2007-03

Particle filter based tracking in a detection sparse discrete event simulation environment

Borovies, Drew A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PARTICLE FILTER BASED TRACKING IN A DETECTION SPARSE DISCRETE EVENT SIMULATION ENVIRONMENT

by

Drew A. Borovies

March 2007

Thesis Advisor:	Christian Darken
Second Reader:	Arnold Buss

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2007	Master's Thesis

4. TITLE AND SUBTITLE: Particle Filter Based Tracking in a Detection Sparse Discrete Event Simulation Environment	5. FUNDING NUMBERS
6. AUTHOR(S): Drew A. Borovies	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

One of the key abilities of agents in military simulations is to react to both detections of and counter-detections by other agents in the environment. While methods have been developed to model these detections and counter-detections, the majority of these methods model detection and counter-detection as an all or nothing prospect in which an un-detected entity at some point crosses an arbitrary threshold of observability and becomes fully detected. In actuality, even extremely uncertain or incomplete detections and counter-detections of opposing entities can provide enough data for entities to make reasonably intelligent decisions on the virtual battlefield. Recent developments in commercial gaming artificial intelligence suggest that particle-based tracking techniques can provide accurate and computationally efficient state estimation of opposing agents within virtual environments. In this work several particle-based methods for obtaining and tracking contacts are explored to determine the feasibility of their use as a general purpose tracking technique in military simulations.

14. SUBJECT TERMS Particle-based tracking, agents, active/passive sensing modes.	15. NUMBER OF PAGES 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PARTICLE FILTER BASED TRACKING IN A DETECTION SPARSE DISCRETE EVENT SIMULATION ENVIRONMENT

Drew A. Borovies
Lieutenant, United States Navy
B.S., Virginia Tech, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
March 2007

Author: Drew A. Borovies

Approved by: Christian Darken
Thesis Advisor

Arnold Buss
Second Reader

Rudolph Darken
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

One of the key abilities of agents in military simulations is to react to both detections of and counter-detections by other agents in the environment. While methods have been developed to model these detections and counter-detections, the majority of these methods model detection and counter-detection as an all or nothing prospect in which an un-detected entity at some point crosses an arbitrary threshold of observability and becomes fully detected. In actuality, even extremely uncertain or incomplete detections and counter-detections of opposing entities can provide enough data for entities to make reasonably intelligent decisions on the virtual battlefield. Recent developments in commercial gaming artificial intelligence suggest that particle-based tracking techniques can provide accurate and computationally efficient state estimation of opposing agents within virtual environments. In this work several particle-based methods for obtaining and tracking contacts are explored to determine the feasibility of their use as a general purpose tracking technique in military simulations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all I would like to thank my thesis advisor Dr. Christian J. Darken for his expert guidance through the thesis process. His responses to the difficulties I encountered during the course of this work seemed to always take the form of recent publications which directly addressed the issues with which I was wrestling. Additionally, his ability to understand and translate my “navy-speak” into mainstream English was instrumental in allowing this work to be completed in a manner which should be reasonably understandable by those without the benefit of a surface navy background.

Without the timely support and proof-reading efforts of my family this work would never have been finished. Despite a heavy travel schedule between Virginia, Pennsylvania, and Hawaii and an impending deployment my family turned draft copies of my thesis around loaded with insightful comments in mere hours. Their lack of simulation background helped to ensure that the documentation of my work with particle filters in the form of this thesis was not written to be understood only by simulationists and computer scientists.

Finally, I would like to thank my classmates at the MOVES Institute for their support and counsel through the “thesis-crunch” at the Naval Postgraduate School. As they mentioned several times, if my work amounted to nothing else, I had successfully designed perhaps the most hypnotic screen-saver to ever grace the screens of naval laptops. Their companionship in the “geeked-out” MOVES study room and good-humor through late nights of coding kept me working on my thesis when I otherwise would have thrown up my hands in frustration and left the work for another day.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT

A continuing effort in both the commercial gaming and defense simulation industry is the advancement of more reasonable actions by computer controlled entities. One of the methods of injecting increased realism into virtual environments is the increased use of autonomous agents in both gaming and simulation applications. An agent is a representation of an entity in the environment which encapsulates some level of autonomous decision making capability. Empowering individual entities to act within environments of their own accord can result in simulation or game outputs with higher levels of realism, particularly if agents' knowledge of those environments is limited.

One of the problems associated with allowing agents to make autonomous decisions within an environment is that of representing the uncertain positions or states of other entities in such a way so as to allow agent decisions to approximate those which would be taken by a human under the same circumstances. While commercial game artificial intelligence has often allowed computer controlled players to "cheat" and have perfect knowledge of the environment, this solution is undesirable in military simulations. Attempts at limiting the amount of information available to computer controlled entities to what could be considered a "reasonable" level must rely on some base representation of the positions of other entities within the environment. A series of discrete observations based on this representation must be flexible enough to provide an increasingly accurate or inaccurate picture of the location or actions of another entity, or a track on that entity, as the situation warrants.

The goal of this research is to examine the feasibility of using particle filter based state estimation techniques as a general purpose method of representing entity situational awareness in military simulations. Methods of modeling detections in military simulations are well developed. Models representing detections via visual, aural, and electromagnetic means have been created with varying levels of fidelity and included across the whole spectrum of military simulations. The conversion of a single detection or series of detections into tracking information for use by entities within the simulation

is to a certain extent also a solved problem. Differing types of detections yield widely varied amounts of state information about the objects being tracked; as a result a wide array of tracking methods have been developed for use in creating tracks from detection information.

While tracking methods are as varied as the models used to manage detections, they are generally tuned to accurately model tracks acquired from specific types of detections. Detection types can generally be classified into one of two general classes: active and passive detections. Although there are exceptions, tracking methods currently employed in military simulations can be classified into those that represent active or passive tracks with little ability to accurately portray tracks acquired through the other class of detections. This is due to the fundamentally different nature in which the detections which are the basis for these two types of tracks are acquired.

Active detections result when a sensor emits some kind of energy into the environment. When this energy "bounces" off another entity in the environment, portions of that energy will return to the vicinity of the originating sensor and the characteristics of this returning energy can be examined by the sensor in its attempts to detect the other entity. Detections acquired in this manner are generally exact in nature, providing both range and bearing information to the sensor with relatively little uncertainty. Radar, laser range-finders, and echo-location are examples of sensors which achieve detections through active means.

The exact nature of active detections makes tracking objects acquired through these means a deceptively trivial affair. The comparisons of a series of detections can result in very accurate course and speed information of the entity being tracked. With high enough rates of emission by the active sensor in question, active tracks can be very responsive to changes in target motion. The exact nature of active tracks must be counter balanced with the fact that the emission of energy into the environment by the active sensor can be utilized by other entities to yield passive detections of the platform sensing through active means.

Passive detections result from a sensor's observation of the surrounding environment. As these observations are obtained, the sensor pulls out details which

correspond to the emissions of other entities into the environment. These observations, or passive detections, can then be used to create passive tracks of the object in question. Detections acquired through passive means are often ambiguous in nature and yield comparatively uncertain information in a tracking sense in comparison to active detections. Sight, hearing, and passive sonar are examples of passive detection modalities.

Methods used to track objects through passive means can generally be separated into those which represent tracked-entity location using managed areas of uncertainty and those that perform target motion analysis (TMA) on a series of passive detections. The former method seeks to bound possible target locations within an area of interest. Some level of knowledge about the target's capabilities is then used to ensure that the area of uncertainty changes as necessary to continually contain the tracked entity. TMA seeks to determine the heading and speed of the tracked entity through the observation of the changing characteristics of a series of passive detections.

While methods exist to track through both active and passive means, the inclusion of several different tracking algorithms in a simulation to adequately handle both types of tracks presents coordination and complexity problems. Many simulations, in a nod to these difficulties, model passive detections through the use of passive detection thresholds and distributions. When the possibility of passive detection occurs, these thresholds and distributions are used to determine if the possible detection was of a meaningful enough nature to result in the acquisition of the entity in question. If an acquisition is calculated to have occurred, the nature of the track used by the detecting entity will resemble an exact active detection of the same entity. In essence, the passive detection problem is treated as an "all-or-nothing" affair. A tracking method capable of handling both active and passive detections with seamless transitions between both detection types would address this problem. Unfortunately, such a tracking method has not yet been adopted by the military simulation community.

The use of particle filters as state-estimation tools was proposed in (Bererton 2004). Tracks obtained using this method, while not directly comparable to detections and tracking in military simulations, somewhat resemble active tracks. In one discussion

of particle-based communication among game agents (Klaas *et al.*, 2005), an example containing fused data from a ranging sensor and a direction sensor indicates that this method could handle passive detections as well as the active detections first discussed. The ability to track entities through both types of detections seems to indicate that this particle filter method could be used in certain classes of military simulations where exacting degrees of tracking fidelity are not required. Prior to this adoption, however, it is necessary to examine some of the differences between the commercial game environment of the proposed particle filter tracking methods and environments in military simulations.

The particle filter state estimation methods mentioned above are proposed for use in relatively small environments, notably in first-person shooter-like domains. These types of environments, while varying in levels of complexity, are notable for the relatively large area of regard of the sensors employed by the agents. One of the key results of this feature is that it can be reasonably assumed that the entire environment can be included in an estimation of the state of opposing agents. By contrast, many military simulations occur in very large environments where the state estimation of opposing entities, were it to take into account the entire environment, would be prohibitively expensive.

Another side effect of the comprehensive area of regard in commercial gaming environments is that continued observations of the environment yield large amounts of information about the state of opposing entities either through the confirmation or rejection of previous state estimations. As the effective area of regard of utilized sensors in an environment decreases, the state-estimation method being employed must be able to make more ambiguous approximations about the state of other entities while retaining the tracking robustness to take into account a wider range of states. Agents representing surface ships on the open ocean might contain tracks on other surface ships acquired days earlier which are currently far outside of detection range, tracks which are not currently held with its own sensors but which are being actively tracked by other friendly surface ships, and tracks which it currently holds with its own sensors all at the same time. As the simulation environment in question grows larger, the number of concrete observations provided by agent sensors decreases while the number of ambiguous passive detections

increases. An adaptation of the particle-filter method of state estimation for use in military simulations must be able to manage a large number of vague passive detections and obtain meaningful information from those detections in addition to providing exact detection and tracking information when the situation warrants.

B. RESEARCH APPROACH

As current implementations of particle filter based state estimation are few, this research will require the construction of a simple environment that adequately reflects the environmental concerns addressed above. Once a framework allowing entities in the environment to track other entities using particle-filters has been implemented, various refinements and techniques for maintaining these particle-filters will be examined to determine the techniques' relative strengths and weaknesses.

C. THESIS ORGANIZATION

The rest of this thesis is organized as follows:

- **Chapter II, Background,** describes various techniques for representing uncertainty distributions and entity states in various fields of study. Methods of using these different techniques as aids to decision making, communication, and prediction are also discussed.
- **Chapter III, Model,** describes the nature of the environment within which the particle tracking technique will be tested. The implementation-specific information about the particle tracking techniques being evaluated will be presented along with the manner in which these techniques were used to enable rudimentary decision making within the virtual environment.
- **Chapter IV, Analysis of Different Tracking Methods**, provides a quantitative analysis of the relative strengths and weaknesses of the different particle tracking methods developed in active and passive contexts with regard to their possible utility in a simulation environment.
- **Chapter V, Conclusions and Future Work,** summarizes the contribution made by this thesis and discusses possible future expansions to the work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. DISTRIBUTION REPRESENTATION

Computer applications requiring agents to track other objects must have some way of representing the locations of other entities in the environment. While a traditional approach in the computer gaming industry has been to provide perfect information to computer controlled players, the practice of limiting environmental knowledge to provide a more realistic experience to players has become increasingly prevalent. Commercial programmers now face the challenge of representing agent knowledge of other entities' position in terms of some probability distribution.

While representing environmental knowledge encompasses a huge number of research areas, the one with which this work is most concerned is representation of object location with some level of uncertainty. Acquiring and tracking a target through the use of uncertainty distributions requires both the means to represent an individual "detection" in an uncertain manner and to estimate target state information from a series of detections. Several methods for representing uncertainty distributions which were considered in the course of this work are discussed below to provide an understanding of the difficulties present in tracking objects through a virtual environment.

Gaussian distributions are briefly reviewed as a departure point for other more exotic methods of representing uncertainty. Gaussian distributions have the advantage that they are mathematically well developed and are a traditional method of representing a position-distribution. However, the extension of Gaussian distributions to the representation of complex or discontinuous uncertainty distributions is somewhat troublesome, and the review of occupancy maps as a tracking method below discuss these problems and proposes a method for handling these difficult situations in a discrete manner. Particle filters address the shortfalls of Gaussian distributions in a continuous rather than discrete manner, and recent proposals have suggested their use in commercial game artificial intelligence for tracking and search. As such, and because they are the basis of this work, they are also reviewed below. The Parzen-windows approach to density estimation is a method for obtaining uncertainty distribution information from a

large number of samples. Although this technique is not a basis for this work, it is used in several instances to visualize the state of particle filters and so is reviewed below.

1. Gaussians

Multivariate Gaussian distributions, also known as multivariate normal distributions, can be thought of as a generalization to multiple dimensions of the one-dimensional normal distribution. With an estimated position and a known measurement error, a probability density distribution can be created to represent the range of possible target locations. As Gaussians are based on the normal distribution, this range of possible locations will encompass the entire environment (although at probabilities close to zero far from the mean). A graph of the standard bivariate normal distribution is shown below:

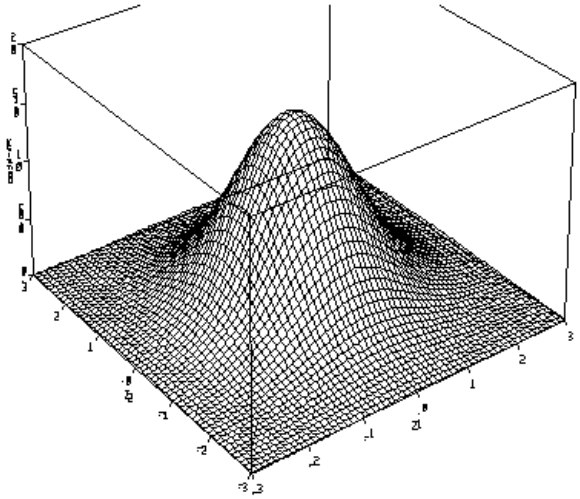


Figure 1. 3D Graph of the Standard Bivariate Normal Distribution

In (Stroupe *et al*., 2000) bivariate Gaussian distributions are examined for their use by robots playing soccer. This discussion begins with a presentation of the canonical form of a two dimensional Gaussian dependent on standard deviations, a covariance matrix, and mean:

$$p(X) = \frac{1}{2\pi\sqrt{|C|}} \exp\left(-\frac{1}{2}\left(X - \bar{X}\right)^T C^{-1} \left(X - \bar{X}\right) \right) \tag{0.1}$$

$$C = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} \tag{0.2}$$

Note that in the above equations X is a vector containing x and y values and that $\bar{X}$ is a vector containing the mean x and y of the distribution. This canonical form represents a Gaussian oriented in the x, y plane. Unfortunately, observations are not normally made in this manner. A more likely method of obtaining observations is through a relative coordinate system. Stroupe's discussion of the problem provides a local coordinate system in line with observations made by soccer playing robots with parameters as shown:

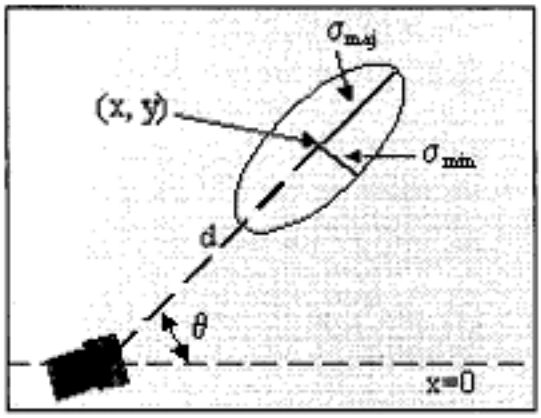


Figure 2. Distribution Parameter Definitions (From Stroup *et al.*, 2000)

The figure above shows that an observation taken in a local coordinate system will likely consist of an observed mean (x, y), an angle corresponding to the major axis of the observation (θ), major and minor axis standard deviations ($\sigma_{maj}, \sigma_{\min}$), and a distance to the mean (d). In order to work with an observation taken in an arbitrary

coordinate system, it must be transformed to the canonical coordinate system. Stroup accomplishes this by first determining the initial covariance matrix of the observation:

$$C_L = \begin{bmatrix} \sigma_{maj}^2 & 0 \\ 0 & \sigma_{\min}^2 \end{bmatrix} \tag{0.3}$$

A rotation of X in equation 1.1 by θ leads to:

$$C^{-1} = R(-\theta)^T C_L^{-1} R(\theta) \Rightarrow C = R(-\theta)^T C_L R(-\theta) \tag{0.4}$$

Transforming Gaussian observations from arbitrary coordinate frames to the canonical form allows Gaussians corresponding to multiple observations to be "merged" so that an estimate of the target's position can be refined to reflect observations from multiple platforms, sensors, or moments in time. Merging multiple observations requires the combination of individual covariance matrices, the computation of the mean of the merged distributions, and the principle axis of the merged distributions. These steps are accomplished using the following formulae:

$$C' = C_1 - C_1\left[C_1 + C_2\right]^{-1} C_1 \tag{0.5}$$

$$\bar{X}' = \bar{X}_1 + C_1\left[C_1 + C_2\right]^{-1}\left(\bar{X}_2 - \bar{X}_1\right) \tag{0.6}$$

$$\theta' = \frac{1}{2}\tan^{-1}\left(\frac{2B}{A-D}\right) \tag{0.7}$$

In equation 1.7 A, B, and D correspond to the top left, top right/lower left, and lower right entries of the merged covariance matrix, respectively. Once the principle axis of the merged distribution has been computed, the rotation into canonical form is reversed:

$$C' = R(\theta')^T C' R(\theta') \tag{0.8}$$

Stroupe demonstrates the effectiveness of this method of merging Gaussians with an example simulating the combined observations of two robots:

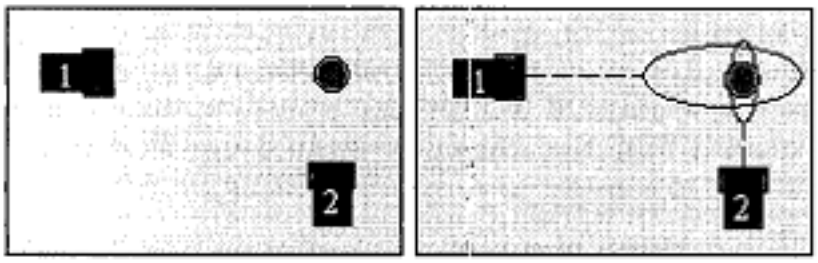


Figure 3. Two Distributed Robots Observe a Target (From Stroupe *et al.,* 2000)

In the left picture, two distributed robots see a target. In the right picture, these observations have generated Gaussians oriented in relative coordinate axes. The individual Gaussian distributions are shown at the left and their merged counterpart is shown at the right in the figure below:

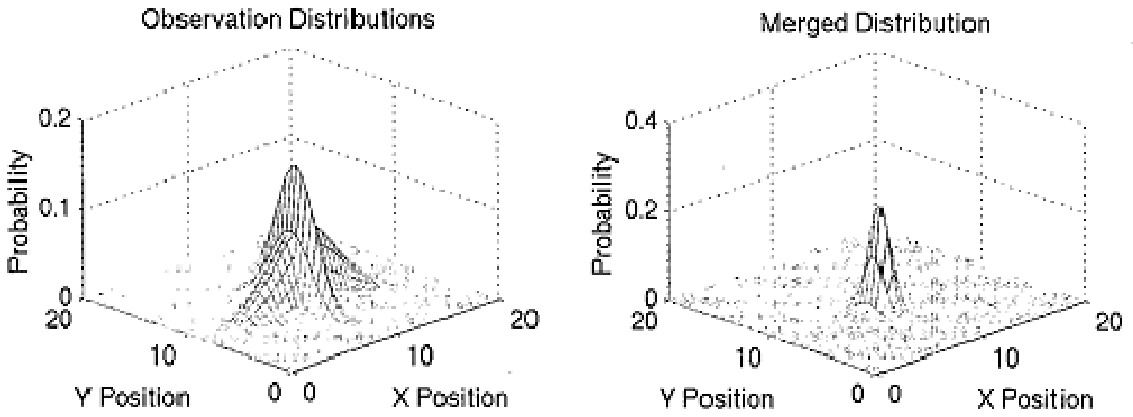


Figure 4. Merged Distributions from Two Observations (From Stroupe *et al.* 2000)

While the discussion above shows that Gaussians can be effectively used to represent target uncertainty, the nature of the underlying distribution imposes constraints on the use of this technique in complex environments. Theses issues, and an uncertainty representation which addresses these issues, are discussed in reviews of occupancy maps and particle filters in following sections.

