The blue semi-circle illustrates the UAV's planning horizon with a radius of 350 m. The seven nodes within the planning horizon are denoted with an X. These nodes would be used to create a list of candidate paths, from which the best path for the UAV will be selected. The UAV then follows the selected path until it approaches another intersection.

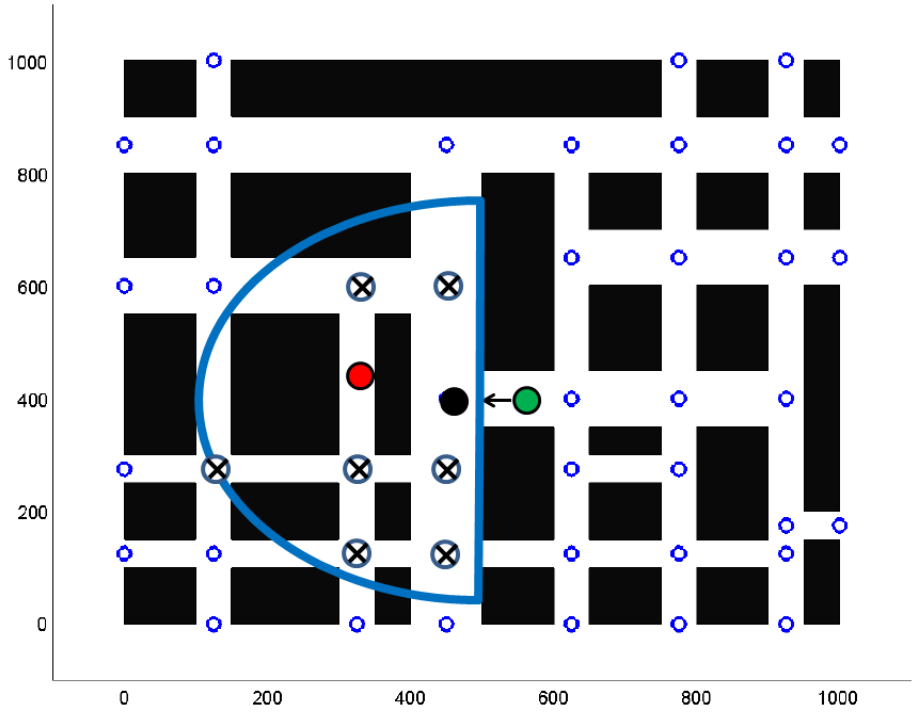


Figure 5.8: UAV planning horizon

The remaining nodes outside of the planning horizon, denoted by blue circles, are included in Fig. 5.8 for completeness. The nodes outside of the planning horizon are not included in the formation of the candidate paths. The collection of intersections and roads within the planning horizon are used to form a directed graph, where the intersections are nodes and the roads are the edges.

It is assumed that the UAV has complete knowledge of the map within its planning horizon, and is therefore aware of the location of each node and dimension and location

of each road within the horizon. Paths must begin at the root node and may consist of a maximum of three node points (intersections) beyond the root node to nodes within the planning horizon. The roads eligible for path creation, given the planning horizon and root node location in Fig. 5.8, are highlighted in yellow in Fig. 5.9. In the scenario given in Fig. 5.9, there are 10 possible paths to consider, as paths may consist of up to three roads and multiple routes to a node are permitted. An effective means to collect paths between nodes in a directed grid-like map is outlined below.

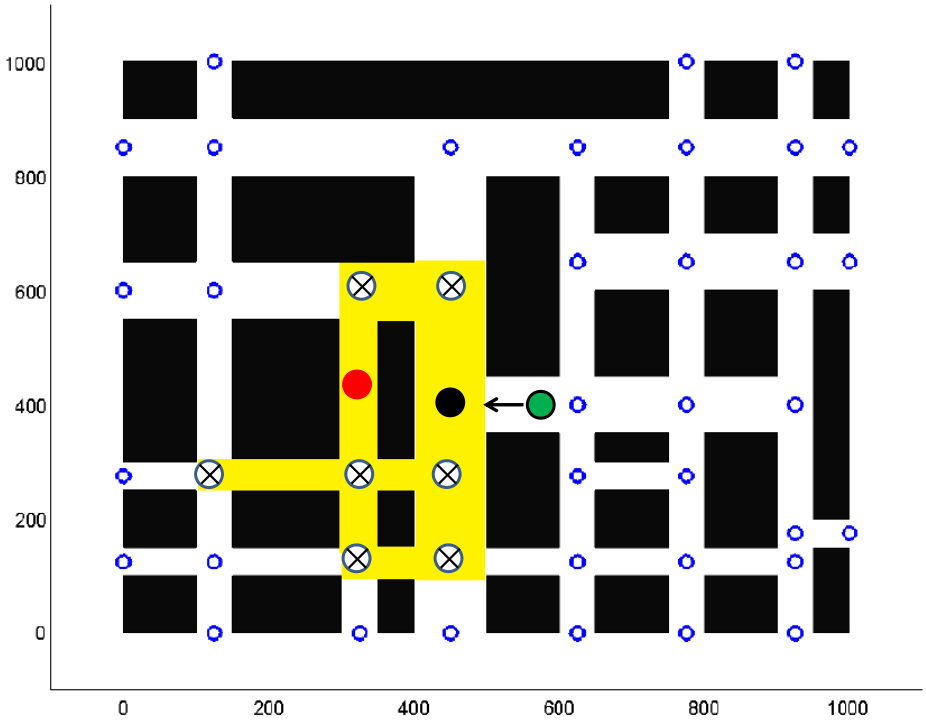


Figure 5.9: Node Map

An algorithm was developed to build a directed graph to represent all candidate paths from the root consisting of a maximum of three roads. This framework was motivated by the use of a directed graph to represent map features such as intersections and curved roadways [29]. In general, the the path formation process begins at the root node and searches the four cardinal directions ($\theta \in [0, \pi/2, \pi, 3\pi/2]$) in order of increasing magnitude for nodes within the planning horizon. This is done by first ranking the set of nodes in a

given direction within the planning horizon in order of increasing distance from the root. When the nearest node in one direction is found, the path to that node is stored and the four cardinal directions are searched from that node for additional nodes. This process continues until all paths within the planning horizon consisting of a maximum of three roads have been collected.