2. Occupancy Maps

In being used as a position estimator the Guassian has a fundamental weakness. Due to the continuous nature of the underlying probability distribution, it is mathematically difficult (and perhaps impossible) to invalidate portions of the probability distribution while leaving other portions unchanged. The figure below illustrates this problem. The left image is of a continuous one-dimensional prediction. The right image shows the same prediction with a span removed.

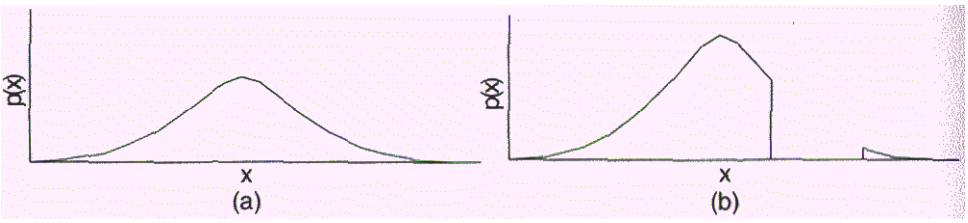


Figure 5. Continuous Prediction vs. Discontinuous Prediction (From Isla, 2006)

Occupancy maps address this problem by transitioning from a continuous probability model to a discrete one. In essence, a grid is projected onto the environment and each portion of the grid is treated as an area of probability. This allows the probability for any given node to be adjusted based on the observability of that node and properties of the tracker and object being tracked. In (Isla, 2006) the figure below is provided as an impetus for examining the occupancy map approach to uncertainty representation.

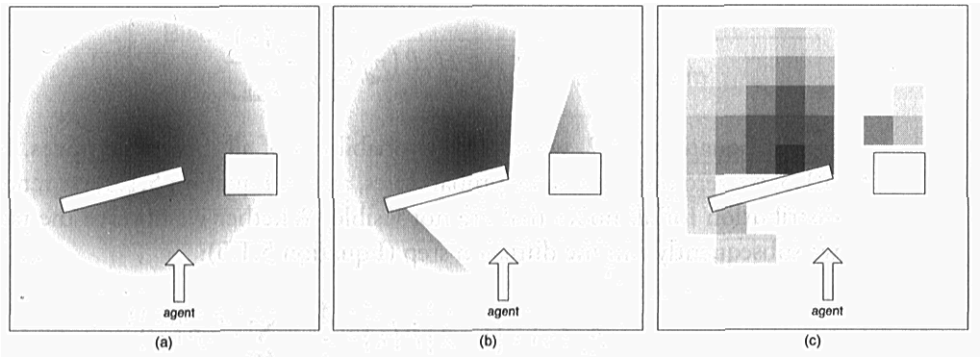


Figure 6. The Problem of Spatial Representation (From Isla, 2006)

At the left, the inability to invalidate portions of the distribution results in the loss of meaningful data about spaces observed to be target-free. The ideal distribution is shown in the middle. By laying a grid over the environment treating each of the grid nodes as a separate "bucket" of probability, the distribution at the right can be obtained.

Isla provides a straightforward algorithm to update the probabilities of grid nodes at each update cycle. A separate method for updating probabilities is necessary for both when the target is observed and when it is unobserved. When the target is observed, the probability distribution is centered around the node, n^*, where the target was observed:

$$\begin{aligned} P_t(n^*) &\leftarrow 1 \\ P_t(n \neq n^*) &\leftarrow 0 \end{aligned} \tag{0.9}$$

In the above equations $P_t(n)$ is the probability that the target is contained in node *n* at time-step *t*. When the target is observed at n^* the probability at that node is set to 1 (or 100%) and the probability at all other nodes is set to 0. When the target is not observed, the nodes can be separated into *V* (visible) and *H* (hidden) sets with their probabilities updated using the following equations:

$$P_{culled} = \sum_{n \in V} P_{t-1}(n), \tag{0.10}$$

$$\forall n \in V, P_t(n) \leftarrow 0, \tag{0.11}$$

$$\forall n \in H, P_t(n) \leftarrow \frac{P_{t-1}(n)}{1 - P_{culled}}. \tag{0.12}$$

In Equation (1.10) the probabilities of all visible nodes are added together. Equation (1.11) then zeros-out the probabilities in the visible nodes as they are known to not contain the target. In Equation (1.12) the distribution of probabilities is renormalized for the nodes that are not visible at that time-step.

Regardless of whether the target is visible or not, a diffusion step is proposed which accounts for the spread of uncertainty associated with the movement of the target:

$$P_{t+1}(n) = (1-\lambda)P_t(n) + \frac{\lambda}{4}\sum_{n' \in neighbors(n)} P_t(n'). \quad (0.13)$$

In Equation (1.13) λ is a diffusion constant in the range $[0,1]$ which reflects the rate at which an agent becomes uncertain about the target's location. The above expression also assumes a square grid with each node having four neighbors. Isla remarks that a hexagonal grid is more desirable than a square grid when dealing with diffusion, as such a grid will result in fewer artifacts when using the simple diffusion model. The adaptation of the above diffusion model to a grid of different polygonal construction is straightforward if the nodes of the grid in question are like sized.

The most computationally expensive segment of the occupancy map algorithm lies with determining which nodes of the map are currently visible and which nodes are not. Performing point-of-view renderings of the environment and ray-cast sampling of several discrete points are proposed as possible methods of making this determination. One of the benefits of using this model is that it can be used to represent simple search behaviors with little effort. An agent attempting to find a target using this uncertainty representation could quite simply approach the grid node in the environment containing the highest probability continually updating its environmental model as it moves. A systematic search for the target would result as probable hiding places are searched and discarded.

Isla also proposes two simple examples of using the occupancy map model to approximate emotional behavior. These behaviors would compare the probability of the location where the target is eventually found with the amount of probability culled at each time-step when the target is not located. When the target is located in a relatively unlikely location a certain level of "surprise" could be represented. Likewise, when the target is not located following the observation of very likely locations "confusion" could result. While occupancy maps elegantly handle complex environments, the reduced accuracy resulting from a discrete environment would limit its effectiveness in military simulations. Particle filters are a way to extend this idea to a continuous environment.

3. Particle Filters

A particle based state estimation technique for game artificial intelligence has recently been proposed in (Bererton, 2004). This technique seeks to address problems in representing discontinuous or irregular probability distributions in a continuous manner as opposed to the discrete manner of the occupancy map technique. In essence, the actual state of the entity being tracked is assumed to come from some distribution which may or may not be of a regular nature. A number of samples (particles) are drawn from this distribution and can then be used to estimate the state of the object being tracked. At each time step observations of the tracker are used to manipulate the particles in such a way so as to fine-tune the estimate as to the state of the entity being searched or tracked.

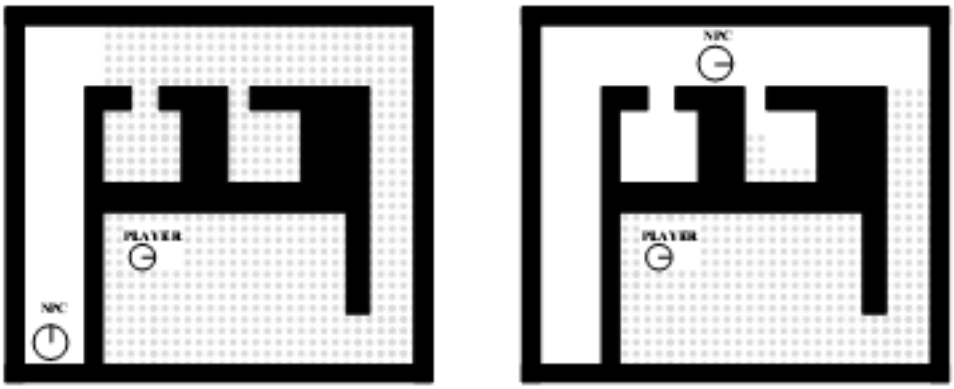


Figure 7. Illustration of Particle Filter State Estimation (From Bererton 2004)

The figure above illustrates this concept. In the left picture, the particles (small shaded circles) represent possible locations of the player. In the right picture, the non-player character (NPC) has moved, and in doing so made several line-of-sight observations which did not result in acquiring the player. The particles residing in those areas which came under observation have been removed from consideration. In Bererton's implementation, the NPC continually moves towards the mean of the particle distribution while making observations, resulting in a systematic search for the player.

Creating and maintaining particle filters so they can be used to acquire and track targets is a relatively simple process. In (Arulampalam *et al.*, 2001) theories, issues, and algorithms for implementing several types of particle filters for tracking are provided. One particular type, the Sampling Importance Resampling (SIR) filter, was chosen by

Bererton for use as a proof of concept that particle filters could be used for state estimation in a simple game environment. The SIR particle filter algorithm is shown below:

SIR PARTICLE FILTER

$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = SIR[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

FOR $i = 1: N_s$ ** Generate Proposal Distribution **

Draw a sample x_k^i from $p(x_k \mid x_{k-1}^i)$

END FOR

FOR $i = 1: N_s$ ** Incorporate Observations **

Calculate $w_k^i = p(z_k \mid x_k^i)$

END FOR

** Renormalize Weights to Sum to One **

Calculate total weight: $t = SUM[\{w_k^i\}_{i=1}^{N_s}]$

FOR $i = 1: N_s$

$w_k^i = t^{-1} w_k^i$

END FOR

- Resample distribution according to $[\{x_k^i, w_k^i, -\}_{i=1}^{N_s}] = RESAMPLE[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$

Figure 8. Sampling Importance Resampling Filter Algorithm (After Arulampalam *et al.*, 2001)

The SIR algorithm is called at every time-step or frame in the game environment and consists of three main steps. The first step is the generation of the proposal distribution. This details sampling a number of times (N_s) the distribution of particles

from the previous time-step to create a rough estimate of the position of the object being tracked. While this follows neatly from the distribution of particles at a given time-step in the middle of execution, there must be some initial distribution from which to start. In Bererton's implementation this initial configuration of particles is a uniform distribution throughout the game environment. This initial configuration could also be tailored to reflect some prior knowledge or intelligence estimate of the target in question.

The second step of the SIR algorithm incorporates tracker observations of the environment to refine the proposal distribution obtained from the first step of the algorithm. This amounts to adjusting the weights of the particles which currently fall under observation. If the target is not currently being observed by the NPC, then the weights of particles will be lowered or reduced to zero. If the target is currently being observed, the weights of the particles will be raised. Once this is accomplished, the weights of all samples are renormalized so that sum of all particle weights will be approximately equal to one.

The third step of the SIR algorithm is a re-sampling of the distribution. This is needed to maintain filter diversity and avoid the effects of degeneracy. In (Arulampalam *et al.* 2001) a description of the particle filter degeneracy phenomenon is provided. Particle filter degeneracy takes the form of negligible weights for the majority of particles in the track. This is undesirable as it implies that a large amount of computational resources will be used to update particles whose contribution to the estimated position of the object being tracked is almost zero. Re-sampling the distribution of particles is one of the methods to avoid this problem. Re-sampling is essentially a method of treating the current proposal distribution (after refinement through observation) as an empirical distribution and sampling from it repeatedly until a new population of particles is obtained. By taking into account the weights of the particles when re-sampling more likely observations are often included in the new distribution many times while un-likely ones will generally be excluded.

By giving some type of movement or diffusion property to the particles being used, such as the brownian (random) method used in (Bererton, 2004), those particles not currently observable by the tracking entity can be used to cover the whole range of

movement by the target. The possibility exists for adding more complex movement and behavior models to the particles, making them more likely to accurately reflect target actions. Controlling the number of particles can also be used to increase or decrease the effectiveness of the agent employing them to estimate the state of the environment or to react to changing computational requirements needed to run the rest of the game in question.

4. Density Estimation through Parzen-Windows

The Parzen-windows technique is an approach to estimating the density of a random variable by examining the data provided by a number of samples. This data is extrapolated to represent the entire distribution and can then be used to estimate the probability of a given point or measure being from the distribution in question.

(Duda *et al.*, 2001) provides an overview of the Parzen-windows technique. This overview begins with the description of a simple window function. Assume that a large number of samples from a distribution are available, and that the region of interest is a *d*-dimensional hypercube. The length of one side of a the hypercube is h_n, and the hypercube will have a volume $V_n = h_n^d$. In order to determine if a given sample falls within the hypercube, it can be tested using the following simple *window function*:

$$\varphi(u) = \begin{cases} 1 & |u_j| \le \frac{1}{2}; \quad j = 1,...,d \\ 0 & \text{otherwise.} \end{cases} \tag{0.14}$$

This window function $\varphi(u)$ represents a hypercube with $h_n = 1$ centered at the origin. If this hypercube is centered at *x* as opposed to the origin, then the number of samples x_i which fall inside this hypercube is given by

$$k_n = \sum_{i=1}^{n} \delta\left(\frac{x - x_i}{h_n}\right) \tag{0.15}$$

The probability that a test point *x* came from the distribution represented by a large number of samples can be determined using the following equation

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{x - x_i}{h_n}\right) \tag{0.16}$$

The power of this method is that window functions need not be as simplistic as the hypercube presented in the example. In order to make $p_n(x)$ a valid density function, it is sufficient to require that a given window function satisfies the following two constraints:

$$\varphi(x) \geq 0 \tag{0.17}$$

$$\int \varphi(u)du = 1 \tag{0.18}$$

In addition to a valid window function, density estimates obtained using these techniques are affected by the *window width* used in the approximation. Window width refers to the area or volume being tested in a test of an individual sample. In the simple hypercube example, the window width is the volume of the hypercube. Larger volume hypercubes increase the chance that an individual test will fall within the area of that sample's effect.

In more general terms, the window width, h_n, has an effect on $p_n(x)$ to the effect that it changes how "smooth" or inclusive the estimate will be. The effect of different window widths on the density approximations obtained using Parzen-windows is shown below:

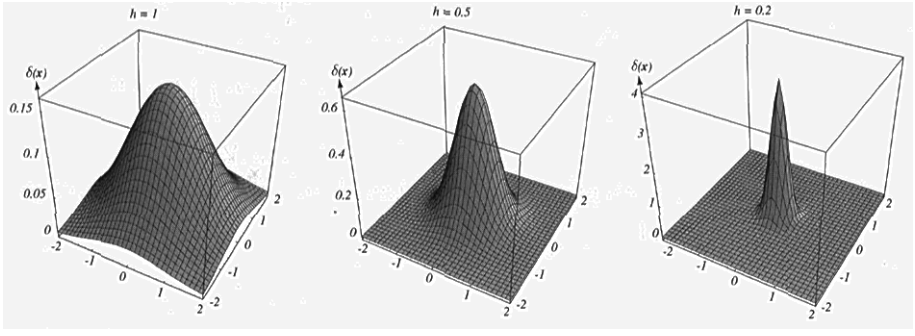


Figure 9. Effect of Window Width on Parzen-Windows (From Duda *et al.* 2001)

The images above are density estimates of a two-dimensional circularly symmetric normal distribution with different window widths. The larger window widths

used towards the left result in smoother approximations, while the smaller window widths to the right result in noisier estimates.

As was mentioned above, the power of the Parzen-windows approach is that different window functions can be used to test individual samples. A common window function used to estimate pattern densities is the standard normal probability density function. This window function takes the form:

$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}} \tag{0.19}$$

With a window width computed based on a predetermined constant and the number of samples, $h_n = h_1 / \sqrt{n}$, $p_n(x)$ takes the form of an average of normal densities centered at the samples:

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_n} \varphi\left(\frac{x - x_i}{h_n} \right) \tag{0.20}$$

Parzen-windows estimates of the standard normal distribution with varying window widths (h_1) and numbers of samples (n) are shown below:

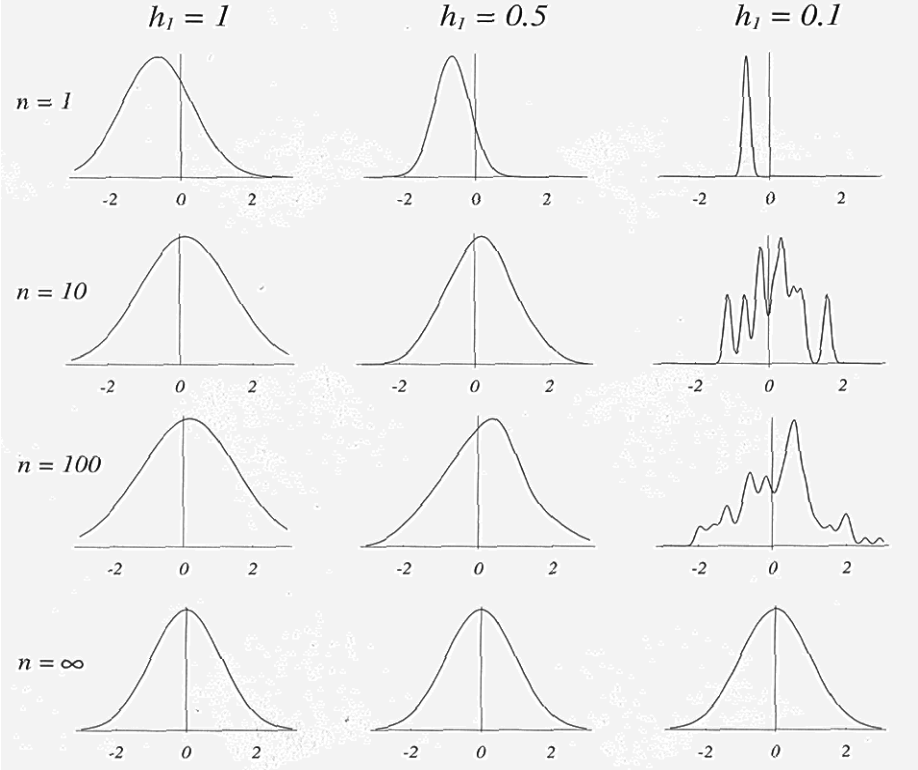


Figure 10. Parzen-Window Estimates of Standard Normal Distribution (From Duda *et al.* 2001)

The Parzen-windows approach is not limited to estimating the densities of smooth functions such as the standard normal shown above. The following figure shows Parzen-windows estimates of a bimodal distribution containing a triangle and uniform distribution. The window function used in this example is identical to that used to estimate the standard normal distribution that is a zero-mean, unit-variance, univariate normal density. While small sample sizes do not result in very accurate estimations, larger sample sizes begin to resemble the true density function.

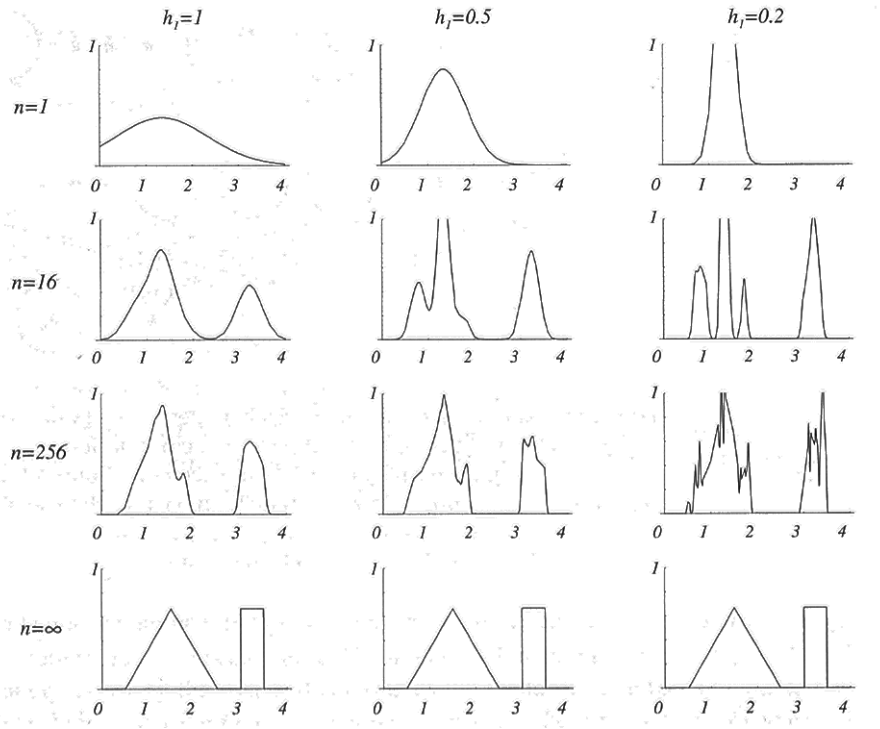


Figure 11. Parzen-Windows Estimates of a Bimodal Distribution (From Duda *et al.* 2001)

Due to the ability of Parzen-windows to approximate the densities of irregular distributions, it will be used several times in this work to visualize the state of the particle filters being used to track targets.