First, the direction $\theta = 0$ is searched from the root for a node within the planning horizon. If a node is found in that direction, the following quantities are stored to the characterize the path from the root to the node: which road is between the root and the node, the distance between the root and the node, the direction of travel from the root to the node, and which node was found. From the node that was found, the direction $\theta = 0$ is searched for a node within the planning horizon. If a node is found in the search direction, the path to that node is stored, and a third node beyond the root is searched for in the direction $\theta = 0$.

If at any point during this process a node is not found in the search direction, the search direction proceeds in order of increasing values of θ until a node is found. If no node is found in any direction, the routine returns to the node it searched to get to the current node, and searches the remaining cardinal directions. When each direction has been searched from the root node, and consequently all nodes within the planning horizon have been searched for connections in each direction, all of the candidate paths have been collected. Each path is characterized by which roads and intersections it involves, the distance between nodes, and the direction each road would be traversed. The example below will be used to describe this process in more detail.

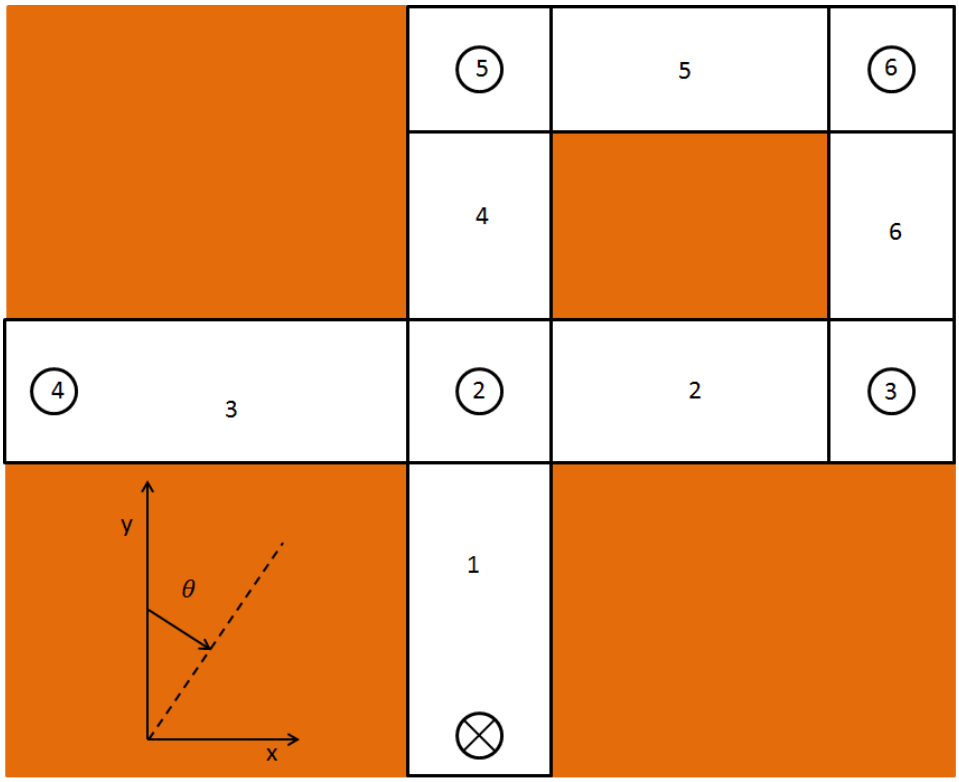


Figure 5.10: Example: Nodes and Roads for Candidate Paths

Consider the road network in Fig. 5.10, where roads and intersections are drawn in white, obstacles in orange, node ID numbers are circled and road ID numbers are not circled. The search angle θ is measured clockwise from the positive y-axis. In this example, the node network in Fig. 5.10 is made up of only the nodes within the planning horizon. Node 4 is the only node located at the edge of the map, and therefore a U-turn would be required to depart from Node 4. The root node is denoted with an X. The intersections and roads within the planning horizon have been collected and will be used to determined the candidate paths.

In this example, it is assumed that the UAV velocity is constant and each road is of equal dimension, therefore the distance between nodes connected by only one road is equal. This results in an equal travel time (T) for the UAV between any two nodes connected by only one road. Beginning at the root node, the nearest node is searched for in in the direction of $\theta = 0$. Node 2 is found, and the path from the root to Node 2 is stored as detailed by

Table 5.2, and the first edge of the directed graph is formed. Figure 5.11 illustrates the root node as Node 1, the road ID number as the number 1 next to the edge connected the two nodes, the direction of travel indicated by the arrow, and the end node ID as Node 2.

Table 5.2: Path to Node 2

Path to Node	**Route ID**	**Edge Number**	**Road ID**	**Travel Time**	**Direction of Travel**	**End Node ID**
2	1	1	1	T	0	2

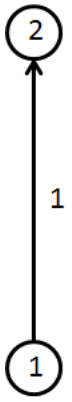


Figure 5.11: Edge of Directed Graph from Root Node to Node 2

After storing the first path to Node 2, a node in the planning horizon is searched for from Node 2 in the direction of $\theta = 0$. Node 5 is found and a path to Node 5 is stored, consisting of the path from the root to Node 2 and the recently found road between Nodes 2 and 5, as in Table 5.3. Another edge is added to the directed graph, as illustrated by Fig. 5.12.

Table 5.3: Path to Node 5

Path to Node	Route ID	Edge Number	Road ID	Travel Time	Direction of Travel	End Node ID
5	1	1	1	T	0	2
		2	4	T	0	5

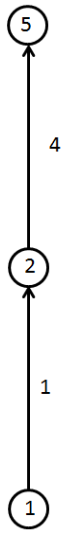


Figure 5.12: Edges of Directed Graph from Root Node to Node 5

From Node 5, the direction $\theta = 0$ is searched for a node. Because there is no node in that direction, the next direction ($\pi/2$) is searched, and Node 6 is found. The path to Node 6 is stored consisting of the three nodes from the root: nodes 2, 5, and 6. Because the path to Node 6 consists of the maximum number of roads (3), no further nodes will be searched for from Node 6. The directed graph is updated to include the recently found path to Node 6.