B. USING DISTRIBUTION REPRESENTATIONS

While the ability to accurately model uncertainty distributions is important, the most accurate representation of an opposing entity's state is useless unless the capability exists to act on that state. Decision making and communication with regard to agents in virtual environments is a large field, and as such cannot be reviewed as a whole in any detail in this work. There are two small subsets of those fields, path finding and collaborative tracking, which are instructive to review so that the utility of the particle filter tracking technique can be demonstrated.

Although this work is focused on determining the feasibility of using particle filters to track targets in a simulation environment rather than actually apply its use, some limited forms of decision making were included in the implementation. This was to verify that the particle filters being employed could be used to make simple decisions

regardless of the different nature of the tracks in this environment. The decision making techniques employed were limited to a path finding algorithm based on the A* algorithm. As such, that technique is briefly reviewed below.

In any military simulation, the ability of entities to communicate relevant information to other entities is an issue of great importance. In order for particle filter tracking to be used in simulations, it must be possible to communicate tracking information to other entities using the tracking information at hand. In (Klaas *et al.* 2005) a communication technique based on particle filters is proposed. Although communication between agents is not implemented in this work, a review of this technique is provided to demonstrate that particle filters can be used to facilitate communication between agents in a simulation environment.

1. A* Search

The planning of a path for an autonomous agent from one location to another in a virtual environment is a common artificial intelligence problem. Common problems involved in path planning include the avoidance of obstacles, the utilization of different types of terrain, and the avoidance of threats. One of the most common search techniques which can be used to solve these path planning problems is the A^* search. In (Stout 2000) a general overview of using the A^* search for path planning is provided.

The A^* algorithm searches a state space for the least costly path from a given starting state to a goal state. It accomplishes this by examining the neighboring or adjacent states of a given state. In path planning these states equate to locations in the environment and an adjacent state is reached by movement of the agent into the adjacent space. As the A^* algorithm runs in a path planning problem, it repeatedly examines the most promising unexplored location of which it is currently aware. When a location is explored, if that location is the goal of the path being planned, the algorithm will halt; otherwise it will record that location's neighbors for further exploration.

In order to determine which locations have been explored and which remain unexplored, A^* keeps track of two lists of states called **Open** and **Closed**. The **Open** list keeps track of unexplored locations and the **Closed** list keeps track of explored locations. In each iteration of the algorithm, A^* removes the most promising location from the **Open**

list for examination. If the location is not the goal location, the neighboring locations to the newly pulled state are sorted. If any of these locations are new, they are placed on the **Open** list. If any of these locations are already on the **Open** list, their respective state information is updated if the current path has a cheaper cost than that already recorded. If any of these locations are on the **Closed** list, they are ignored as they have already been explored. If during the course of the algorithm the **Open** list becomes empty before the goal location is found, there is no path to the goal from the start location.

The most promising location on the **Open** list is determined using an estimated path cost. This cost consists of two elements: the already incurred cost to reach that location and the estimated remaining cost from that location to the goal. While the cost incurred to reach the location currently being examined can be relatively easy to calculate, the remaining cost to the goal must be estimated through the use of a *heuristic*, or a set of loosely defined rules. It is in the definition of a useful heuristic that the efficiency of an A^* search can be most effected. A common heuristic used for simple path planning is the straight line distance from the location being examined to the goal location.

One of the reasons that A^* is so common is that it has several useful properties. The first is that if a path exists from the start location to the goal then A^* will find a path. The second property is that if the remaining cost estimate is always an underestimate of the actual remaining cost to the goal, then A^* will find the optimal path from the start location to the goal. The third property is that A^* is the most efficient search method to use a given heuristic. No search method that uses the same estimate heuristic will find an optimal path by examining fewer states than A^*.

As was stated above, the states in a path search are usually different locations in the environment. Determining which locations to consider in the path search is a far from trivial matter. While some environments contain a "natural" set of locations, such as an underlying square or hexagonal grid, many environments, particularly three dimensional environments, do not. A variety of methods for partitioning spaces into searchable nodes exist, and in the end there is no right answer. A particular partitioning technique must be chosen which complements the environment meets the needs of the programmer.

Although cost functions can be very simple, the utility of A* can be greatly enhanced by including more extensive estimates of incurred and remaining cost. In (van der Sterren 2002) cost functions are proposed which take into account tactical concerns such as cover present at given locations and exposure to enemy lines-of-fire. While more extensive cost functions can result in better path-finding behavior from agents, this improvement must be counterbalanced with the increased computational costs of performing an extended search. With intelligent state partitioning and cost functions, A* can be used to plan paths in almost any situation.

2. Particle Based Communication

In (Klaas *et al.*, 2005) a method for communicating target localization information based on particle filters is proposed. The technique makes use of a predictive density that is a mixture of the predictions of individual agents. The primary difference between this technique and the individual track technique proposed in (Bererton 2004) is that the master predictive density incorporates observations (weighted particles) into the whole by adding a weighting factor to individual agent predictions. Klaas's predictive density is represented by the following equation:

$$p(x_t \mid z_{1:t-1}) = \sum_{m}^{n_a} \pi_m \int p(x_t \mid x_{t-1}) p_m(x_{t-1} \mid z_{1:t-1}) dx_{t-1} \tag{0.21}$$

In the above equation, π_m is the weighting coefficient for each agent *m*, $p(x_t \mid x_{t-1})$ is the prediction for the current time-step from the previous prediction density, and $p_m(x_{t-1} \mid z_{1:t-1})$ is the predictive density for the individual prediction for each agent *m*. By replacing the analytical portion of the above equation with a particle filter approximation, the predictive density becomes:

$$p\left(x_t \mid z_{1:t-1}\right) = \sum_{m}^{n_a} \pi_m \sum_{i}^{N} w_{t-1,m}^{(i)} p\left(x_t \mid x_{t-1,m}^{(i)}\right) \tag{0.22}$$

The particle approximation above contains the same weights for individual observers (π_m) with each individual prediction taking the form of a weighted mean of that individual's particles. The basic particle filter update algorithm for this method is very similar to that provided in (Bererton 2004) with the exception that no re-sampling

step is required. This is due to the fact that the predictive density is a mixture of particles from several agents, all of which resample their individual distributions as needed.

The predictive density technique as described so far assumes that the particles from individual agents are available at every time step, indicating constant communication among agents. Sporadic communication can be accomplished by changing the weighting factors (π_m) when a specific agent is not communicating during a given time-step. The size of messages (sets of particles) passed between agents can also be limited to either save computation or impose realistic communications constraints on the process. If the size of the communication allowed is less than the number of particles contained in a filter, the agent can sample their own particle filter the required number of times and send the resultant particles as their communication.

If sporadic communication is allowed, some manner for determining when to send messages to other agents must be devised. Klaas facilitates sporadic communication by setting a threshold, τ, which represents the likelihood of all observations since time *t'*. When the likelihood drops below this threshold, communication will occur. This likelihood can be easily computed by examining an individual agent's particle filter, specifically the un-normalized weights of the particles.

Recall from the discussion of particle filters above that after a short amount of time a small number of particles will have large weights while the majority of particles will have negligible weight. As this begins to occur, the average un-normalized weight of the particles will drop, indicating that the particle filter is indicating a new "very likely" position for the target being tracked. It is precisely for this reason that particle filters renormalize the weights of their particles and resample the distribution as parts of the update algorithm. By saving the sum of the un-normalized weights of the particles each time through the algorithm, individual agents can determine when their picture of the environment has changed sufficiently enough to warrant a communication to the other agents.

The procedure used by Klaas to trigger sporadic communication is shown below:

1. Save un-normalized weights in particle filtering algorithm: $W_{t,a}^{\Sigma} = \sum_i \tilde{w}_{t,a}^{(i)}$

2. Update likelihood for time *t*: $L_{t,a} = L_{t,a} \cdot W_{t,a}^{\Sigma}$

3. If $L_{t,a} < \tau$, trigger communication, reset $L_{t,a} = 1$.

Although the theoretical model provided by Klaas uses identical weights for the observations of all the individual agents, Klaas mentions that this need not be the case. A mixture of weighting techniques such as allowing individual agents to value their own observations more heavily than others' could result in a much more realistic (although not as accurate) distributed tracking state.

Klaas provides several examples showing particle based communication among several agents. The image below shows one of these examples in which a group of three agents are attempting to locate an opposing agent in a jungle:

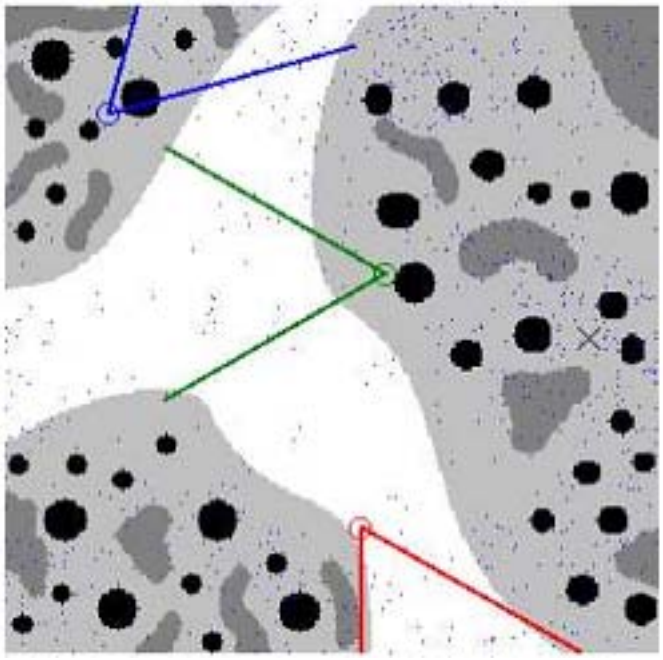


Figure 12. Coordinated Particle Filter Search (From Klaas *et al.*, 2005)

In the figure above, the three agents are searching for a fourth agent represented by a black "X." The environment features varying levels of occlusion which reduce the probability of detecting the target. The searching agents have already determined that the target is not within the high-visibility areas of the environment and have shared this information with each other, resulting in very few particles in those regions. The gray

areas, with reduced probability of detection, have a correspondingly higher concentration of particles. A smoothed probability density corresponding to the situation above is shown below:

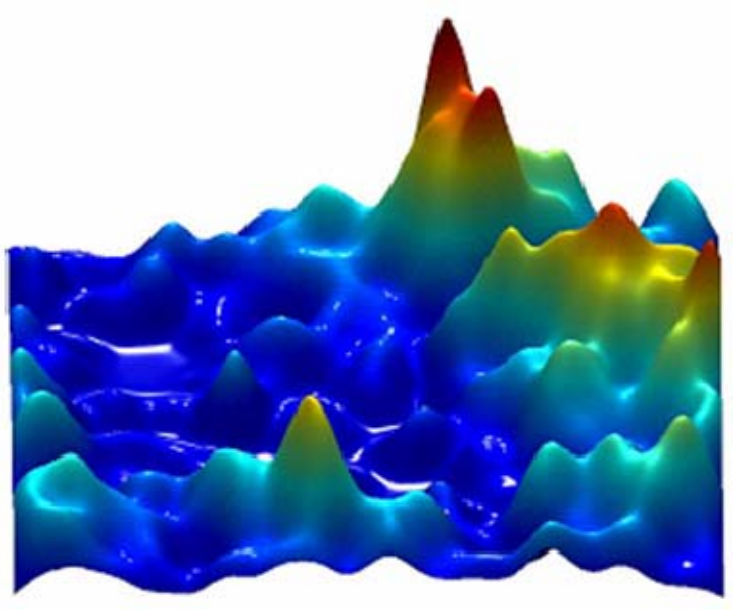


Figure 13. Probability Density Corresponding to Jungle Search (From Klaas *et al*., 2005)

III. MODEL

A. ENVIRONMENT

The typical environment in a military simulation differs greatly from the game environments for which particle based tracking was first proposed. While the test environment used in this work is relatively simple, its characteristics are sufficiently different from typical game environments to warrant a slightly different particle tracking technique than that proposed in (Bererton 2004). The characteristics of this environment are discussed below.

1. Characteristics

The environment chosen for use in this work closely resembles an "open ocean" environment. The environment contains a large amount of space compared to many game environments, and it is for all practical purposes devoid of obstructions. By limiting the agents in this test model to those resembling ships, the detection and tracking process can be limited to one that is essentially two dimensional in nature. Additionally, organic sensors used to detect other platforms will have comparable ranges due to the curvature of the Earth. With a large amount of space and limited active detection ranges, the majority of detections occurring within the simulation will be passive in nature. Theses detections will be largely uncertain, and the particle filter technique's ability to accurately track other platforms through passive means will be readily evident.

Ship-like platforms moving through the open ocean also have a very low speed in comparison to amount of space in the environment. This results in placing increased importance on determining the heading and speed of opposing targets. In order to effectively maneuver to force contact with a platform with similar capabilities, agents in this environment must be able to effectively estimate other agents' headings and speeds from their particle filter tracks on these agents. Simply moving towards the mean of the particle filter representation of another platform's position would be an unattractive method of searching, as it would most likely result in a "tail chase" with little possibility of acquiring a firm track on the target.

In order to focus on the utility of particle filter tracking techniques all platforms in this environment are capable of completely disambiguating emissions from other platforms. In other words, if a platform passively detects several other platforms in the course of a simulation, it will be able to correlate these detections exactly with their corresponding platforms. While there are certain passive sensors which are capable of this level of sensor disambiguation, such as sonar, most do not have this capability. As track correlation is a rather large and complicated field of study, track disambiguation is left out of this initial work. While agents in this environment can exactly correlate detections to corresponding entities, they have no prior knowledge of the locations or states of other platforms. Indeed, as will become clear when the particle filter tracking technique is fleshed out below, they will never have complete knowledge of other platforms, and will be forced to deduce this information from the state of their particle based tracks.

There are no weapons represented in this environment. Individual agents move through the environment for the sole purpose of sensing and tracking other platforms. The addition of engagement capabilities would have added tactical implications to the act of sensing and tracking, and thus are beyond the scope of this work.

2. Implementation

A simple simulation environment was created in Java utilizing the Simkit package (Buss 2002). Simkit is a library which supports the creation of component based discrete event simulation models. In discrete event simulations time does not advance in so called time-steps. Instead, simulation time is immediately advanced to the time of the next occurring event in the simulation. When an event is processed, corresponding state variables within the simulation are altered, further events are scheduled or canceled as appropriate, and time is advanced to the next scheduled event. The display window which shows the state of the simulation is a Java2D display. The size of the simulation environment was variable, but the display was always partitioned into 10x10 grid squares to provide a visual reference of position. Each unit of distance in the simulation roughly corresponded to one nautical mile. Thus the empty environment pictured below consists of a 100nm x 100nm area.

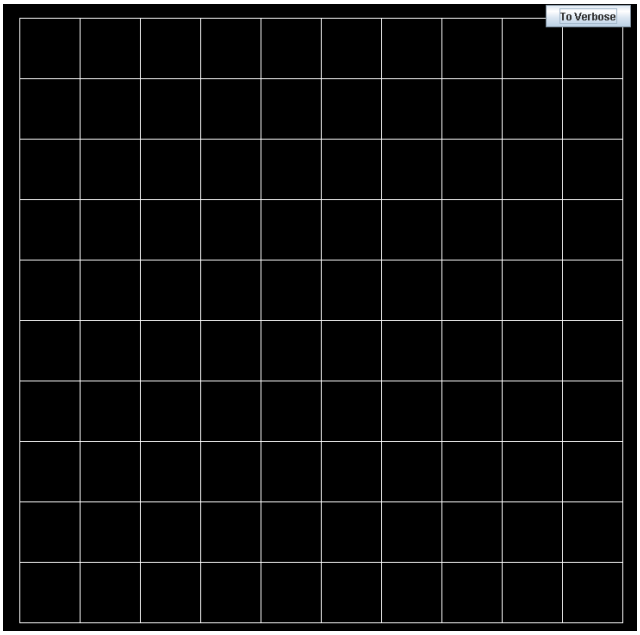


Figure 14. Empty 100 by 100 Simulation Environment

Two different *sides* consisting of any number entities were represented in the simulation. Each side was represented in the visual display using a simple icon and identifying color. As the two colors chosen were blue and red, through the rest of the work entities on corresponding sides will be referred to using this color (e.g. red's track of blue). The figure below shows the appearance of a base blue entity on the left and a corresponding red entity on the right.

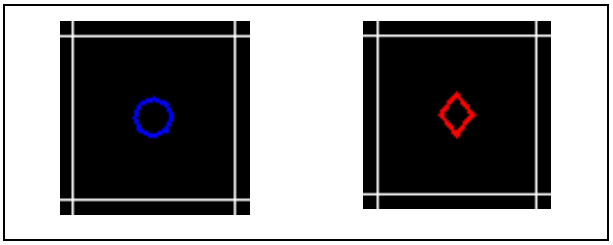


Figure 15. Appearance of Base Blue and Red Entities

The course (or heading) of entities in this environment are given in degrees. A heading of 0/360 degrees corresponds to "up" or due north on the screen and headings proceed clockwise in the manner of compasses or gyro repeaters. One simulation time unit was picked to roughly correspond to one minute, and entity speeds are given in distance traveled in one simulation hour (60 simulation minutes). Both of these

conventions are used to draw speed leaders on platforms in the environment. These speed leaders are simply lines drawn from the center of an entity's icon extending in the direction in which they are headed. The length of the leader is determined by the current speed of the entity. The end of the leader shows where a platform will be located in 15 simulation minutes if it maintains its current course and speed. The figure below shows four different entities with different courses and speeds.

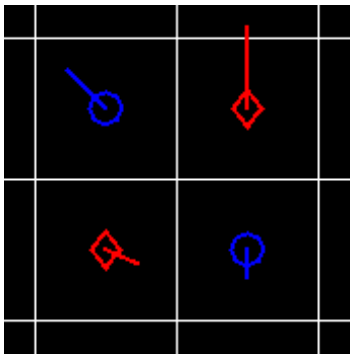


Figure 16. Speed Leaders Representing Various Courses and Speeds

All platforms in the simulation were equipped with identical sensor suites. These sensors consisted of a continually operational passive sensor of relatively short range (visual), an intermittently operable active sensor of longer range (radar), and an additional passive sensor capable of detecting emissions of other agents' radar type sensor. Each sensor type had an associated range and bearing ambiguity for the purpose of turning detections into particle based tracks. Although the particle filter tracking techniques employed varied as described in further sections, the characteristics of the sensors were constant throughout the development of the tracking methods and so are detailed below.

Sensor	Max Range	Counter Detection Range	Range Ambiguity	Bearing Ambiguity	Sweep Time
Visual	10.0	N/A	0.3	±1.5°	0.2
Radar Detector	60.0	N/A	0.8	±5.0°	N/A
Radar	30.0	60.0	0.05	±1.0°	0.02

Table 1. Simulation Sensor Parameters

Sensors are *cookie cutter* sensors as described in (Buss 2005). That is, if a sensor is enabled and there is another platform within the detection or counter detection range of that sensor, than detection will occur as a result. The range and bearing ambiguity parameters will be discussed in further detail in following sections. The *sweep time* refers to how long it takes a given sensor to complete a 360 degree sweep of the environment. While a sensor is operating it attempts to detect other platforms in every sweep time interval. For the radar sensor this can be equated to the time it takes for the antenna to rotate one time (1.2 seconds this simulation) and for the visual sensor the amount of time it takes a lookout to identify and report a contact (12 seconds). Note that the radar detector does not possess a sweep time, as it is continually "listening" for other entities' radar emissions. All of the values above were chosen arbitrarily and do not attempt to represent existing sensor systems with a high level of fidelity.