Table 5.4: Path to Node 6

Path to Node	Route ID	Edge Number	Road ID	Travel Time	Direction of Travel	End Node ID
6	1	1	1	T	0	2
		2	4	T	0	5
		3	5	T	$\pi/2$	6

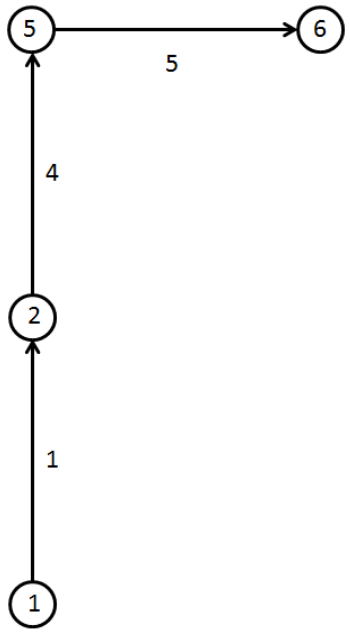


Figure 5.13: Edges of Directed Graph from Root Node to Node 6

As no more nodes may be searched for from Node 6, the routine will return to the node it found immediately prior to Node 6, which is Node 5. The next direction to search from Node 5 is $\theta = \pi$. That would require a U-turn, which is not permitted when not at the map's edge. The routine then searches the final direction from Node 5, $\theta = 3\pi/2$, and there are no nodes in that direction. Therefore, the routine returns to the node it found prior to Node 5, which is Node 2. As the direction $\theta = 0$ has already been searched from Node 2, the routine searches the next direction for Node 2, $\theta = \pi/2$. Node 3 is found, and the path to Node 3 is stored in the path list and the directed graph is updated. Next, the direction

of $\theta = 0$ is searched from Node 3 and Node 6 is found, resulting in a second route to Node 6. The path list is updated accordingly.

Table 5.5: Path to Node 3

Path to Node	**Route ID**	**Edge Number**	**Road ID**	**Travel Time**	**Direction of Travel**	**End Node ID**
3	1	1	1	T	0	2
		2	2	T	$\pi/2$	3

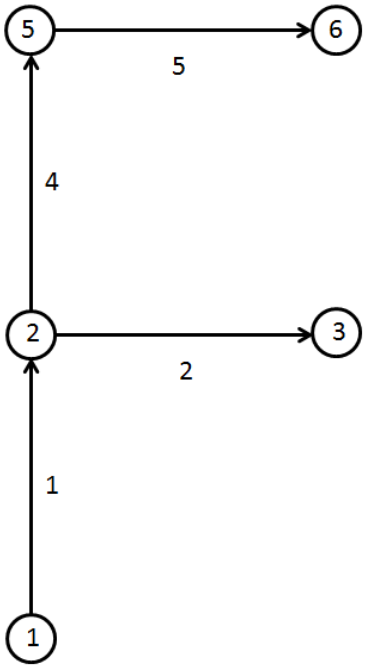


Figure 5.14: Directed Graph after finding Node 3

Table 5.6: Paths to Node 6

Path to Node	Route ID	Edge Number	Road ID	Travel Time	Direction of Travel	End Node ID
6	1	1	1	T	0	2
		2	4	T	0	5
		3	5	T	$\pi/2$	6
6	2	1	1	T	0	2
		2	2	T	$\pi/2$	3
		3	6	T	0	6

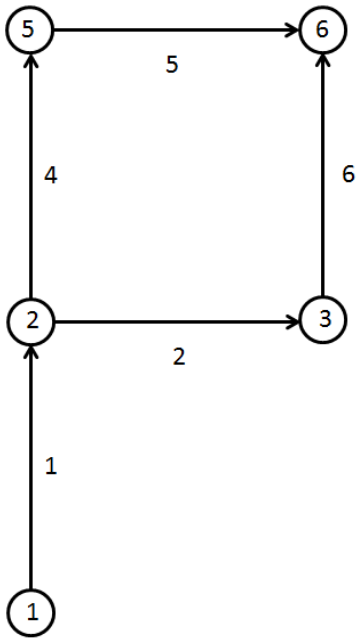


Figure 5.15: Directed Graph after finding a second route to Node 6

Again, no additional nodes are searched for from Node 6 because the paths to Node 6 consist of the maximum number of edges. Therefore, the routine returns to the previous node, Node 3. As there are no nodes in the directions $\pi/2$ or π from Node 3, and Node 2 would require an illegal U-turn, the routine retraces its path to Node 3 to Node 2. Searching for a node in the direction of π from Node 2 would require an illegal U-turn, therefore the

next direction is searched. The direction $3\pi/2$ is searched from Node 2 and Node 4 is found and the path to Node 4 is stored and the directed graph is updated.

Table 5.7: Path List

Path to Node	**Route ID**	**Edge Number**	**Road ID**	**Travel Time**	**Direction of Travel**	**End Node ID**
4	1	1	1	T	0	2
		2	3	T	$3\pi/2$	4

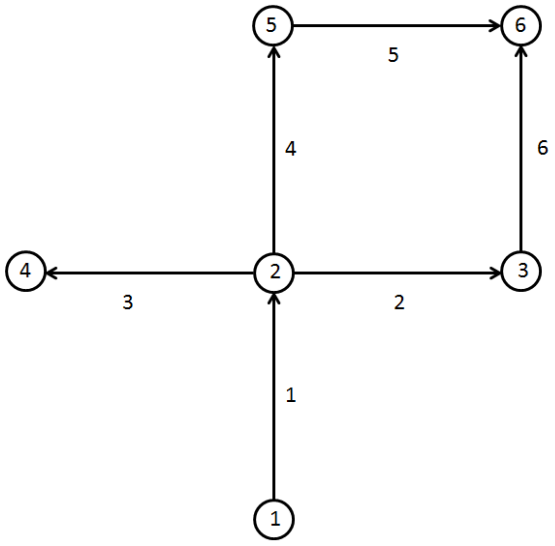


Figure 5.16: Directed Graph after finding Node 4

Node 4 is at the edge of the map, therefore a U-turn will be permitted, resulting in another path to Node 2. As all nodes within the planning horizon have been searched in each direction, the candidate path list is complete and is given in Table 5.8. There are a total of seven candidate paths in the final directed graph illustrated by Fig 5.17.

Table 5.8: Final Path List

Path to Node	Route ID	Edge Number	Road ID	Travel Time	Direction of Travel	End Node ID
2	1	1	1	T	0	2
2	2	1	1	T	0	2
		2	3	T	$3\pi/2$	5
		3	3	T	$\pi/2$	6
3	1	1	1	T	0	2
		2	2	T	$\pi/2$	3
4	1	1	1	T	0	2
		2	3	T	$3\pi/2$	4
5	1	1	1	T	0	2
		2	4	T	0	5
6	1	1	1	T	0	2
		2	4	T	0	5
		3	5	T	$\pi/2$	6
6	2	1	1	T	0	2
		2	2	T	$\pi/2$	3
		3	6	T	0	6

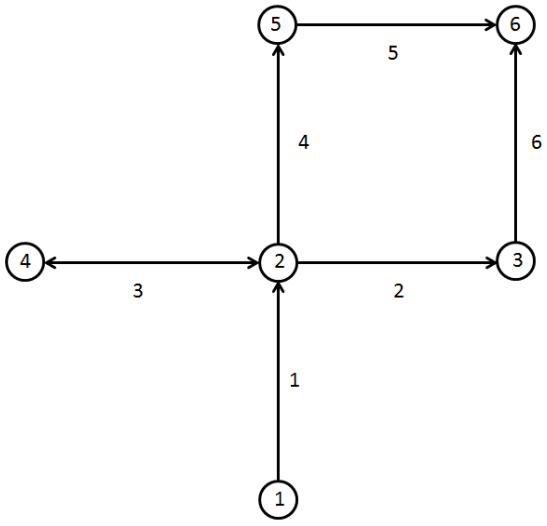


Figure 5.17: Directed Graph of Example Path Planning Problem

Following the collection of the candidate paths, a means of selecting the path with the most benefit was developed. Section 5.4 outlines the maximum likelihood method for

selecting the best candidate path that was developed in this work, and compares that method to an entropy reduction method presented in previous work.

5.4 Path Selection

Once the candidate paths have been collected, the best path for the UAV must be selected. It is desirable to choose a path that will enable the UAV to intercept the target as quickly as possible based only on the information provided by the particle filter. Numerous metrics exist for path selection such as minimal mean square error, maximal Information gain, minimal Shannon entropy, and maximal likelihood of detection. In Chapters 3 and 4, the large regional sensor measured the region of highest particle weight. The same metric will be used to select the best candidate path. In this work, it will be shown that selecting the path that holds the highest particle weight at the time the UAV will traverse it provides for effective target intercept. In addition, comparable performance to a entropy reduction method is achieved with a significant reduction in computational time. The time to intercept the target and the amount of time to simulate one second will be used as metrics to compare the two methods presented herein.

In the path planning comparison below, a path may consist of a maximum of three roads. Both methods will make their path selections based on similar cost functions in an effort to maximize the amount of information gained, minimize the distance between the UAV and the expected value of the particle cloud, and minimize the travel time of the UAV.

$$J_{path} = \sum_{i=1}^{N_{edges}} \frac{1}{\text{Path Value}_i} + d_{i=N_{edges}} + t_{path,i} \tag{5.15}$$

The second two terms in the summation above will be the same for both methods. In the maximum likelihood method, the first term will involve the sum of the particle weights to be encountered on a path. In the minimum entropy method, the first term will involve a mutual information utility function.

Computational time is taken into account when the UAV approaches an intersection to ensure enough time is available to select the next path and have a decision made when the UAV arrives at an intersection. Although path are planned to have a length of up to three roads, corresponding to upwards of 20 seconds in the future, paths are be replanned at each intersection to account for additional measurement data from outside sources, as consistent with receding horizon control. Therefore, four seconds before reaching the next intersection, a new path is planned.

5.4.1 Maximum Likelihood

In previous work, path selection was based on maximizing the reduction in the entropy of the *pdf* [24], [26], [28]. In this work, path benefit is based on the weight of the particles along a path at the appropriate time, the distance between the final node on a path and the expected value of the particle cloud, and the amount of travel time required by each path.

$$J_{path} = \sum_{i=1}^{N_{edges}} \frac{R}{N w_{edge_i}} + \left(\frac{d_i}{r_{plan}}\right)^Q + \frac{t_{path}}{2} \tag{5.16}$$

The cost function to be minimized is defined by Eq. (5.16), where N_{edges} is the number of edges, or roads, in a path, N is total number of particles, d_i is the distance between the final node in the path and the expected value of the particle cloud, r_{plan} is the planning radius for the UAV, and t_{path} is the amount of time it would take the UAV to complete the path. The parameters R and Q may be used to adjust the impact of the weight term and the distance term on the cost of a path. In this work, R is set to unity and Q is 2. A path is chosen through a simple search for a minimum cost using the *min* function in MATLAB.

The particle weight of a road was calculated by propagating the particle cloud forward in time without measurements from outside sources (human operatives). The weight of a particles on a road is measured at two times: at the midpoint of the road and at the end node. At each of those times, the region of the road the UAV has just passed is measured for particle presence. The time of interest to which the particle cloud is propagated is determined by

the speed of the UAV and the length of the road, which is stored in the candidate path list. Figure 5.18 illustrates the two regions of two roads on a candidate path. The blue regions would be measured at the midway point of the roads, times 5 and 15, respectively. The orange regions highlight the second half of the road and the end node, which are measured for particle weight at times 10 and 20. The road the a particle is on is stored in its state vector. This reduces computational time by avoiding searching each road for N particles at each time of interest.