When a sensor is operating it is drawn as a circle with the proper radius centered at the location of its owning platform. The visual sensor is drawn as a light blue circle, radar as a yellow circle, and the radar detector as a dashed red circle. A figure showing the appearance of operating sensors is shown below. The blue platform is operating its visual and radar detecting sensors. The red platform is operating those same sensors with the addition of its radar.

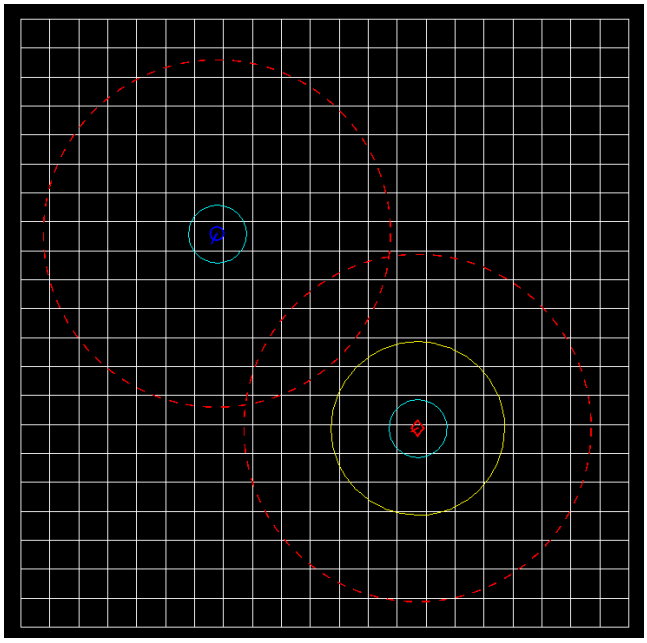


Figure 17. Simulation Sensor Arc Appearance

When a detection occurs within the environment, a particle-based track will be created and maintained using the methods described in the sections below. A track corresponds to an entity's estimation of the location, heading, and speed of another entity in the environment based on a series of detections. This track can be drawn on the representation of the simulation environment. The particles constituting this track are drawn in a lighter color of the platform which owns the track. A blue track of a red entity will be drawn using cyan particles and a red track of a blue entity will be drawn using orange particles. The tracks are used to create estimated positions, headings, and speeds of opposing platforms. These estimated positions are drawn as a white square with an associated speed leader. The figure below shows the appearance of a blue track (of a red platform) which has been acquired using its radar detector with associated estimated position.

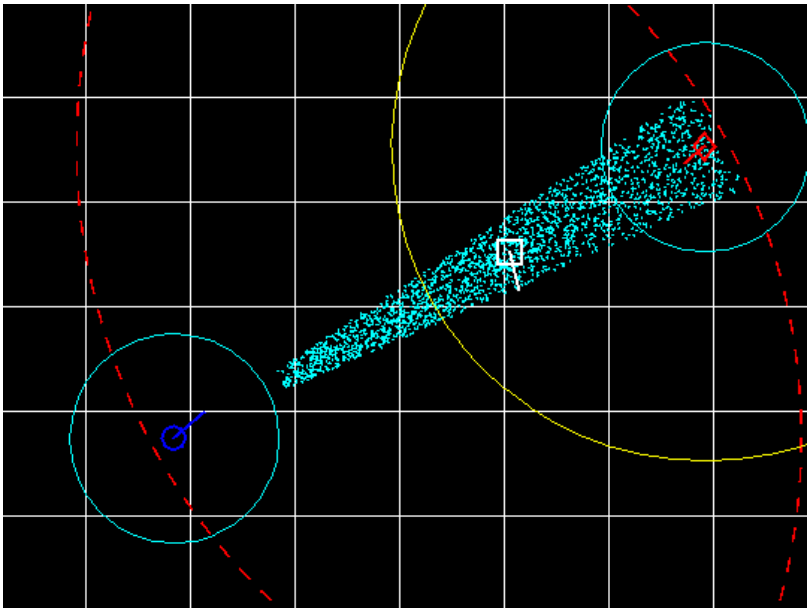


Figure 18. Basic Track Appearance

Platforms move through the environment using a waypoint system. Waypoints can represent either a discrete location in the simulation environment or an area of the environment. Patrol plans consisting of a series of waypoints can be transited either in a set or pseudo-random order. An area being used as a patrol plan is transited by randomly selecting points uniformly distributed across the area either once or multiple times depending on the context in which it is used. Patrol plans are also mutable, and a rudimentary path-finding system based on the particle tracks an entity holds is described in detail in a later section. When patrol plans are displayed in the visualization of the simulation, waypoints are displayed as either a green circle or box (depending on if the waypoint is a location or an area) with a series of green lines connecting waypoints in the order in which they will be traversed. Course changes happen instantaneously when an entity reaches a given waypoint and heads towards another. The figure below shows the appearance of patrol plans when they are drawn. The blue entity will traverse a series of four points and the red entity will patrol the area represented by the green box.

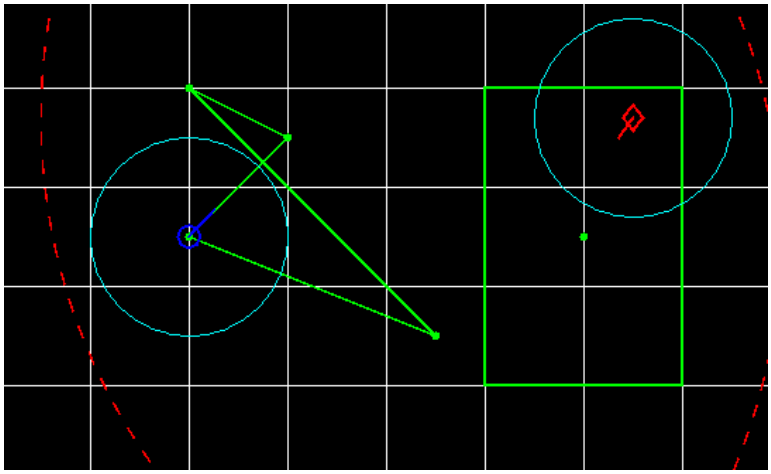


Figure 19. Patrol Plan Appearance

Platforms operate their radar-type sensors according to a radiation plan. For this work those plans were relatively simple. They consist of a simulation start time for the first operation of the radar, the length of time to operate the radar, and the interval of time to remain silent in between radar operation periods. While the radar is operating, it will continually sweep the area of the environment within its range for opposing targets. Due to the small sweep time of the radar-type sensor, even a relatively short period of radiation will result in many possible detections or counter detections.

All entities in the environment are registered with a sensor mediator as described in (Buss 2005). When sensor sweep events are pulled off the event queue, the entity which scheduled the event forwards its current position and sensor information to the mediator for processing. The mediator uses this information to determine the number and type of detection events which will occur and will create new tracks or update existing tracks based on the situation. These new or updated tracks are then returned to their corresponding owners for further use. The detection events managed by the sensor mediator drive the creation and maintenance of the particle based tracks owned by platforms in the simulation. The nature of these tracks is described in detail below.

B. PARTICLE TRACKING TECHNIQUE

The particle tracking technique described below attempts to provide accurate position, heading, and speed information about another platform to its owning entity. Due to the nature of the simulation environment, it must be able to accomplish this goal

for large numbers of relatively exact active detections, a small number of highly uncertain passive detections, and everything in between.

In order to handle a wide variety of detections, an intelligent method must be devised for creating initial distributions of particles when another platform is detected for the first time. The technique proposed in this research is different from that used in (Bererton 2004) and is detailed in the Track Creation section below. Additionally, there must be a means to update distribution of particles in a track based on updated detection information or disqualification of a previous distribution. The methods used in this work are detailed in the Track Maintenance section.

Following the creation of methods to enable the two previous requirements, it became evident that including a small amount of intelligence in individual particles could be used to increase the utility and accuracy of the particle based track in certain situations. This lead to the development of several simple particle based behaviors which demonstrate the possible usefulness of this idea. These changes to the base particle filter are described in the Contextual Particles section.

1. Track Creation

Tracks are created when the sensor mediator processes a valid detection event of another platform for the first time by a given entity. Note that a detection can take the form of an accurate radar or visual detection or an uncertain counter-detection of another entity's radar via a radar detector. Due to the large distances and relatively short range of platforms' sensors in this simulation, the majority of detection events encountered were of the later variety. However, the temptation to optimize this tracking method to handle these types of detections was avoided in order to ensure that it would be able to effectively represent all manner of detections.

The sensor mediator creates a new track in two steps. The first step is the creation of a detection distribution. The method used to create this distribution is generic and can therefore be used for both active and passive detections. The second step is to create a large number of particles which take the form of a large number of weighted samples from this distribution. Once these two steps are complete the resultant collection of particles is forwarded to the corresponding entity for use. Even without further

refinement, this new collection of particles can be used to create a rough estimated position of the tracked platform. An in depth review of initial detection distributions, distribution sampling, and estimated position creation is provided below.

a. Initial Detection Distributions

In (Bererton 2004), an initial distribution of particles distributed randomly throughout the environment is used. This allows an entity to start with no knowledge regarding the whereabouts of the target it is attempting to localize. Bererton also proposed that some prior knowledge of target location could be used to create an initial distribution. This second proposal for an initial distribution makes more sense for a military simulation due to the size of the environment. While this test-bed simulation starts with no prior distribution of possible target locations in the environment, including this feature in an actual simulation could be accomplished with little difficulty.

This simulation creates an initial distribution based on the first detection event of a given entity for the detection platform. In this environment these distributions have three components: a location component, a heading component, and a speed component. Of these three elements, the location component is the most troublesome to create. Due to the vast range of detection types which could occur in a military simulation, a generic method for representing location distributions is presented which allows detection events to accurately model the level of uncertainty inherent in both active and passive detections.

The generic location component as proposed here requires three parameters. These are: the actual bearing and range of the detected entity, the bearing ambiguity of the detection, and the range ambiguity of the detection. The figure below shows these three parameters.

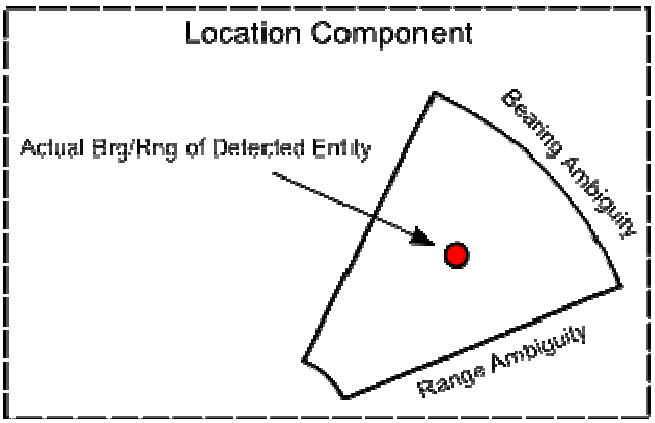


Figure 20. Location Component of Initial Detection Distribution

The actual bearing and range of the target is readily available to the sensor mediator which contains links to all the entities in the simulation. The range and bearing ambiguities are dependent on the sensors involved in the detection. As every entity contains information regarding the capabilities of its sensor suite, this information is also available for the construction of the location distribution. These pieces of information are used by the mediator to calculate the minimum and maximum ranges and bearings of the location distribution that will result from the detection. These four values are illustrated below.

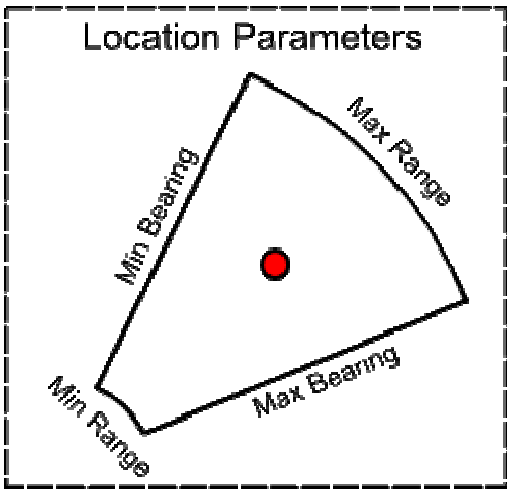


Figure 21. Location Distribution Parameters

In this simulation the sensor characteristics are as shown below:

Sensor	Max Range	Counter Detection Range	Range Ambiguity	Bearing Ambiguity	Sweep Time
Visual	10.0	N/A	0.3	±1.5°	0.2
Radar Detector	60.0	N/A	0.8	±5.0°	N/A
Radar	30.0	60.0	0.05	±1.0°	0.02

Table 2. Simulation Sensor Parameters

The minimum and maximum bearings are readily calculated using the provided bearing ambiguity and the actual bearing of the target at the time of the detection. Minimum and maximum ranges are calculated using the actual range and the range ambiguity of the corresponding sensor. Range ambiguity is represented as perfect (no error) with a value of 0.0 up to none (no range information whatsoever) with a value of 1.0. Thus the calculations carried out to define a location distribution in this simulation are as shown below.

Calculate bearings ($Brg_{\min,\max}$) based on actual bearing (Brg_{act}) and sensor ambiguity (Amb_{brg}):

$$Brg_{\min} = Brg_{act} - Amb_{brg} \tag{0.23}$$

$$Brg_{\max} = Brg_{act} + Amb_{brg} \tag{0.24}$$

Calculate ranges ($Rng_{\min,\max}$) based on actual range (Rng_{act}), sensor ambiguity (Amb_{rng}), and range of the detecting sensor (Rng_{sensor}):

$$Rng_{\min} = \max\left(Rng_{act} - Rng_{act}Amb_{rng}, 0\right) \tag{0.25}$$

$$Rng_{\max} = \min\left(Rng_{act} + Rng_{act}Amb_{rng}, Rng_{sensor}\right) \tag{0.26}$$

Note that in this simulation all location distributions will have their bearings centered about the actual bearing of the target. More complicated sensor models

could easily change this, as the only important result is that the location distribution has two boundaries on bearing. Regardless of the range ambiguity of the detection sensor, the range boundaries will never result in possible locations "behind" the detector or out of range of the detecting sensor. More complicated range calculations could also be used to determine these boundaries, such as bounding the lower range not at the location of the detector but at the range of other operating sensors with lower detection ranges. The figure below shows the effect of differing bearing and range ambiguities on the location distributions that the sensor mediator will construct. In (a) a distribution with small bearing and range ambiguities (relative to the other examples) is shown. In (b) and (c) distributions with good bearing/bad range and bad bearing/good range ambiguities are shown. In (d) a distribution with both high bearing and range ambiguities is shown.

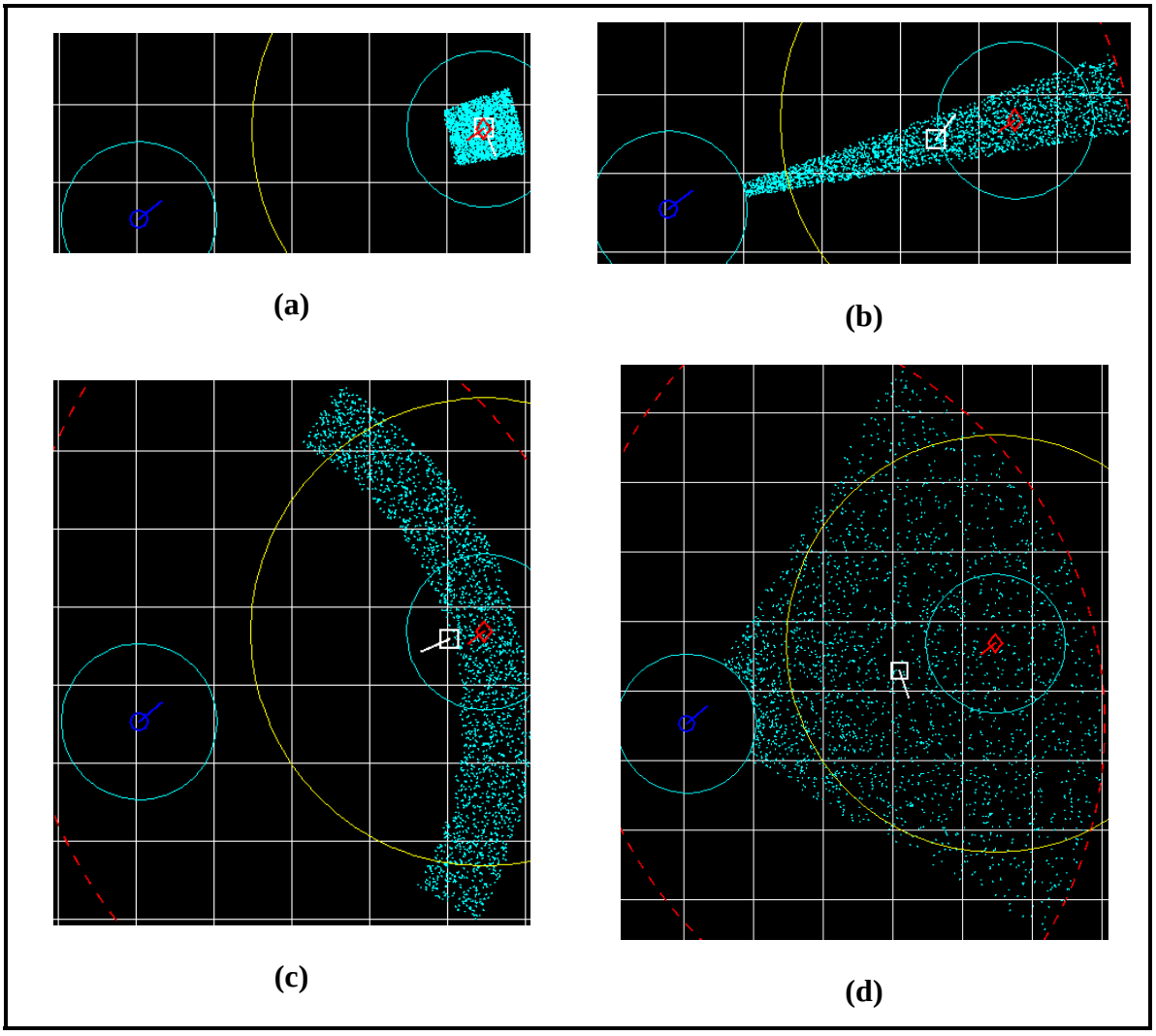


Figure 22. Location Distributions with Differing Bearing/Range Ambiguities

The other two components of an initial detection distribution are the heading and speed components. With no prior knowledge of target heading, as is the case in this simulation, the initial heading distribution encompasses all possible headings [0, 360]. The speed distribution likewise encompasses all possible speeds from standing still to the maximum speed of entities in this simulation [0, 32]. Given some prior knowledge of target intent or capabilities, the nature of these distributions could easily be changed. Although it is not the case in this simulation, it is possible to give entities incorrect information about both the movement and sensing capabilities and intents of opposing entities. As the sensor mediator constructs detection distributions according to the detecting entity's knowledge base, the detection distribution created would reflect

these misconceptions and provide inaccurate track information to the detecting platform. A figure showing the components of a complete detection distribution is provided below.

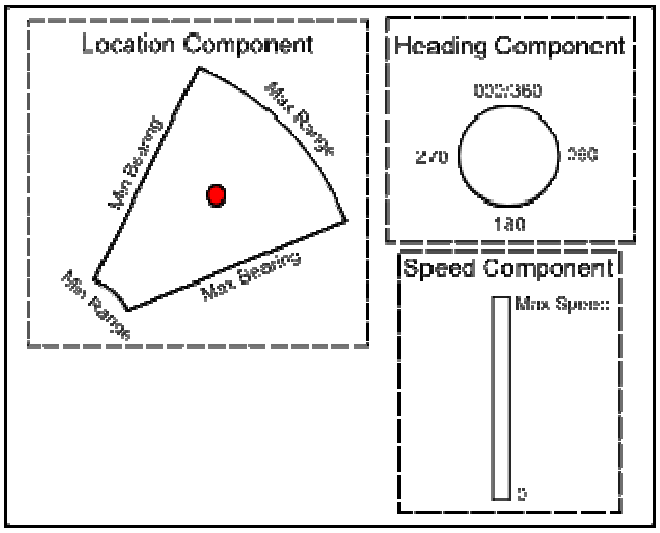


Figure 23. Components of an Initial Detection Distribution

b. Detection Distribution Sampling

Once the sensor mediator has constructed the distribution corresponding to a given initial detection, it will draw a large number of samples from that distribution for use as particles. Particles in (Bererton 2004) were simply weighted points in space. Due to the fact that position estimation was of primary interest, particles did not have a motion model, but were simply moved a random x and y distance at each time step. In order to extract heading and speed information from particle tracks in initial versions of particle tracks in this simulation, individual particles will have a heading and speed in addition to a location and weight. The particles created will be naïve of their surroundings. That is, they will maintain their course and speed until invalidated via one of the techniques described in later sections.