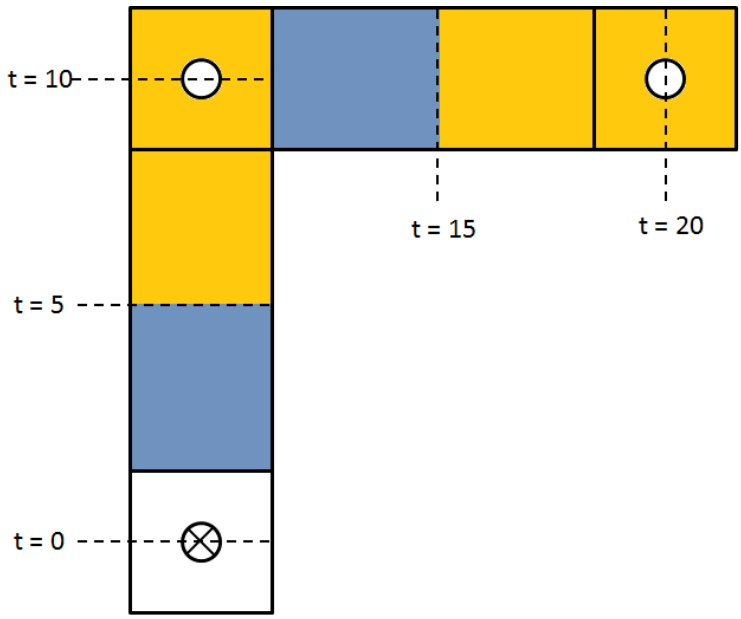


Figure 5.18: Two Part Road Weight

If there are no particles on a road, the road weight is set to $1/N$ to avoid dividing by zero in Eq. (5.16). The particle weight on the road of interest is then calculated to determine target likelihood at the time when the mobile sensor will be on that road. This is a computationally expensive process that is also necessary in minimum entropy methods.

5.4.2 Information Utility Function

A minimum entropy path selection routine as described in [25] was implemented to provide for comparison to the maximum likelihood approach described above. As mentioned

in Section 2.3, the Information Utility Function (IUF) is computed by:

$$\mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right) = \mathbf{H}\left(\mathbf{z}_k\right) - \mathbf{H}\left(p\left(\mathbf{z}_k|\mathbf{x}_k\right)\right)$$

The IUF is based on the idea that minimizing the posterior uncertainty is equivalent to maximizing the difference between the uncertainty that any particular observation will be made $\mathbf{H}\left(\mathbf{z}_k\right)$, and the uncertainty of the measurement model, $\mathbf{H}\left(p\left(\mathbf{z}_k|\mathbf{x}_k\right)\right)$. The entropy terms are approximated by the following expressions:

$$\mathbf{H}\left(\mathbf{z}_k\right) \approx -\int \left\{ \sum_{i=1}^{N} \left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right) \log_2 \sum_{i=1}^{N} \left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right) \right\} d\mathbf{z}$$

$$\mathbf{H}\left(\mathbf{z}_k|x_k\right) \approx -\int \sum_{i=1}^{N} \left[w_k^i p\left(\mathbf{z}_k|x_k^i\right) \log_2 p\left(\mathbf{z}_k|x_k^i\right)\right] d\mathbf{z}$$

Because in this work, only one measurement is received per time step, the integral falls away, reducing the IUF to:

$$\mathbf{I}\left(\mathbf{z}_k;\mathbf{x}_k\right) \approx -\sum_{i=1}^{N} \left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right) \log_2 \sum_{i=1}^{N} \left(w_k^i p\left(\mathbf{z}_k|x_k^i\right)\right) + \sum_{i=1}^{N} \left[w_k^i p\left(\mathbf{z}_k|x_k^i\right) \log_2 p\left(\mathbf{z}_k|x_k^i\right)\right]$$

The weights of the particles $w_k^i, i = 1, ..., N$, are readily available. The measurement likelihood $p\left(z_k|x_k^i\right)$ is computed based on the UAV sensor model and the ith particle's position, as discussed in Section 3.4. A false report belief of 1% was used to avoid taking the logarithm of zero.

5.5 Method Comparison Results

Results for the maximum likelihood (ML) and Information Utility Function (IUF) sensor routing methods are presented below. The target was initialized at a random location and

the UAV was initialized at a constant location to begin each run, and the target traversed a random path throughout each run. For a metric of comparison, an evaluation was done of the expected amount of time it would take the UAV to travel between two random locations in the map (See Appendix A). The expected value of the UAV's travel time from a random point in the environment to a random stationary point is 33.33 seconds. An average time to detect for the perfect regional sensor for a target on a random path starting at a random location was found to be 47.31 seconds. This time was added to the expected value of travel time between two points to be used as a metric of comparison for target interception time. Therefore, if the target were to be stationary, the expected intercept time should be approximately 80 seconds.

The Traffic Motion Model was used to propagate the particles in time and 1000 particles were used. The risk assessment was included and the sensor belief and number of effective particles were tuned to the optimized results obtained in Chapter 4. The results in Table 5.9 indicate that the ML and IUF routines yield similar intercept times, while the IUF routine outperforms the ML routine slightly in each case. After observing the similarity in the intercept times, it can be concluded that there is not a significant difference in the paths chosen by the maximum likelihood routine and the IUF routine. This also implies that the ML routine indirectly reduces the entropy of the particle cloud as the IUF routine does, but at a much reduced computational expense. This is a desirable result as the path planning is to be done on board a small UAV, where computational power is limited.

The results in Table 5.9 indicated that in the case of a perfect sensor, the UAV intercepted the target at approximately 81 seconds in both cases, which slightly greater than the expected value of the time to intercept a stationary target of 80 seconds. This shows the importance of path planning and the effectiveness of the chosen methods. Additionally, the detection time in the case of a perfect sensor is less than the average value of 47.31 seconds that was obtained without the presence of a UAV. This is because although the UAV did not

take measurements, it could intercept the target on its own before the target was located by the large regional sensor.

Table 5.9: Time to Intercept and Run Time Results, without UAV measurements

Sensor Model	**Path Selection Routine**	**Time to Intercept**	**% False Alarms**	**Average Detection Time**
Perfect	ML	81.28 sec	0	37.62 sec
Perfect	IUF	80.44 sec	0	32.45 sec
10% False Report	ML	111.35 sec	12.50	45.16 sec
10% False Report	IUF	123.13 sec	15.50	43.44 sec
20% False Report	ML	150.15 sec	40.50	36.67 sec
20% False Report	IUF	147.75 sec	44.50	39.91 sec

When a sensor with a false report rate is utilized, the time to intercept increased for both the ML and IUF routines beyond the 80 second mark to intercept a stationary target. This delayed interception time is due to false reports incorrectly shifting particle weight, despite the efforts of the risk assessment, throughout the path planning process. The ML routine outperforms the IUF method in the case of a 10% false report rate sensor in regard to intercept time. This reinforces the conclusion that the ML routine is an effective means of path planning that requires reduced computational expense when compared to the IUF routine. In addition, the false alarm rates for both imperfect sensor models are much lower than those presented in Section 4.3. This occurs because the UAV is capable of intercepting the target before the large regional sensor.

Table 5.10 lists the results given the use of the UAV as a measurement source. As expected, the inclusion of the measurements from the UAV decreased the time to intercept. The most significant decrease in interception time occurred in the case of the 10% false report rate sensor. In addition, when a perfect regional sensor is used, the average detection time decreased in the case of both ML and IUF routines when compared to the average value of 47.31 seconds achieved without the UAV. The reduction in the time to intercept the target provided by UAV measurements was less pronounced in the case of a perfect sensor and a

Table 5.10: Time to Intercept and Run Time Results, with UAV measurements

Sensor Model	**Path Selection Routine**	**Time to Intercept**	**% False Alarms**	**Average Detection Time**
Perfect	ML	77.65 sec	0	34.21 sec
Perfect	IUF	81.53 sec	0	35.01 sec
10% False Report	ML	105.98 sec	17.26	43.30 sec
10% False Report	IUF	102.52 sec	14.00	39.88 sec
20% False Report	ML	149.90 sec	43.50	36.08 sec
20% False Report	IUF	146.42 sec	43.00	31.50 sec

sensor with a 20% false report rate. This illustrates the strong effect of the large regional sensor on the particle cloud. The regional sensor is able to observe large areas (40,000 m^2) every 3 seconds, and there is complete coverage of the simulation space at all times. The UAV is restricted to observe the area within its vicinity, with maximum coverage of 6,000 m^2 over 3 seconds. To further illustrate this, the simulation was run with only measurements from the UAV and no regional sensor. The particle distribution was resampled every 3 seconds and a measurement was taken by the UAV every second.

Table 5.11: Comparison of ML and IUF routines using UAV measurements without regional sensor data

Routine	**Time to Intercept**
Maximum Likelihood	225.70 sec
Information	201.30 sec

The results in Table 5.11 illustrate the importance of the large regional sensor, and therefore the inclusion of non-traditional measurement sources into the target intercept problem. In addition, the presence of the UAV is vital to the successful localization and tracking in the presence of a regional sensor with a false report rate. The results in Chapter 4 indicate that given the regional sensor alone, poor tracking results are obtained in the presence of false reports. This is often the result of accepting a false report to be true, and therefore possibly

losing the target completely. The inclusion of the UAV and an intelligent path planning routine provides for consistent target localization. Further benefit to intercept time reduction would be obtained with the inclusion of additional UAV's.

The computational expense for each scenario is listed in Table 5.12. The run time is decreased by use of the ML method both with and without UAV measurements. In addition, the use of UAV measurements does not add significant computational expense to either routine.

Table 5.12: Time to simulate one second

Path Selection Routine	**Without UAV Measurements**	**With UAV Measurements**
ML	0.3671 sec	0.3839 sec
IUF	0.4358 sec	0.4518 sec

Finally, the ML and IUF methods were compared by just using the large regional sensor to locate the target. No UAV was simulated and no paths were planned. A comparison was done to determine which region to poll based off of either the sum of the particle weights, which is comparable to the maximum likelihood method, or which region would provide the highest value of the Information Utility function. The work presented in previous chapters polled the sensor in the region with the highest particle weight sum. The Information Utility Function approach was implemented in the regional sensor framework for comparison. The times to detect the target were compared. A perfect sensor was used and only the Traffic Motion Model with 1000 particles was used in this study. The results give in Table 5.13 indi-

Table 5.13: Comparison of ML and IUF routine using only large regional sensor data

Routine	**Time to Detect**
Maximum Likelihood	47.31 sec
Information	49.96 sec

cate that the maximum likelihood routine yields similar results to the maximum information routine, with a significantly decreased computational expense.

The results presented above illustrate that the particle weight to be encountered along a path is a sufficient and computationally efficient metric for path ranking. The particle filter framework provides for this intuitive metric without the need to perform additional calculations. Proper tuning of particle filter parameters such as sensor confidence and the number of effective particles provide for improved target tracking, and is therefore necessary when particle weight is to be used as the path selection metric.

Chapter 6

Conclusion

The development of a particle filter based method of localizing, tracking, and intercepting a mobile ground target in an urban scenario given nontraditional and inconsistent measurement data was presented. The large regional sensor model was a significant challenge in this work. The target's location could only be measured to be within the region of the detection, and the time span between measurement reports reinforced the importance of a sophisticated dynamic model. In the presence of a perfect sensor, the dynamic model used to propagate the particle cloud in time plays a vital role in tracking performance. In addition, the importance of particle spatial resolution must be taken in to consideration to improve tracking performance and reduce detection time in the presence of the aforementioned measurement model.