The parameters of a naïve particles and corresponding notation which will be used through the rest of this work are shown below:

Naïve Particle np_i				
Weight	*X* Location	*Y* Location	*X* Velocity	*Y* Velocity
w_i	x_i	y_i	vx_i	vy_i

Table 3. Naïve Particle Parameters

Creating a particle track from an initial detection distribution requires sampling from the distribution N_s times. Each sample requires four pseudo-random draws: two from the location distribution and one each from the heading and speed distributions. The results of these four draws along with the position of the detecting platform ($Pos_{x,y}$) are used to calculate the parameters of each particle. The process of sampling from the detection distribution to obtain a particle based track is shown on the next page.

Sample from Initial Detection Distribution

FOR $i = 1 : N_s$ ***Generate Initial Detection Distribution***

$Brg_i = U[Brg_{\min}, Brg_{\max}]$ ***Draw Bearing of Particle***

$Rng_i = U[Rng_{\min}, Rng_{\max}]$ ***Draw Range of Particle***

$Hdg_i = U[Hdg_{\min}, Hdg_{\max}]$ ***Draw Heading of Particle***

$Spd_i = U[Spd_{\min}, Spd_{\max}]$ ***Draw Speed of Particle***

$w_i = \frac{1}{N_s}$ ***Set Weight of Particle***

$posrad_i = \frac{\pi(Brg_i - 90)}{180}$ ***Change Bearing from Degrees to Radians***

$x_i = Pos_x + Rng_i * \cos(posrad_i)$ ***Set Particle *x* Position***

$y_i = Pos_y - Rng_i * \sin(posrad_i)$ ***Set Particle *y* Position***

$velrad_i = \frac{\pi(Hdg_i - 90)}{180}$ ***Change Heading from Degrees to Radians***

$vx_i = \frac{Spd_i * \cos(velrad)_i}{60}$ ***Set *x* Velocity Component***

$vy_i = \frac{-Spd_i * \sin(velrad)_i}{60}$ ***Set *y* Velocity Component***

END FOR

Figure 24. Sampling from a Detection Distribution to Create a New Track

In the above process, the subtraction of ninety degrees in the conversion of degrees to radians is due to the difference between the position of zero degrees in this simulation and in mathematical conventions. Due to speed represented in the simulation as distance traveled in one simulation hour, the division by sixty creates a unit vector pointing in the proper direction.

When the above sampling process is complete, N_s particles have been created in a roughly uniform manner from the detection distribution with equal weight. The four figures below show a new particle track containing 2,500 particles created from an initial detection distribution. The first figure shows the appearance of the new track in the visualization window. The following three figures show Parzen-windows estimates of the distribution density for the heading, speed, and position of an initial detection.

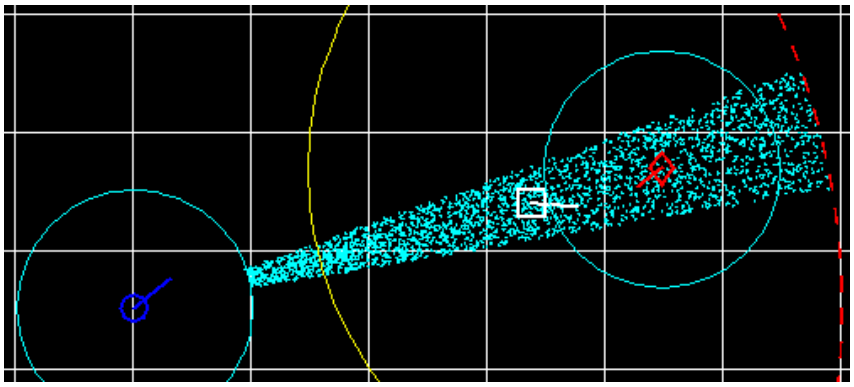


Figure 25. New Track Created by Sampling an Initial Detection Distribution

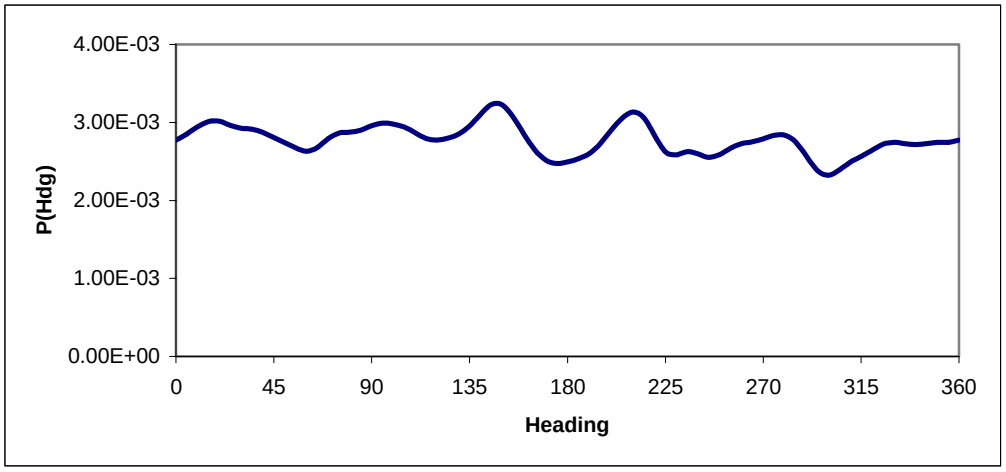


Figure 26. Parzen-Windows Approximation of Initial Detection Distribution Heading Density (Window Width 7.2)

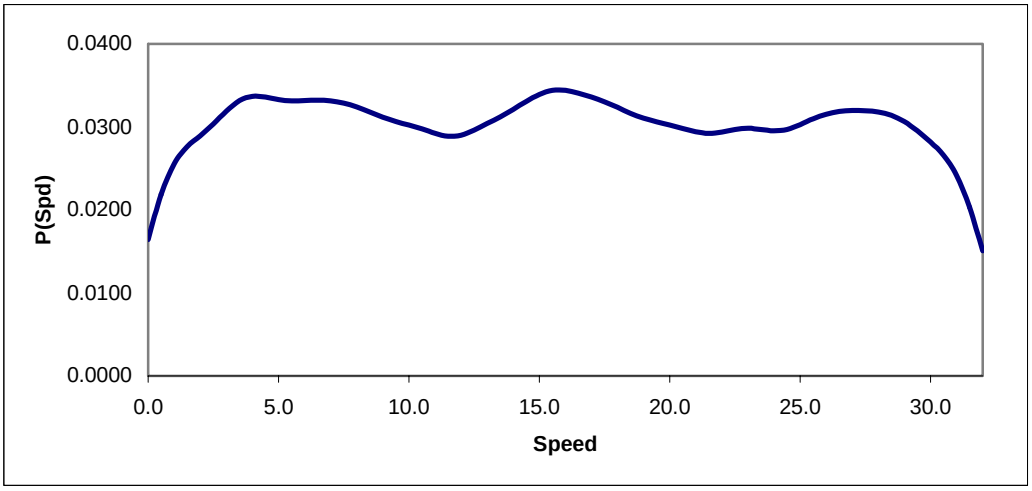


Figure 27. Parzen-Windows Approximation of Initial Detection Distribution Speed Density (Window Width 1.0)

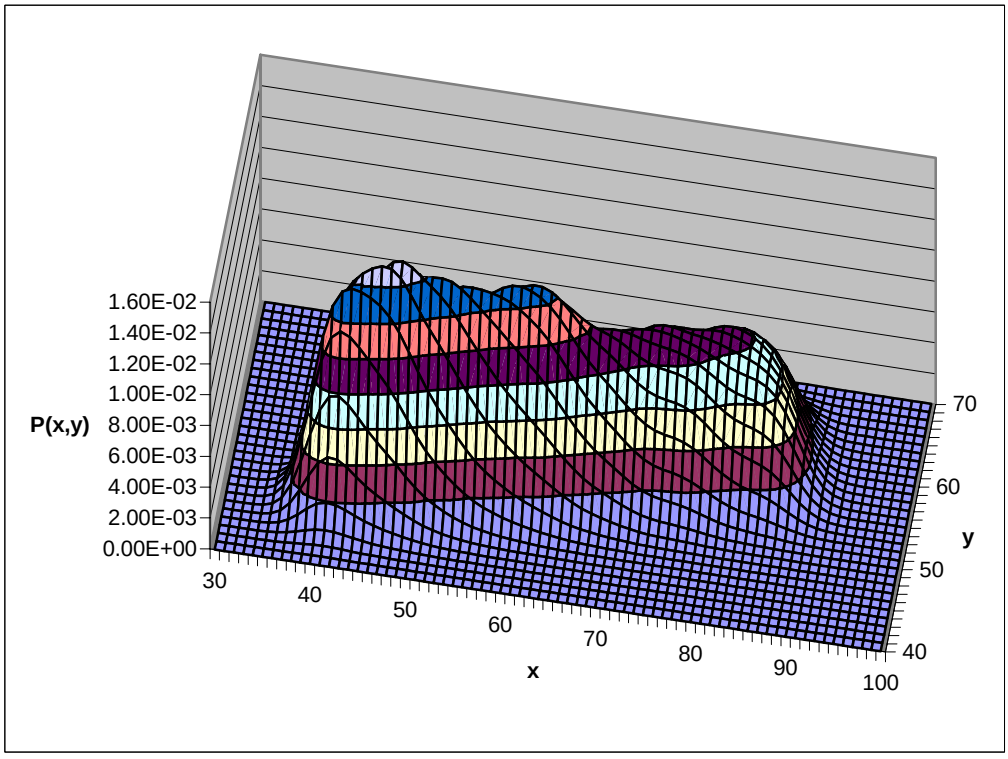


Figure 28. Parzen-Windows Approximation of Initial Detection Distribution Position Density (Window Width 2.0)

c. Estimated Position Calculation

Obtaining an estimated position from a particle track is a trivial process. The actual utility of using an estimated position versus treating the track as a collection of possible locations will be discussed in a later section. The estimated position at a given point in time can be calculated by computing the weighted average of the particles currently in the track. In order for this position to be valid, the sum of the weights of particles in the track must sum to one. An estimated position for a track containing N_s naïve particles is computed using the procedure shown below.

Estimated Position Calculation

FOR $i = 1 : N_s$ ***Compute Estimated Position of Entity being Tracked***

$x_{est} = x_{est} + w_i x_i$ ***Compute Estimated *x****

$y_{est} = y_{est} + w_i y_i$ ***Compute Estimated *y****

$vx_{est} = vx_{est} + w_i vx_i$ ***Compute Estimated *x* Velocity***

$vy_{est} = vy_{est} + w_i vy_i$ ***Compute Estimated *y* Velocity***

END FOR

Figure 29. Computing an Estimated Position from a Particle Track

Note that the above procedure computes an estimated *x* and *y* velocity. This can be converted to a [0, 360] heading and [0, *max speed*] speed using similar mathematical procedures to those shown in the initial detection distribution creation section. Additionally, if no particles have been added to or removed from the track using one or more of the methods described in future sections, then the estimated heading and speed will be unchanged and does not need to be computed.

2. Track Maintenance

Tracks constructed using the above detection distribution and sampling method adequately describe the initial areas of uncertainty resulting from the detection of new entities. Additionally, these tracks are immediately useful in providing rough estimated positions, albeit with heading and speed information of dubious utility. If these initial

tracks were used without modification throughout the simulation, the areas of uncertainty they represent would expand according to the member particles' heading and speed information until they encompassed the entire environment. In order to prevent this and to allow entities to continually refine their estimates of the positions, headings, and speeds of opposing platforms, methods were provided to *disqualify* groups of particles and *repopulate* the track with new groups of particles.

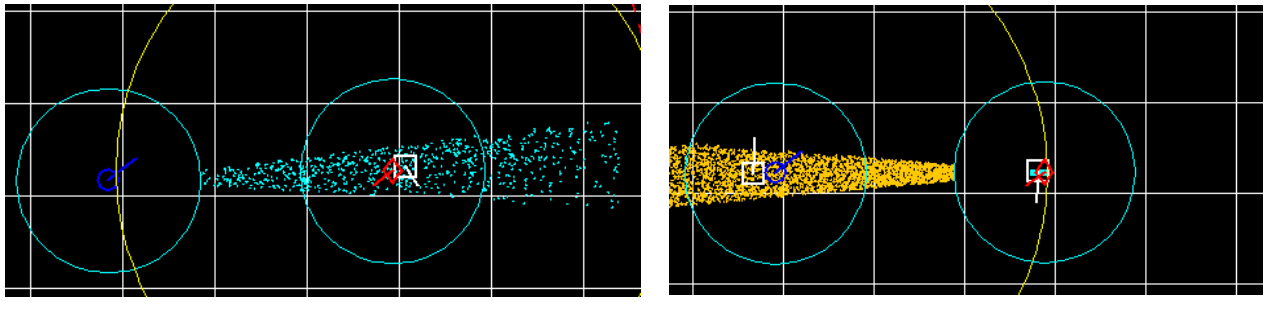
a. Particle Disqualification

In (Bererton 2004) individual particles were removed from the distribution by reducing the weight of observed particles (which did not result in an acquisition of the target) and then re-sampling the population. Particles with very small or zero weights would not be included in the re-sampled distribution as often and can thus be removed from consideration. In this simulation, rather than reducing the weights of particles which come under observation without an acquisition occurring, they are disqualified from consideration by removing them from the collection of particles currently contained in the track. The weights of particles which are still valid are renormalized so that the sum of weights in the particle track will remain one.

The disqualification of particles takes place through the observations of the tracking entity. As this simulation is a discrete event simulation, these observations take the form of sensor sweep events being processed by the sensor mediator. The actions taken when a sensor sweep results in the detection or counter-detection of an opposing entity for the first time were described in a preceding section. Once a track exists for a given target, further sensor sweep events may result in another detection event or a sanitization event.

Detection events occurring for a target which is already held in a track by the detecting entity will result in a disqualification of particles from the current track if those particles do not fall inside the area of the new detection. This area is essentially an abbreviated initial detection distribution consisting only of the position component. Once the sensor mediator has constructed the position distribution from the detecting entity's sensor information, it compares the new distribution to the particles in the detector's current track. Those particles which fall outside this distribution are disqualified and removed from consideration. Depending on the natures of the previous and current

detections, a large number of particles can be disqualified using this method. The figure below illustrates the disqualification of particles through new detection events. In (a), the blue platform has an inexact track of the red platform acquired via its radar detector. In (b), the blue platform has turned on its radar and achieved an active detection of the red platform. The majority of the particles in blue's track have been disqualified, leaving only those which were in the close vicinity of the new detection.



(a) (b)

Figure 30. Disqualification of Particles via Detection Events

Sensor sweep events can also result in the disqualification of particles from an existing track through sanitization of areas of the environment. Sanitization occurs when the area of regard of the tracker's sweeping sensor overlaps the position of one or more particles currently held in its track of another entity. In environments with low numbers of active detections, the ability to sanitize areas of the environment allows entities to make better use of passive or very time late detections which have spread over large portions of the environment. Note that in this simulation, as every entity continually operates its visual sensor, platforms are able to constantly sanitize the area of the environment falling within visual range. With a longer range, the radar sensor is a more effective sensor for sanitizing areas of the environment, but it carries with its use the possibility of counter detection by opposing platforms.

Disqualification of particles through sanitization is achieved in the same manner as through detection events with one exception. This exception is the nature of the abbreviated detection distribution against which to test particles. The position distribution used for sanitization has range parameters [0, *max sensor range*] and bearing

parameters [0, 360]. This equates to a position distribution consisting of the footprint of the sensor. This sanitization distribution is then compared to the particles residing in the entity's current track of another platform. Any particles following inside this distribution will be disqualified and removed from consideration.

The following figure illustrates the disqualification of particles through sanitization. In (a), the blue platform has been operating its radar, allowing red to track it passively. In (b), the blue platform has secured its radar and continued moving to the southwest. Red's passive track of blue has been expanding based on the headings and speeds of its member particles. In (c), the red platform has turned on its radar. Although the blue platform has moved out of red's radar range, all the particles in red's passive track of blue which fell inside the operating radar's footprint have been disqualified and removed from consideration. Notice the improvement in the red platform's estimated heading and speed for the blue platform obtained by disqualifying a large number of particles.

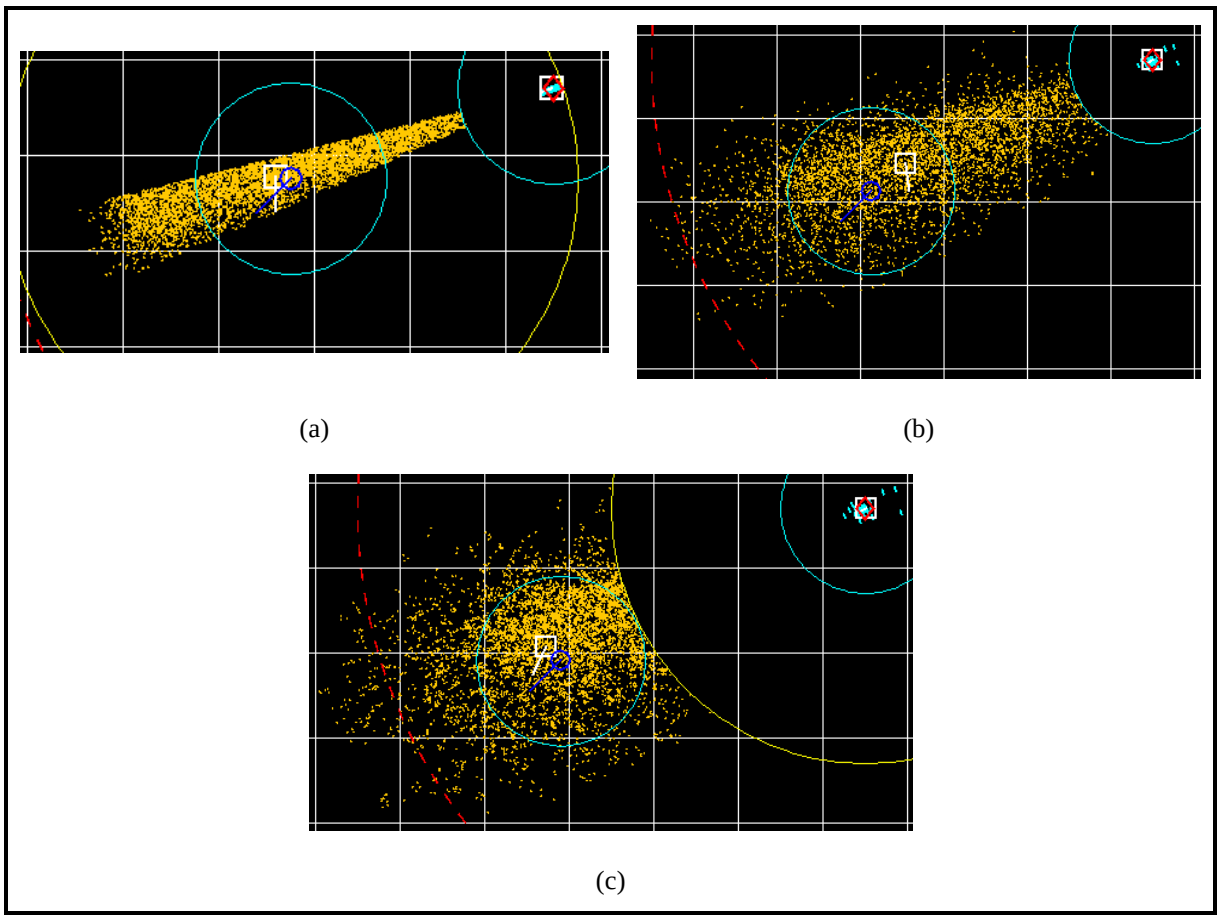


Figure 31. Particle Disqualification via Sanitization

b. Repopulation Algorithms

The disqualification of particles through additional detections and sanitization allows entities to refine their tracks of opposing entities through continued observation of the environment. These refinements result in increasing accuracy in the estimated positions, headings, and speeds of tracked platforms. However, if the only means of altering the track following a detection was the continued disqualification of particles, there could be a real possibility of disqualifying all the particles in the track. While losing a track is a possibility that needs to be allowed for, the need exists to repopulate the tracks with new valid particles when the opportunity presents itself.

Four methods for repopulating tracks with new particles were implemented in this simulation. The first method is used to replace particles when new detections or sanitizations have not resulted in a drastic alteration of the track state and is

similar to the *sampling importance re-sampling* method described in (Arulampalam, 2002). This technique is described in the *partial repopulation* section below. The second and third methods for carrying out repopulation are used when a new detection or sanitization has resulted in the disqualification of a large number of particles, providing a vastly different picture of the tracked platform's state, and are similar to the *regularization* re-sampling method described in (Arulampalam, 2002). These techniques are discussed in the *weighted position* and *estimated heading* bulk repopulation sections below. The last repopulation algorithm is a bulk repopulation algorithm which combines the weighted position and estimated heading bulk repopulation methods. This algorithm is discussed in the *combined* bulk repopulation section below. One of the difficulties associated with both of these repopulation methods is the possibility that some of the repopulated particles will be inconsistent with prior observations. To some extent this problem cannot be solved without providing the tracking platform with knowledge of the actual location of the tracked entity. The two bulk repopulation algorithms described below attempt to minimize the impact of inconsistent particles on the particle track through two different methods.

Although the repopulation methods resemble the re-sampling methods described in (Arulampalam, 2002) and (Bererton, 2004), they are slightly different due to the discrete event nature of this particular simulation. The re-sampling methods described in the above two works, in addition to preventing degeneracy problems, ensured that the particle filter was filled with an identical number of particles following each update. The repopulation algorithms described in this work are not run at every update of the particle filter. Instead, they are run when the number of valid particles remaining in the track falls below a certain threshold. These thresholds are similar to the sporadic communication thresholds described in (Klaas *et al* 2005). When the track information changes by a significant amount (as represented by the thresholds) one of the repopulation methods will be triggered to refill the track with new particles which represent the new track picture. Allowing particles to be disqualified from the track with the possibility of no repopulation results in a greater number of particles to be disqualified which results in better heading and speed estimations. There are two

thresholds associated with every particle track. The higher threshold is the one that triggers the partial repopulation algorithm. The lower threshold triggers one of the bulk repopulation algorithms.

Partial Repopulation Algorithm **–** The partial repopulation algorithm will be utilized when the disqualification of particles via detection or sanitization events has caused the ratio of valid particles remaining in the track to drop below a certain threshold. This repopulation technique is used to refill the particle track to its maximum capacity with valid particles. These new particles will take the form of duplicates of particles which are still considered valid. Valid particles are chosen for duplication in a manner similar to the re-sampling algorithm presented in (Arulampalam *et al.* 2001). A random draw will be compared against the weights of the particles still remaining in the track, with the result that particle with higher weights will have a higher probability of being duplicated during the repopulation process. Following the addition of new particles, the weights of the particles will be renormalized. The partial repopulation algorithm is presented in pseudo-code on the next page.

Partial Repopulation Algorithm

$c_1 = 0$ ***Initialize CDF***

FOR $i = 2 : N_{valid}$ ***Construct CDF***

$c_i = c_{i-1} + w_i$

END FOR

FOR $j = 1 : I_s$

$u_j = U[0,1]$ ***Get Random Draw***

$i = 1$ ***Start at Bottom of CDF***

WHILE $u_j > c_i$ ***Move Along CDF***

$i = i + 1$

END WHILE

$x_j^I = x_i; y_j^I = y_i; vx_j^I = vx_i; vy_j^I = vy_i; w_j^I = w_i$ ***Create Duplicate Particle***

END FOR

FOR $k = 1 : I_s$ ***Move Duplicate Particles to Valid Particles***

$N_{S+k} = I_k$

END FOR

Renormalize

Figure 32. Partial Repopulation Algorithm

Note that due to the movement model of naïve particles (never changing heading or speed) over many partial repopulations there will be a large number of particles coincident at several points in the environment. As this is a relatively inefficient use of a large number of particles, a different movement model for particles which allowed the calculation of accurate headings and speeds while achieving a spread

in the particle filter was implemented. This movement model will be described in the *contextual particles* section below.

This repopulation method is not suitable for maintaining a particle track in all cases. While it does an excellent job of capturing the location, heading, and speed of contacts which do not maneuver, it can fail on a maneuvering target. This concept is shown in the figure below. In (a), the blue platform's continued use of partial repopulation has resulted in an excellent track of the red platform. In (b), the red platform has executed a significant course change. If a detection or counter-detection of the red platform were to occur at this point, all the particles in blue's track would be disqualified, resulting in no means to perform a partial repopulation of the track.

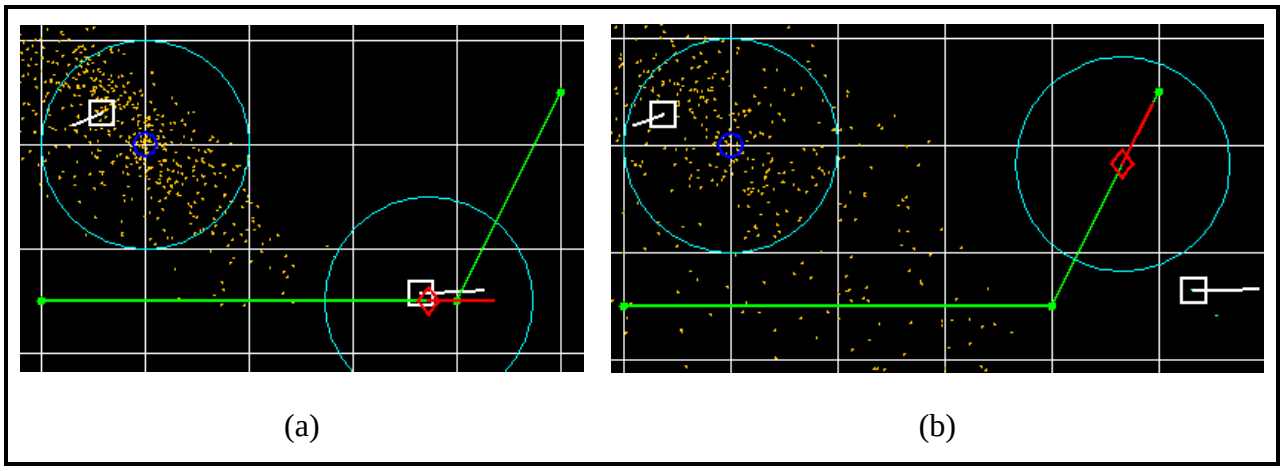


Figure 33. Unsuitability of Partial Repopulation as Sole Repopulation Method

In (Bererton 2004) if a track is lost due to observation of all the particles in a track without a target acquisition, the distribution of particles is reset to the initial distribution. Bererton's initial distribution consisted of the particles being uniformly distributed throughout the environment. While there is an initial distribution in this work, the initial detection distribution, it will not be used to regenerate a track which is not suitable for partial repopulation. Instead, two bulk repopulation algorithms are provided which generate a new set of particles based on a detection event and an old estimated position.

Weighted Position Bulk Repopulation Algorithm – In the event that a new detection event results in the disqualification of a very large number of

particles, this algorithm will attempt to create a new diverse particle track which reflects information from a past estimated position. This algorithm will be used to repopulate the track when the number of valid particles remaining in the track falls below a certain threshold due to the disqualification of particles from a new detection (not sanitization) event. New particles will be added to the track based on the detection distribution created by the sensor mediator to reflect the event. These particles will have positions, headings, and speeds varied uniformly across the distribution. The reflection of past tracking information will be taken into account by varying the weights of the new particles based on their distance from the last estimated position acquired via a detection event.

The weighting of new particles is accomplished by comparing their position to the location and speed of the last estimated position. Particles whose locations could be reached with little modification in speed on the part of the tracked target will be weighted higher than those particles whose locations require a significant change in speed by the target. The exact weighting of particles is determined by constructing a *window* around the estimated speed from the last detection and determining where in this window the speed required to reach the location of the new particle falls. A speed window with no numeric values is shown below.

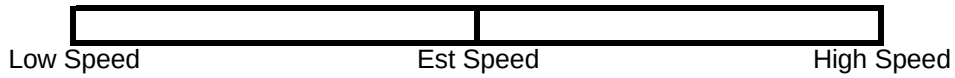


Figure 34. Blank Speed Window

While the minimum speed required to move from the old estimated position to the position of a new particle will never fall below zero, it is very possible that the maximum speed could be well above the maximum speed available to entities in the simulation. This requires that the low end of the speed window be allowed to extend below a speed of zero to ensure that the window is centered on the estimated speed. Speeds corresponding to the low and high ends are computed based on the old estimated speed. These values are computed as shown below.

DETERMINING END SPEEDS OF SPEED WINDOW

$s_l = 0$

$s_h = MAXSPD$

IF $s_{est} \leq \frac{MAXSPD}{2}$

$s_l = 2 * s_{est} - MAXSPD$

ELSE

$s_h = 2 * s_{est}$

Figure 35. Determining Ends of Speed Window for Position Bulk Repopulation Algorithm

Applying the above calculations to an estimated speed of ten with a maximum speed of thirty-two would yield the following speed window:

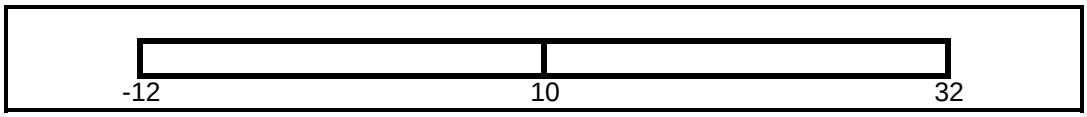


Figure 36. Speed Window for Estimated Speed of Ten

Applying the same calculations to an estimated speed of twenty-seven with a maximum speed of forty results in the following speed window:

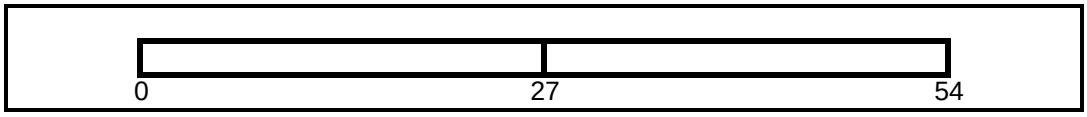


Figure 37. Speed Window for Estimated Speed of Twenty-Seven

Once the speed window is constructed, it is used to weight new particles being added to the track. The speed necessary to reach the new particle's location is computed based on the distance between this location and the old estimated position and the amount of simulation time elapsed from the creation of the estimated position. Once this speed is determined, the speed of the estimated position is subtracted

from it. Differences resulting in negative numbers indicate that the speed needed to reach the new particle is less than the estimated speed, and positive results indicate that the needed speed is greater. The distance of the calculated speed from its corresponding endpoint is found and used as the weight of the new particle. This will result in particles whose locations can be reached at exactly the estimated speed having maximum weight and those with required speeds far from the estimated speed having smaller weights. The two charts below show the weights which would be assigned to particles with various speed requirements for estimated position speeds of ten (a) and twenty-seven (b).

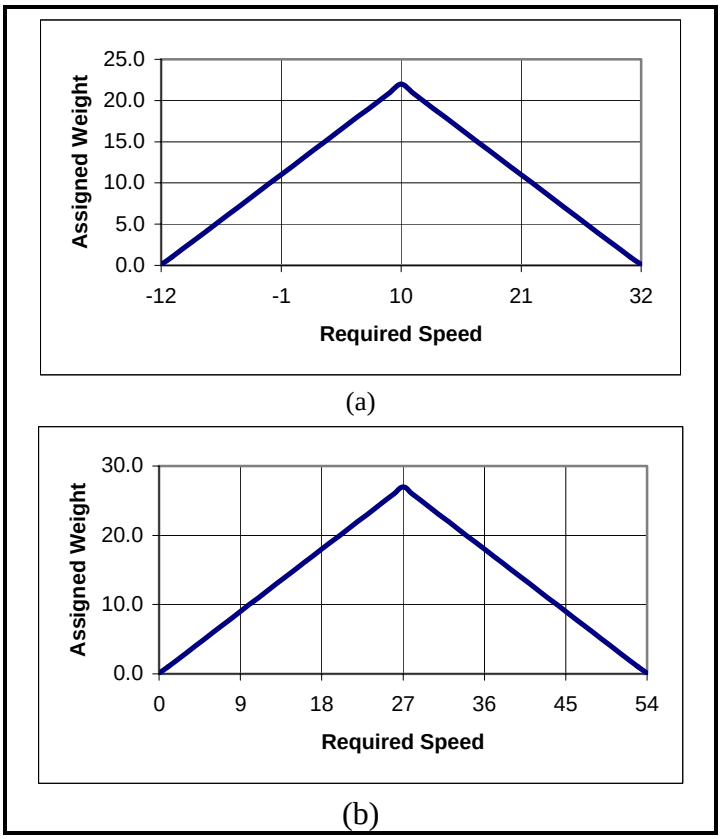


Figure 38. Assigned Weights for Various Required Speeds Based on Estimated Position Speeds of Ten (a) and Twenty-Seven (b)

Due to the possibility of required speeds increasing above the high end of the window, the minimum weight needs to be clamped at an arbitrarily small number to prevent excessive speed requirements resulting in negative particle weights. For example, in the speed window constructed above for a speed of ten, a required particle speed of forty without this clamp would result in a particle weight of negative

eight. In this simulation weight values were clamped at 0.1 to avoid having particles with zero weight in the track as a result of this repopulation algorithm. The possibility also exists for some particles to still remain in the track when this algorithm is used to repopulate the track. These particles are assigned the maximum weight obtainable through the use of this algorithm and maintained in the track. Following the assignment of these large weights to the particles the weights must be renormalized so that the sum of all particle weights in the track equal one. Note that the only attribute of new particles which are affected by this algorithm are the weights. The location, heading, and speed of all new particles will be drawn randomly from the detection distribution forwarded to the tracking entity by the sensor mediator. The complete Weighted Position Bulk Repopulation Algorithm is shown on the next page.

Weighted Position Bulk Repopulation Algorithm

Calculate Speed Window

FOR $i = 1 : N_s$ ***Set Weight of Remaining Particles to Maximum***

$w_i = s_h - s_{est}$

END FOR

FOR $j = 1 : I_s$ ***Redefine Invalid Particles***

$x_j, y_j, vx_j, vy_j = U[Detection]$ ***Values from Detection Distribution***

$d_j = \sqrt{\left(x_j - x_{est}\right)^2 + \left(y_j - y_{est}\right)^2}$ ***Compute Distance from Est Posit***

$s_{req} = d_j * \frac{60}{t_{cur} - t_{est}}$ ***Compute Speed Required***

$s_{diff} = s_{req} - s_{est}$ ***Find Speed Difference***

IF $s_{diff} < 0$ ***Assign Appropriate Weights***

$w_j = \max(s_{req} - s_l, w_{\min})$

ELSE

$w_j = \max(s_h - s_{req}, w_{\min})$

$N_{s+j} = I_j$ ***Move New Particle to Track***

END FOR

Renormalize

Figure 39. Weighted Position Bulk Repopulation Algorithm

The effect of using this repopulation method on an existing particle track is shown in the figure below. In (a), the red platform has a track on the blue platform which is the result of an initial detection. In (b), the blue platform has turned on its radar, resulting in a counter-detection by the red platform. Due to the large number of particles which were disqualified, the weighted position bulk repopulation algorithm was used to repopulate the track. The picture in (c) is the result of the same situation with a different repopulation method employed. Notice how the estimated position in (b) is skewed towards the actual position of the blue platform despite the large number of particles in the "neck" of the detection distribution.

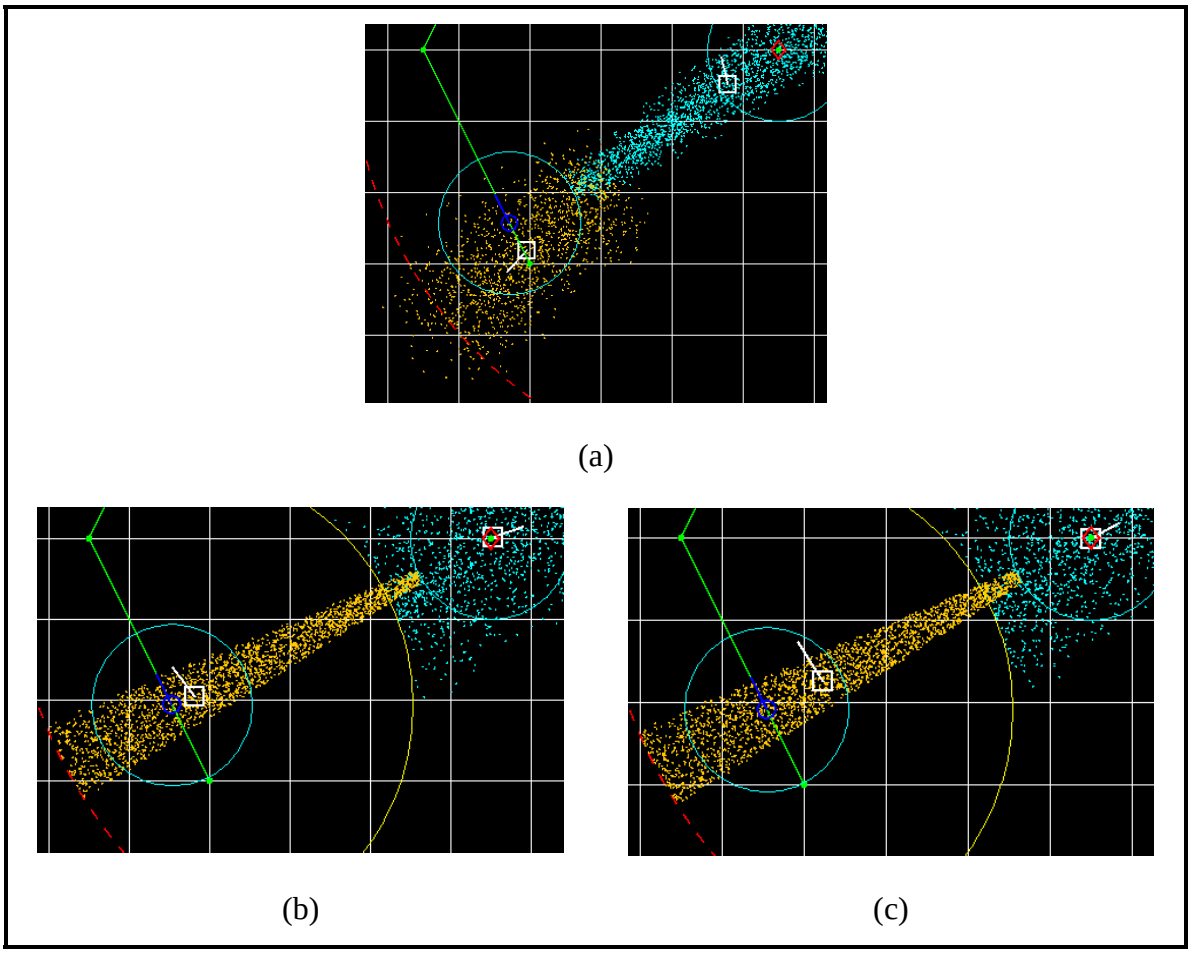


Figure 40. Result of Applying the Weighted Position Bulk Repopulation Method

A Parzen-windows estimate of the weight distribution density of the particle track following the application of the weighted position algorithm in the

above situation is shown below. The surface in the figure corresponds to the weights of particles present at the *x* and *y* positions of the track shown in (b) of the above figure. Note that the majority of the weight is towards the lower left of the track as opposed to the upper-right despite that fact that the actual concentration of particles is higher in the upper-right portion of the track.