In the presence of a sensor with a false report rate, the measurement update becomes more important than the dynamic model update. The particle filter's belief in the sensor's false report rate, coupled with the frequency at which the distribution is resampled must be taken in to account in order to maintain proper spatial resolution of the particle cloud in the presence of a sensor with a false report rate. Additionally, means must be taken to assess when a measurement is possibly false. The flux of particle weight into or out of a region may be used to assign a cost to accepting a measurement through the Bayesian risk assessment framework. A risk assessment is necessary to provide proper tracking in the presence of false measurements. A risk assessment and proper tuning of the number of effective particles and sensor belief proved beneficial in maintaining proper spatial resolution in the presence of a sensor with a false report rate.

These conclusions proved vital in the navigation of a UAV to the target by providing motivation for a new path selection metric. A metric for path selection in the particle filter framework was developed based on the cumulative sum of particle weight to be encountered on a path. This metric provides insight to future likelihood of target intercept while taking advantage of the particle filter framework. A comparison was done to an existing method where the new metric matched performance and required reduced computational expense. This work will benefit the modern war fighter, in addition to researchers considering a particle filter due to a non-differentiable measurement model.

Additionally, it is observed that given set computational capabilities, a tradeoff may be made to give priority to the dynamic model or the path selection metric. The amount of time to simulate one second with the Dispersion model and the Information Utility Function is approximately equal to that of the use of the Traffic Motion Model and the Maximum Likelihood method. Significantly improved tracking performance results from using the Traffic Motion Model, while target intercept time is not as significantly delayed by use of the Maximum Likelihood method. By developing a sophisticated dynamic model, a less computationally intensive path selection metric is applicable and successful.

Further improvement to the work discussed herein would be achieved by a more robust risk assessment routine to reject false measurements. Currently, if the risk is deemed unacceptable, the measurement is still accepted but the particle distribution is not resampled. Investigation into the practice of rejecting measurements given the current framework could provide for improved tracking performance and false alarm rate reduction.

In addition, further study into the effect of various values of the coefficients R and Q in the cost function could provide decreased intercept times. In the presented results, the distance term in the cost function is the distance between the UAV and the expected value of the particle cloud. This is an effective metric after the target has been found by the regional binary sensor, but ineffective before that time of detection because the expected value tends to be in the center of the city until the target is detected. An improvement to this would be

to adjust Q to include the distance term only after the target is found, or to use the distance between the UAV and the center of the region of highest weight until the target is found.

Future expansion of this problem could include the idea of particle interaction, which may provide for improved tracking performance. Particles are statistically required to be uncorrelated, and therefore may not necessarily be aware of what other particles are choosing to do. However, real time traffic updates are often readily available, and would fit nicely into the presented framework to allow for adaptive particle motion models, without violating the statistical properties of the particle filter.

Finally, mobile sensor units could be introduced into the existing framework with minimal effort. This could provide more a more realistic sensor network structure. In addition, the present framework permits the acceptance of multiple measurement reports at once or at varying intervals. This would provide for the inclusion of measurements from additional sources such as security cameras and satellite images.

Bibliography

[1] Ristic, B., Arulampalam, S., and Gordon, N., *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House, Boston, 2004.

[2] Crassidis, J.L. and Junkins, J.L., *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[3] Gordon, N.J., Salmond, D.J., and Smith, A.M.F, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings-F*, vol. 140, no.2, pp. 107-113, 1993.

[4] Doucet, A., Godsill, S., and Gordon, N., "An introduction to sequential Monte Carlo Methods," in *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

[5] Boers, Y., Driessen, J.N., "Particle filter based detection for tracking," *Proceedings of the American Control Conference*, vol. 6, pp. 43934397, 2001.

[6] Kalman, R.E., "A new approach to linear filtering and prediction problems," *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)), 3545, 1960.

[7] Kalman, R.E., Bucy, R.S., "New results in linear filtering prediction theory," *Transactions of the ASME Journal of Basic Engineering*, 83(1), 95108, 1961.

[8] Liu, J.S., *Monte Carlo Strategies in Scientific Computing*, Springer, Verlag, 2001.

[9] Agate, C.S., Wilkerson, R.M., Sullivan, K.J., "Utilizing negative information to track ground vehicles through move-stop-move cycles," *Proceedings of SPIE on Signal Processing, Sensor Fusion, and Target Recognition*, vol. 5429, pp. 273283, 2004.

[10] Arulampalam, M.S., Gordon, N., Orton, M., Ristic, B., "A variable structure multiple model particle filter for GMTI tracking," *Proceedings of the International Conference on Information Fusion*, vol. 2, pp. 927934, 2002.

[11] Agate, C.S., Sullivan, K.J., "Road-constrained target tracking and identification using a particle filter," *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets*, vol. 5204, pp. 532543, 2003.

[12] Yang, C., Bakich, M., Blasch, E., "Nonlinear constrained tracking of targets on roads," *Proceedings of the International Conference on Information Fusion*, pp. 235242, 2005.

[13] Ulmke, M., Koch, W., "Road-map assisted ground moving target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, 42(4), 12641274, 2006.

[14] Ekman, M., Sviestins, E., "Multiple model algorithm based on particle filters for ground target tracking" In: Proceedings of the International Conference on Information Fusion, pp. 18, 2007.

[15] Kamrani, F., Ayani, R. "Simulation-aided path planning of UAV" In: Proceedings of the 2007 Winter Simulation Conference, pp. 13061314 (2007)

[16] Orguner, U., Schon, T.B., Gustafsson, F., "Improved target tracking with road network information" In: Proceedings of the IEEE Aerospace Conference, pp. 111 (2009)

[17] Hong, L., Cui, N., Bakich, M., Layne, J.R.: Multirate interacting multiple model particle filter for terrain-based ground target tracking. IEE Proceedings on Control Theory and Applications 153(6), 721731 (2006)

[18] Kravaritis, G., Mulgrew, B.: Variable-mass particle filter for road-constrained vehicle tracking. EURASIP Journal on Advances in Signal Processing 2008, 113 (2008)

[19] Skoglar, P., Orguner, U., Tornqvist, D., Gustafsson, F.: Road target tracking with an approximative Rao-Blackwellized particle filter. In: Proceedings of the International Conference on Information Fusion, pp. 1724 (2009)

[20] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration, IEEE Signal Processing Mag., pp. 6172, March 2002.