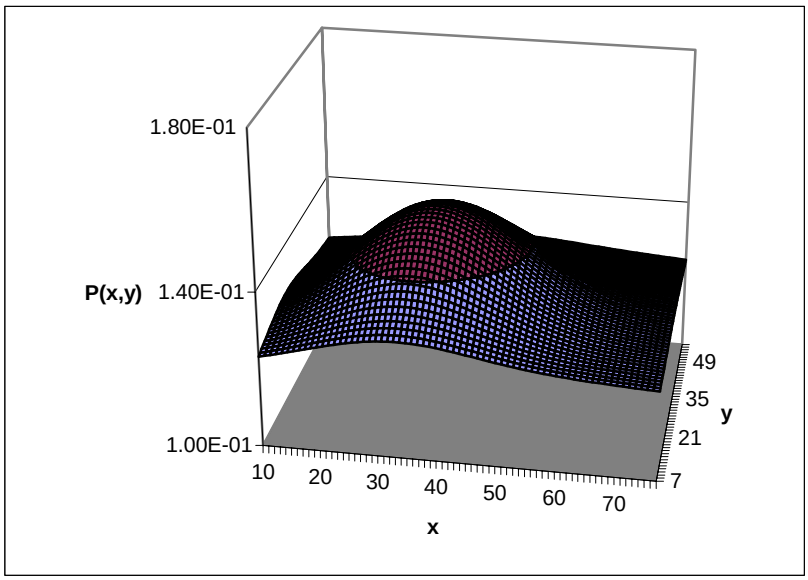


Figure 41. Parzen-Windows Approximation of the Weight Distribution Density for a Track Following Weighted Position Bulk Repopulation (Window Width 4×10^{-4})

Estimated Heading and Speed Bulk Repopulation Algorithm – This algorithm also attempts to repopulate a particle track by taking into account a past estimated position. Like the weighted position repopulation algorithm, it is triggered by a detection event which results in the disqualification of a large number of particles from a track. A threshold ratio of the remaining number of particles is used to decide if a sufficiently large number of particles were disqualified. Rather than changing the weights of particles based on their distance from the last estimated position, this algorithm will alter the heading and speed of particles based on their orientation to the last estimated position.

The alteration of headings and speeds is accomplished by first filling the new detection distribution with uniformly distributed particles. The location of

each of these new particles is then used to calculate the course and speed needed from the old estimated position to reach the new particle's location. In order to allow for changes to course and speed by the target at any point, half of the particles are left with random courses and speeds. All of the new particle will be weighted equally with the average weight of any particles remaining in the track. This process is shown on the next page.

Estimated Heading and Speed Bulk Repopulation Algorithm

Find w_{avg} ***Find Average Weight of Remaining Particles***

FOR $j = 1 : I_s$ ***Redefine Invalid Particles***

$w_j = w_{avg}$

$x_j, y_j = U[Detection]$ ***Positions from Detection Distribution***

$u_j = U[0,1]$

IF $u_j < 0.5$

$d_j = \sqrt{\left(x_j - x_{est}\right)^2 + \left(y_j - y_{est}\right)^2}$ ***Find Distance Traveled

$s_j = \min\left(d_j * \frac{60}{t_{cur} - t_{est}}, MAXSPD\right)$ ***Find Speed Needed***

$hdg_j = BearingTo\left(x_{est}, y_{est}, x_j, y_j\right)$

$vx_j = \frac{s_j * \cos\left(hdg_j\right)}{60}$ ***Set *x* Velocity Component, Hdg in Radians***

$vy_j = \frac{-s_i * \sin\left(hdg_j\right)}{60}$ ***Set *y* Velocity Component***

ELSE

$vx_j, vy_j = U[Detection]$ ***Values from Detection Distribution***

END FOR

Move Invalid Particles to Valid Particles

Renormalize

Figure 42. Estimated Heading and Speed Bulk Repopulation Algorithm

The result of applying this algorithm is a completely repopulated particle track with half of the particles having courses and speeds which "fan" out from the old estimated position. This is shown in the figure below. In (a), the blue platform has an accurate track on the red platform. This picture is taken right after the blue platform has turned off its radar, so the estimated position shown will be the one used in the repopulation algorithm. In (b), the red platform has made a significant course change. In (c), the red platform has turned on its radar resulting in a counter-detection by the blue platform. Due to the small number of particles left in the track, the estimated heading and speed bulk repopulation algorithm has been triggered. Notice the estimated heading and speed of the contact with relation to the old estimated position in (a). In (d), the red platform has turned off its radar and time has progressed. The particles which had a heading and speed assigned based on the estimated position have continued the movement toward the north-east while the particles with random headings and speeds have continued to spread out.

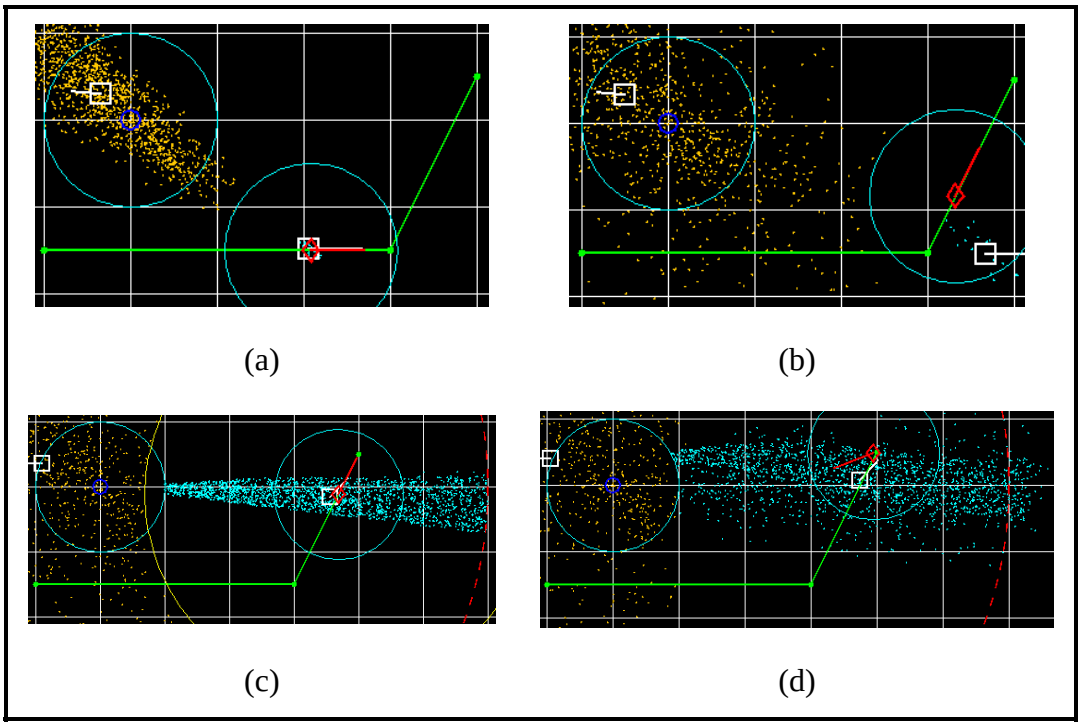


Figure 43. Result of Applying the Estimated Heading and Speed Bulk Repopulation Algorithm

Parzen-windows estimates of the heading and speed distribution density of the track following the application of the heading and speed repopulation method are shown below. Due to the application of the repopulation algorithm, they are very different from the approximately uniform distribution of headings and speeds which result from an initial detection.

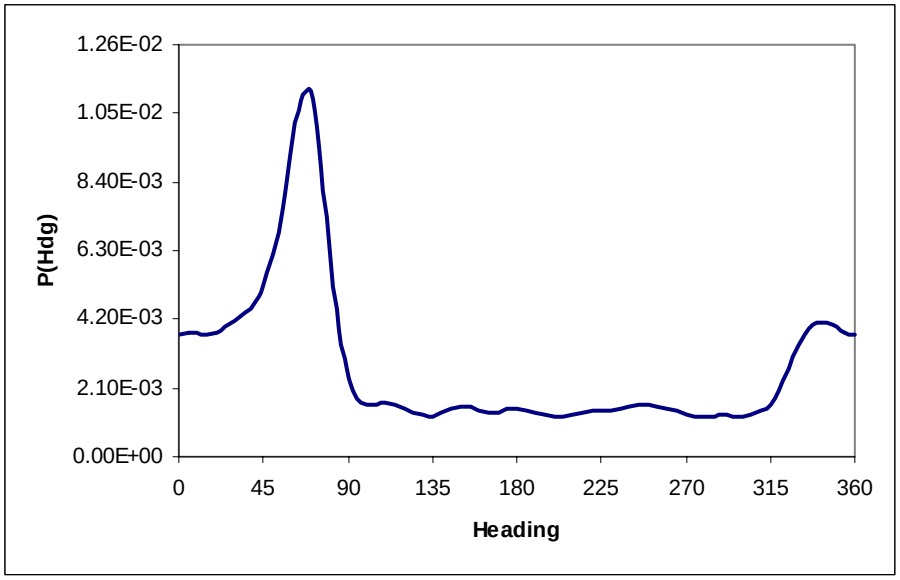


Figure 44. Parzen-Windows Approximation of Heading Distribution Density Following Estimated Heading and Speed Bulk Repopulation Method (Window Width 7.2)

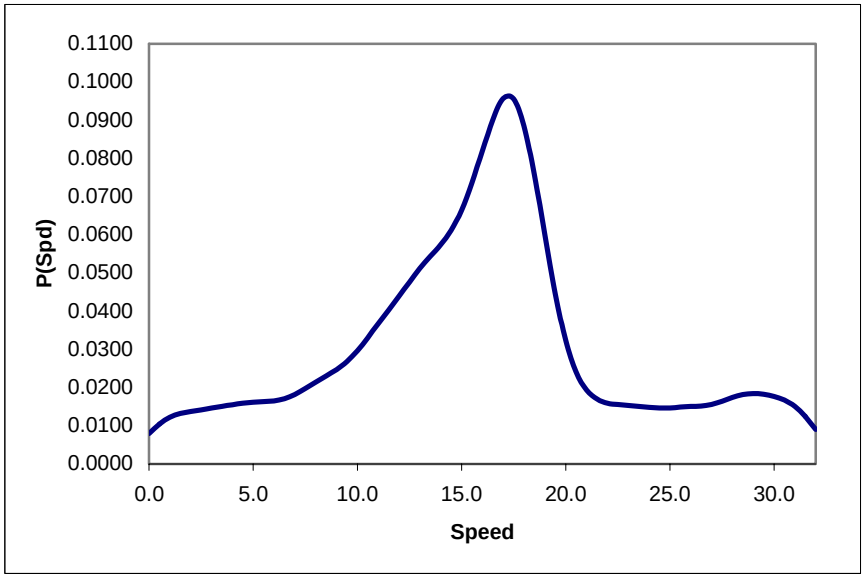


Figure 45. Parzen-Windows Approximation of Speed Distribution Density Following Estimated Heading and Speed Bulk Repopulation Method (Window Width 1.0)

Combined Bulk Repopulation Method **-** It is possible to combine the two bulk repopulation methods described above into one bulk repopulation method. In the *combined bulk repopulation method*, particles will have their heading and speed computed based on the estimate heading and speed bulk repopulation method and their weights determined using the weighted position bulk repopulation method. The different effects of these three bulk repopulation methods are shown below. The red platform's track of the blue platform has just undergone bulk repopulation due to an updated counter-detection of blue's radar.

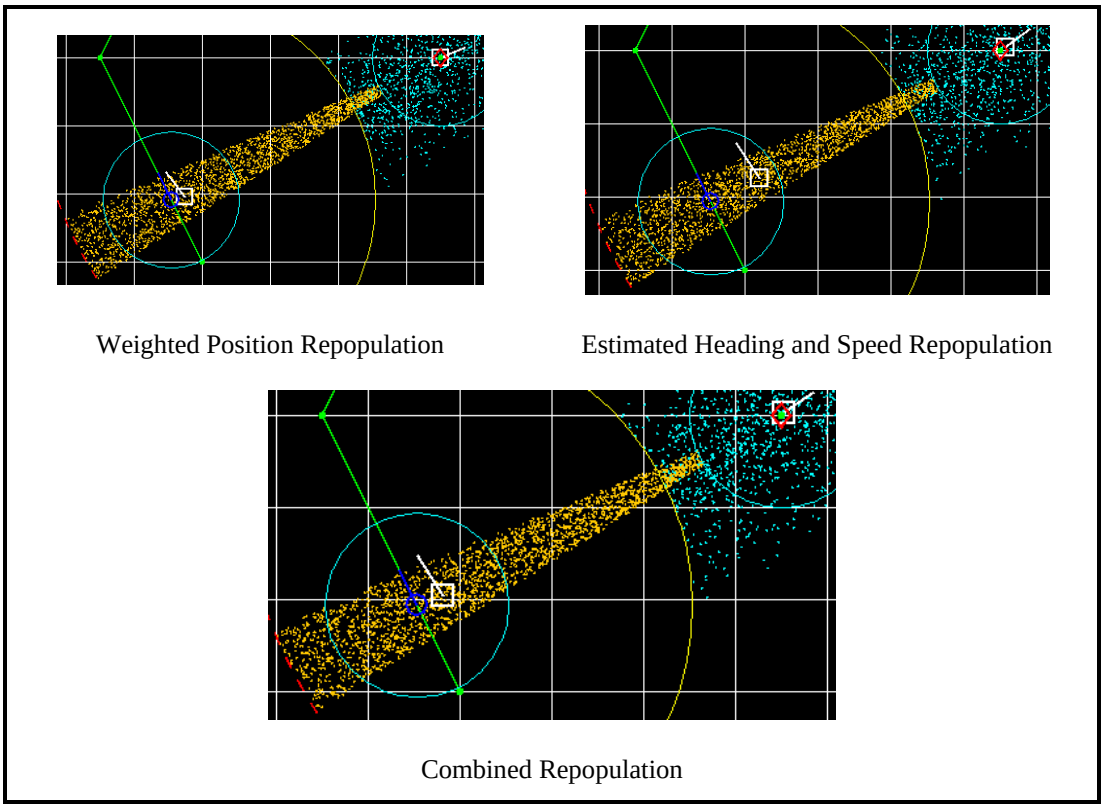


Figure 46. Effects of Different Bulk Repopulation Methods

3. Complete Naïve Particle Track Update Algorithm

The disqualification and repopulation methods described above provide the means to maintain a particle track after an initial detection and continually refine that track's accuracy through its lifespan. Due to the discrete event nature of the simulation, these methods are not employed on a continual basis but only when a sensor sweep event occurs through the course of the simulation. When these events occur, particles which no longer reflect possible locations of the tracked entity can be disqualified and, if

appropriate, new particles which do reflect a possible state of a tracked entity can be added to the track. Putting all of the above techniques together yields the parameters of a naïve particle track as used in this simulation. These parameters and their corresponding notation are shown below.

Naïve Particle Track Parameters			
Maximum Number of Particles	Partial Repopulation Threshold	Bulk Repopulation Threshold	Bulk Repopulation Method
$N_{\max}$	r_P	r_B	R_B

Table 4. Naïve Particle Track Parameters

With a track existing due to the sampling of an initial detection distribution, the algorithm employed to update the track at each sensor sweep event is as shown on the next page.

Naïve Particle Track Update Algorithm

FOR $i = 1 : N_S$ ***Update Particle Positions Based on Time Since Last Event***

$x_i = x_i + vx_i * t_{elapsed}$, $y_i = y_i + vy_i * t_{elapsed}$

END FOR

IF Sweep Event Results in Detection Event

$I_i = N_i \forall N_i \notin [Detection]$ ***Disqualify Particles***

Renormalize Remaining Particles

IF $\frac{N_s}{N_{\max}} < r_P$ AND $\frac{N_s}{N_{\max}} > r_B$

Repopulate according to Partial Repopulation Algorithm

IF $\frac{N_s}{N_{\max}} < r_B$

Repopulate according to R_B (Bulk Repopulation Algorithm)

ELSE ***Sweep Event Resulted in Sanitization***

$I_i = N_i \forall N_i \in [Detection]$ ***Disqualify Particles***

Renormalize Remaining Particles

IF $\frac{N_s}{N_{\max}} < r_P$ AND $\frac{N_s}{N_{\max}} > r_B$

Repopulate according to Partial Repopulation Algorithm

IF $\frac{N_s}{N_{\max}} < r_B$

Discard Track

Calculate New Estimated Position and Store Unless Track Lost

Figure 47. Naïve Particle Track Update Algorithm

The algorithm above shows that the only way for a platform in this simulation to lose a track is to disqualify enough particles through sanitization that the total number remaining fall below the bulk repopulation threshold. This is due to the fact that no detection distribution will exist in which to distribute new particles. When a track is lost, no new estimated position is calculated, saving the older estimated position for use with the corresponding bulk repopulation method if a new detection event occurs, and all particles are discarded.

4. Contextual Particles

The naïve particle filter as described above does an admirable job of generating accurate estimated positions, headings, and speeds for tracked entities in the simulation for both active and passive tracks. However, the naïve particle approach begins to lose its effectiveness when the tracked entity begins to maneuver. This issue is particularly troublesome when tracking via passive means. As several of the examples have shown, when a tracked entity moves out of detection range and maneuvers, the usefulness of the track in re-locating the target is questionable if no more detection events are forthcoming.

Contextual particles were developed specifically to counter this weakness in the naïve particle approach. The primary aim was to allow the extraction of estimated heading and speed data from a track while at the same time providing for a spread in track uncertainty in between detection events. A secondary aim was to provide a means to inject simple movement behaviors into individual particles so that a particular particle track could represent an uncertainty picture which would occur if the entity being tracked was behaving in a specific manner. Accomplishment of the first goal is described in the *general particles* section below. The extensions to general contextual particles which addressed the second goal are described in the *transitioning particles* section below. One of the strengths of these contextual particles is that they can be substituted for naïve particles in the particle filter update algorithm with no changes as all differences are encapsulated in the individual particles.

a. General Contextual Particles

General particles are based on the concept that at any point in time, a target being tracked may or may not be maneuvering. Naïve particles, in a manner similar to target motion analysis techniques, assume that the heading and course of the

target will remain static over the time of the tracking problem. When a target maneuvers, the tracking problem is "reset" and the process to determine the target's heading and speed begins anew. General contextual particles obviate the need to reset the tracking problem through the use of a particle-level movement model which accounts for the possibility that the tracked target may change or maintain its course and speed at any time.

The general contextual particles implement this movement model through the use of a movement model update algorithm. This algorithm is scheduled for individual particles at some predetermined interval and when pulled off the event queue modifies the particle's state as appropriate. The algorithm relies on a provided probability of maintaining course and speed (p_{chg}). In this simulation, that probability was chosen as 0.5, meaning that a particle was equally likely to change or maintain course and speed at any update time. The simple algorithm for updating a contextual particle's movement model is provided below.

General Contextual Particle Update Algorithm

FOR N_i ***Each Particle Updates Individually***

$u_i = U[0,1]$

IF $u_i < p_{chg}$ ***Particle will Change Course and Speed***

$hdg_i = U[0,360]$ ***Note that these will be converted to vx_i, vy_i ***

$spd_i = U[0, MAXSPD]$

ELSE

Maintain course and speed (no action)

Figure 48. General Contextual Particle Update Algorithm

Other than the algorithm above, an interval of time between individual particle updates is required to schedule particle update times with the event queue. In this simulation, the interval [3.0,15.0] in simulation minutes was used. Using the general

contextual particles prevents the duplication of identical particles which occurs when using the naïve particle filter. This allows greater diversity and provides for an increase in uncertainty as a function of time since the last detection.

The results from replacing naïve particles with general contextual particles in a particle filter track are shown in the figure on the next page. In (a) and (b), the blue platform is using naïve particles to fill its particle track. In (c) and (d), general contextual particles are being used. In (a) and (c), the estimated position of the red platform is the result of an active track by the blue platform. The blue platform has just turned off its radar. In (b) and (d), the red platform has continued to move and made a significant course change. The naïve particle track presents an estimated position based on the last known course and speed while the contextual track has spread to reflect the lack of any detection information in the intervening time periods. If a new detection were to occur, the naïve track would have to repopulate the track using one of the bulk repopulation methods while the contextual track would rely on partial repopulation which would result in a more accurate heading and speed estimate of the red platform.