[21] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. David, "Collective motion, sensor networks, and ocean sampling, Proc. IEEE, vol. 95, no. 1, pp. 4874, January 2007.

[22] X. Feng, K. A. Loparo, and Y. Fang, "Optimal state estimation for stochastic systems: An information theoretic approach, IEEE Trans. Automat. Contr., vol. 42, no. 6, June 1997.

[23] C. Kreucher, K. Kastella, and A. O. Hero III, "Information based sensor management for multitarget tracking, in Proc. SPIE, vol. 5204, Bellingham, WA, August 2003, pp. 480489.

[24] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Mutual information methods with particle filters for mobile sensor network control, in Proc. 45th IEEE Conf. Decision and Control, San Diego, CA, December 2006, pp. 10191024.

[25] Hoffmann, G.M., Tomlin, C.J., "Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters", *IEEE Transactions on Automatic Control*, vol. 5, Issue 1, p. 32-47, 2010.

[26] Skoglar, P.; Orguner, U.; Gustafsson, F. , "On information measures based on particle mixture for optimal bearings-only tracking," Aerospace conference, 2009 IEEE , vol., no., pp.1-14, 7-14 March 2009.

[27] Boers) Boers, Y., Driessen, H., Bagchi, A., and Mandal, P., "Particle Filter Based Entropy", *13th Conference on Information Fusion (FUSION)*, 2010, vol., no., pp.1-8, 26-29 July 2010.

[28] Ryan, A., "Information-theoretic tracking control based on particle filter estimate," in *Proceedings AIAA Guidance, Navigation, and Control Conf.*, Honolulu, HI, August, 2008.

[29] Berg-Yuen, P.E.K, Mehta, S.S., Waden, D.L., Curtis, J.W., "Particle Filter-Based Ground Target Tracking in a Constrained Road Network", 3rd International Conference on the Dynamics of Information Systems, 2011.

[30] LaValle, S., *Planning Algorithms*, Cambridge University Press, 2006.

[31] Spletzer, J.R. and Taylor, C.J., "Dynamic Sensor Planning and Control for Optimally Tracking Targets", *The International Journal of Robotics Research*, No. 1, Jan. 2003, p. 7-20.

[32] Wang, Y., Hussein, I.I., and Erwin, R.S., "Risk Based Sensor Management for Integrated Detection and Estimation", *AIAA Journal of Guidance, Control, and Dynamics*,2011, (in press).

[33] O'Reilly, P.G., "Local Sensor Fault Detection Using Bayesian Decision Theory", Conference Proceedings: UKACC International Conference on Control, Sept. 1998, Vol. 1, p. 247-251.

[34] Nelson, D.R.; Barber, D.B.; McLain, T.W.; Beard, R.W.; , "Vector Field Path Following for Miniature Air Vehicles," Robotics, IEEE Transactions on , vol.23, no.3, pp.519-529, June 2007.

[35] Grinstead, C.M. and Snell, J.L., *Introduction to Probability*, American Mathematical Society, 1998.

Appendices

Appendix A

Expected Travel Time

A performance metric is desired for the estimated travel time between two random points in a 1000 m x 1000 m urban environment. A UAV approaching a stationary target at a constant speed is considered. Assuming travel is required in a grid-like fashion, travel is only permitted in one direction at at time. This metric could be determined by dividing the expected value of distance traveled between two random points in the environment by the constant speed of travel. The following derivation of the expected value of the distance between two random points is used in the comparison of the path planning metrics.

The expected value of a random variable is given by:

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx \tag{A.1}$$

Equation (A.1) is to be used to determined the expected value of the distance between to random points in a uniform distribution. Therefore, an expression for the probability density function for the distance between two random points is required.

Consider two independent random numbers from the interval $[0, d]$, X and Y, where X represents the x-coordinate of the UAV starting position and Y represents the target's x-coordinate. The probability density functions of X and Y are given by:

$$f_X(x) = f_Y(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq d \\ 0 & \text{if otherwise} \end{cases} \tag{A.2}$$

The difference between X and Y is defined as $Z = X - Y$. Because X and Y are independent, the density of their sum is the convolution of their densities [35]. Therefore, the density

function of $Z = X - Y$ is:

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(z+y) f_Y(y) dy \tag{A.3}$$

Because $f_Y(y) = 1$ if $0 \leq y \leq d$ and 0 otherwise, $f_Z(z)$ reduces to

$$f_Z(z) = \int_{0}^{d} f_X(z+y) dy \tag{A.4}$$

Now, the integrand is 0 unless $0 \leq z+y \leq d$ (i.e., unless $-z \leq y \leq d-z$), where it is 1. So, if $0 \leq z \leq d$,

$$f_Z(z) = \int_{0}^{1-z} dy = d - z \tag{A.5}$$

and if $-d \leq z \leq 0$,

$$f_Z(z) = \int_{-z}^{d} dy = d + z \tag{A.6}$$

and $f_Z(z) = 0$ otherwise.

The previous example was done in terms of a uniform distribution between 0 and an unknown constant d that corresponds to the dimension of the space. Because distance is always positive, the expected value of the absolute value of z is taken.

$$\begin{aligned} E(|z|) &= \int_{-\infty}^{\infty} |z| f_Z(z) dz \\ &= \int_{-d}^{0} -z(d+z) dz + \int_{0}^{d} z(d-z) dz \\ &= d/3 \end{aligned}$$

The value of d in this study is 1000 m, therefore the expected value of the distance in the x-direction between two points is $\rho_x = 1000/3 = 333.33$ m. The same would result in the y-direction. The grid-like structure of the road network in this work requires travel in one direction and then another. This results in a total expected value for the traveled distance

between two points in the given 1000 m x 1000 m x-y plane (ρ) as:

$$\begin{aligned}\rho &= \rho_x + \rho_y \\ &= 333.33 \text{ m} + 333.33 \text{ m} \\ &= 666.66 \text{ m}\end{aligned}$$

Given a constant speed of 20 m/s and assuming the target remained stationary, this would result in an expected travel time of 33.33 sec.