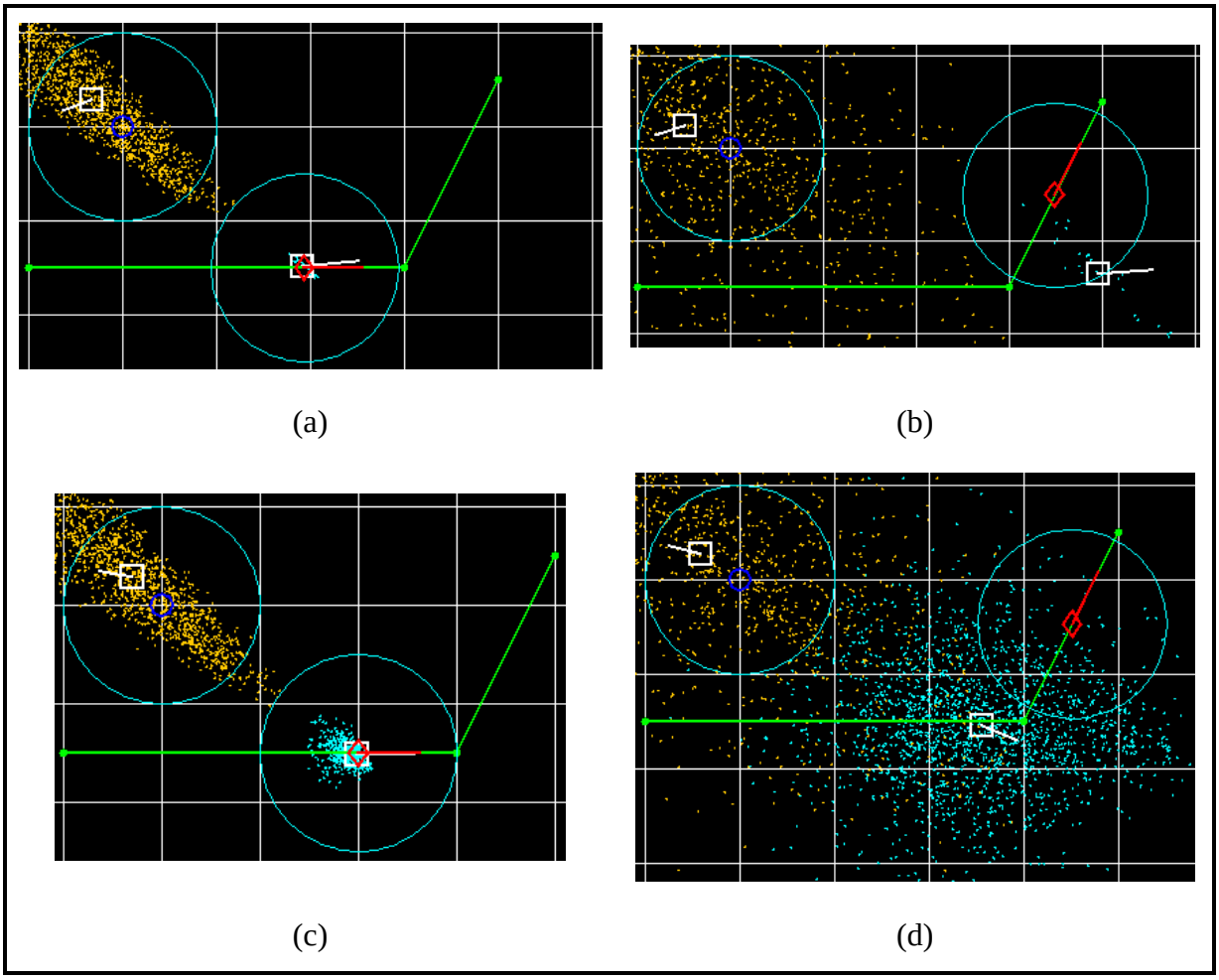


Figure 49. Difference between Naïve and General Contextual Particle Tracking

b. Transitioning Contextual Particles

Once the general contextual particle scheme was implemented it became clear that this method of updating individual particle movement models could be used to include more complex behavior in the tracks. If a decent and relatively simple behavioral model was available for entities within the simulation, it could be included in the individual particle movement models. This concept was tested by allowing general contextual particles to transition to a more advanced movement model with a small probability (p_{trans}) at each individual particle update cycle. Once a particle was transitioned, all further particle movement model updates were governed by a different behavior. This concept is reflected in the transitional contextual particle update algorithm shown below.

Transitional Contextual Particle Update Algorithm

FOR N_i ***Each Particle Updates Individually***

IF N_i NOT *transitioned*

$u_i = U[0,1]$

IF $u_i < p_{chg}$ ***Particle will Change Course and Speed***

$hdg_i = U[0,360]$ ***Note that these will be converted to vx_i, vy_i ***

$spd_i = U[0, MAXSPD]$

IF $u_i > p_{chg}$ AND $u_i < p_{chg} + p_{trans}$

Maintain course and speed (no action)

IF $u_i > p_{trans}$ ***Transition Particle

$vx_i, vy_i \leftarrow [Behavior]$

transitioned = TRUE

ELSE ***Particle has Already Transitioned***

$vx_i, vy_i \leftarrow [Behavior]$

Figure 50. Transitional Contextual Particle Update Algorithm

Two simple transitioned behaviors were implemented in this work to demonstrate the use of transitional contextual particles. These two behaviors are "hiding" and "seeking" transitioned behavior. Both of these movement models are based on a (possibly false) assumption that the platform being tracked either wants to avoid or force contact. If the assumption is that the platform being tracked wants to avoid contact, then upon detecting another entity it will change course to a reciprocal bearing of the detection. For example, a platform attempting to avoid detection gains a track on an opposing platform due west (270 degrees) of its current location. The avoiding platform would then turn to a new course directly away from the bearing of the other platform, in this case due east (90 degrees). With a similar maximum speed to other entities, this

reciprocal course will offer the best chance of forcing another platform to engage in a tail chase to acquire the target. If the assumption is that the platform being tracked wants to force contact, then upon detecting another entity it will maneuver to an intercept course. As these particles are a reflection of an assumed behavior on the part of the other entity being tracked they have access to the actual state of the tracking entity. Thus, the behavior of the particles will display a "worst case" scenario. In other words, they will display the possible behavior of the tracked entity as if it had perfect knowledge of the tracking entity. To reflect these behaviors, transitional contextual particles in a track of such a platform will exhibit the same behavior. A transitional probability for particles was chosen to be $p_{trans} = 0.1$ in order to ensure a large number of transitions for visualization purposes.

Transitional particles which exhibit a hiding behavior are shown below. In (a), the blue platform is operating its radar and has acquired a track of the red platform. In (b), the blue platform has secured its radar and continued moving along its patrol plan. Several of the contextual particles in blue's track have transitioned into the hiding behavior. These particles are colored green, and their behavior (moving on a reciprocal bearing from blue) has begun to alter the estimated position of the red platform. In (c), this process has continued and a large number of particles have transitioned into the hiding behavior. Due to having perfect knowledge of the tracker's location, the estimated position has changed to reflect an updated course estimate based on the blue platform's new location.

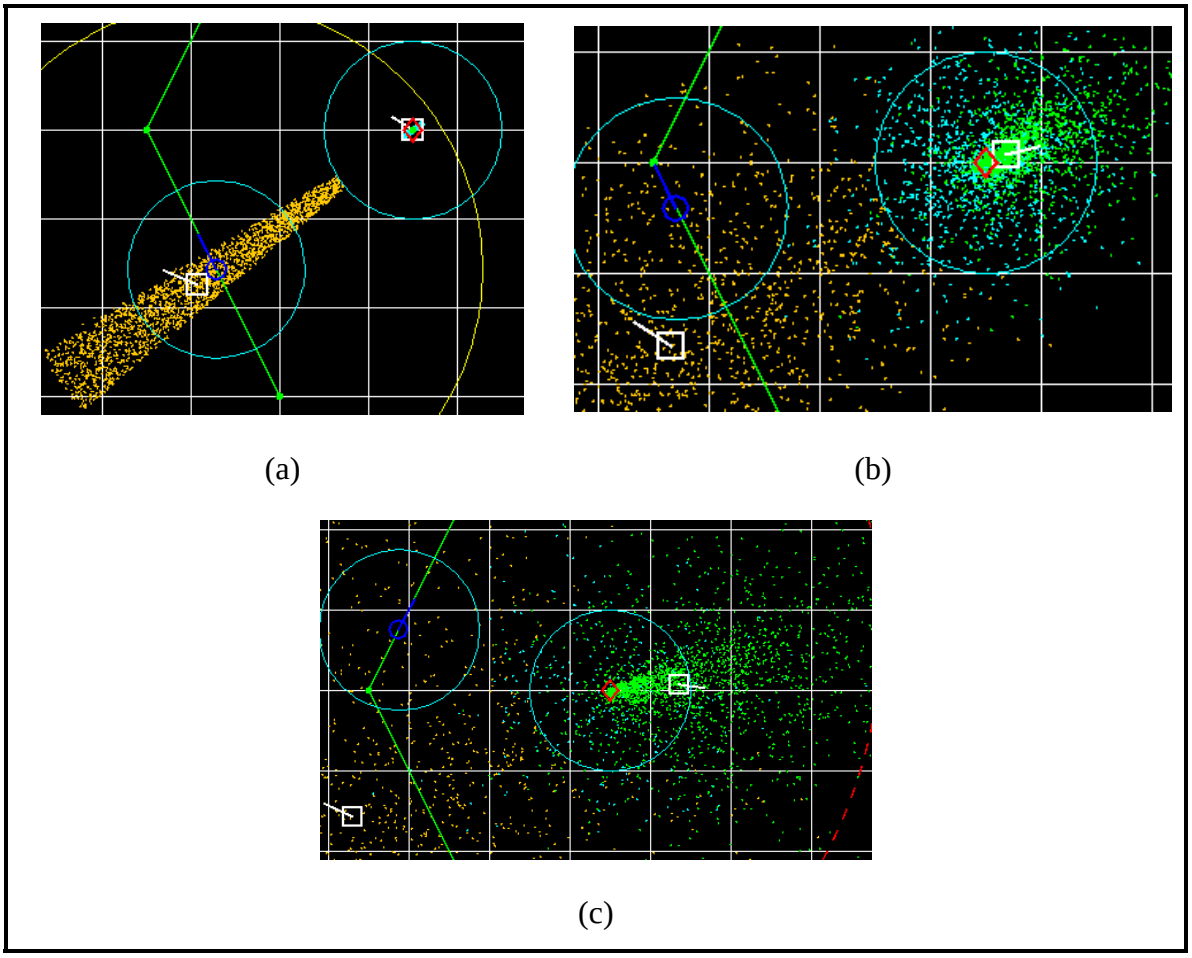


Figure 51. Transitional Contextual Particles Displaying a “Hiding” Behavior

Transitional particles which display a seeking behavior are shown below. In (a), the blue platform is operating its radar and has acquired a track of the red platform. In (b), the blue platform has secured its radar and continued moving along its patrol plan. Several of the contextual particles in blue’s track have transitioned into the seeking behavior. These particles are colored green and their behavior (moving on a course to intercept blue) has begun to alter the estimated position of the red platform. In (c), this process has continued, and a large number of particles have transitioned into the seeking behavior. Due to the continued sanitization of particles which enter into visual range of blue, the estimated position for the red platform has only moved a small distance. However, the estimated course has changed to reflect the blue platform’s new heading and speed.

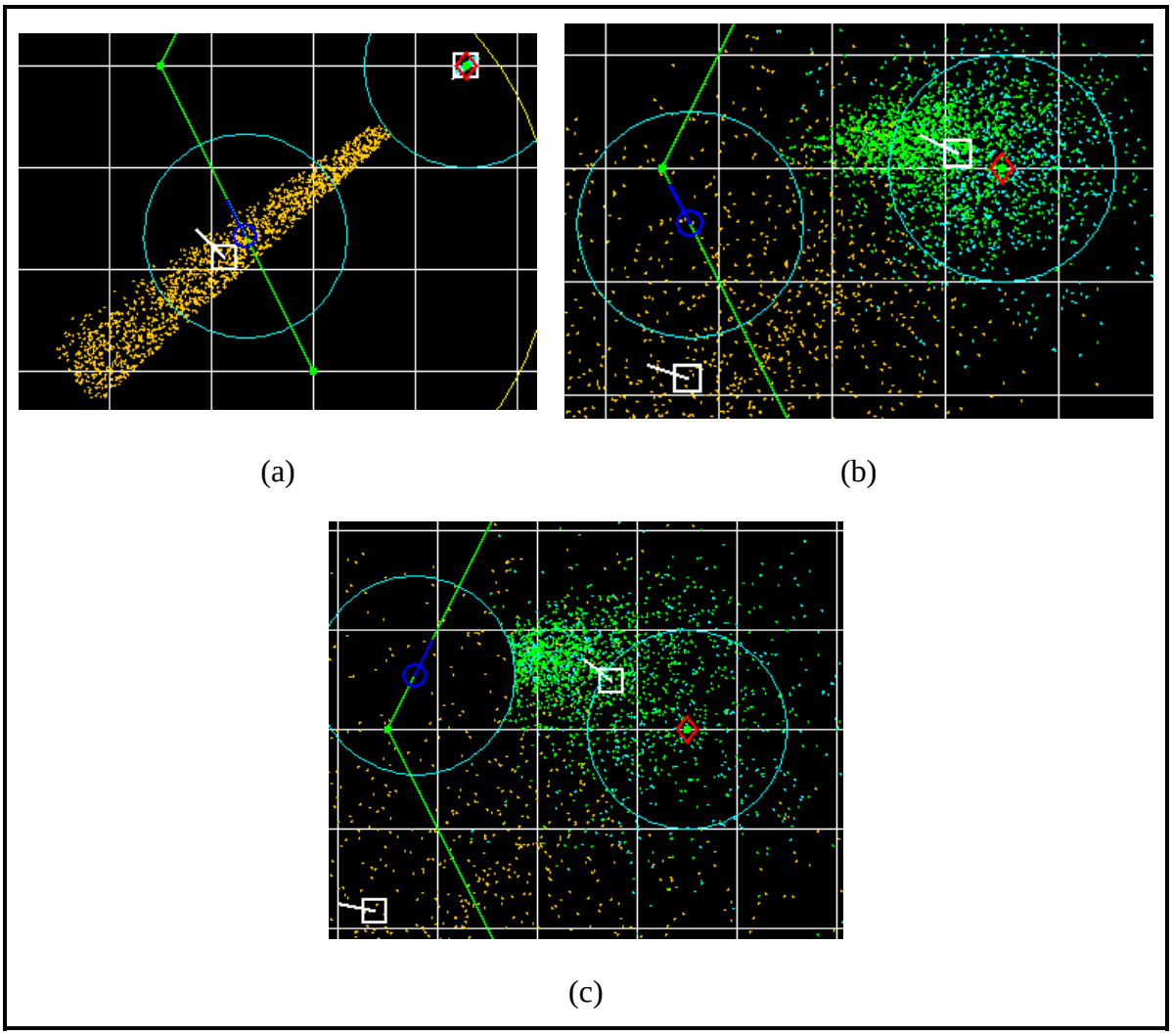


Figure 52. Transitional Contextual Particles Displaying a “Seeking” Behavior

While the two transitional behaviors shown above are relatively simple in nature, they nonetheless illustrate the possible usefulness of contextual particles in particle based tracking. With more complex or numerous transitional behaviors, particle filter tracks could present an increasingly realistic (although possibly wrong) picture of tracked entity behavior. With more than one possible transitional behavior for a given particle, the possibility of classifying a track based on the types of transition particles it held exists. For example, a particle track consisting of possible transitions to both hiding and seeking behavior could be used to determine if the tracked entity was displaying a hiding or seeking behavior. A metric designed to quantify the number or rate of disqualification of particles which have transitioned to a specific behavior could be used

to change the transition probabilities of contextual particles in the track, thus fine tuning the behavioral aspects of the particles being used for tracking.

C. USING THE PARTICLE TRACK

The usefulness of a tracking technique is to a certain extent characterized by the accuracy of the state estimations made about the tracked entity. Estimated position, heading, and speed have been mentioned as the primary measures of interest in this work. One of the benefits of the particle filter tracking method is its ability to represent the state of tracked entities both as a single point of interest and as an area of uncertainty. The usefulness of single point estimations, referred to in the work above as the estimated position, is discussed in a section below. The area of uncertainty representation of the particle based track is discussed along with a simple path planning method as implemented in the simulation environment.

1. Estimated Positions Again

The estimated position as illustrated in many of the figures in preceding sections can be calculated in a trivial manner using the state of the particle track at any time. These calculations are shown below.

Estimated Position Calculation

FOR $i = 1 : N_s$ ***Compute Estimated Position of Entity being Tracked***

$x_{est} = x_{est} + w_i x_i$ ***Compute Estimated *x*****

$y_{est} = y_{est} + w_i y_i$ ***Compute Estimated *y*****

$vx_{est} = vx_{est} + w_i vx_i$ ***Compute Estimated *x* Velocity***

$vy_{est} = vy_{est} + w_i vy_i$ ***Compute Estimated *y* Velocity***

END FOR

Figure 53. Computing an Estimated Position from a Particle Track

This method of calculating an estimated position yields a reasonably accurate result when the particle filter is composed of naïve particles and detections occur relatively frequently. This is due to the movement model of naïve particles; naïve

particles will maintain their course and speed until disqualified. Additionally, the high rate of detections keeps the spread of the particle filter to a minimum in between updates. This decreases the possibility of having to perform a bulk repopulation. With low rates of detections, this estimate can still be useful if the estimated heading and speed bulk repopulation algorithm is used to repopulate the track. Due to distribution of courses which result in the particle track using this algorithm, the estimated position will move with the tracked platform as opposed to the weighted position method which results in many random headings and speeds.

When using general contextual particles, the accuracy of this method of calculating estimated positions varies with the length of time from the last detection event to the current time's estimated position. Immediately following a detection event, the estimated position will have a higher degree of accuracy due to the ability to conduct partial repopulation of the particle track in a greater variety of circumstances. Since particles used in a partial repopulation accurately reflect the state of the tracked entity, an estimated position calculated immediately following repopulation will be the most accurate. However, as contextual particles change their motion to provide a spread to the area of uncertainty in the absence of further detections, the estimated position will stay pinned to the center of the area of uncertainty (the location of the last estimated position). This behavior is shown below. In (a), the platform has just secured its radar. The estimate position is reasonably accurate for a passive detection. In (b), the blue platform has continued moving. Roughly half of the general contextual particles have assumed random headings and speeds which caused "spread" in the particle cloud. While the estimated heading and speed is still accurate, the position estimate has been biased towards the center of the forming cloud and has not moved with the blue platform.

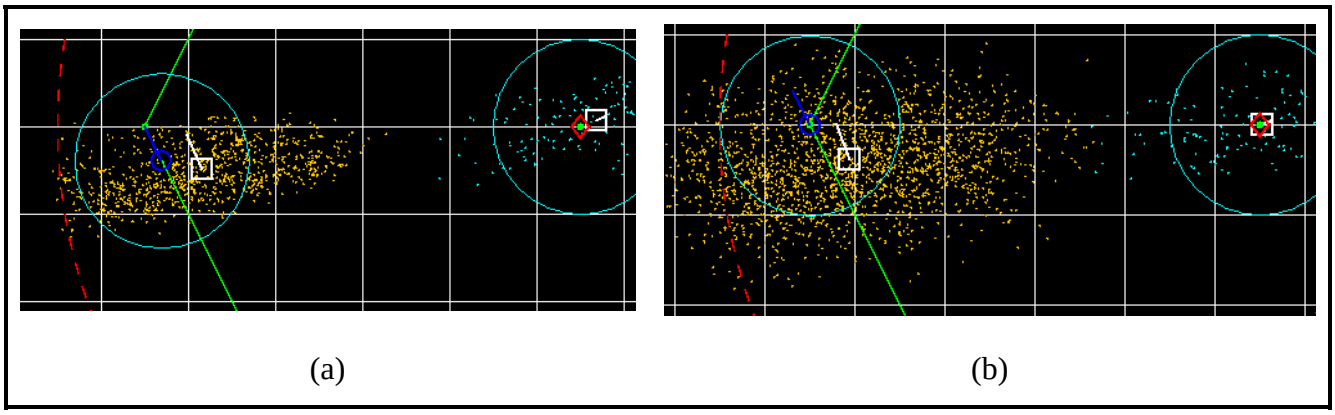


Figure 54. Limited Usefulness of Estimated Position when Using General Contextual Particles

This problem with the estimated position as calculated above can be addressed by storing estimated positions resulting from detection events and moving them forward in time independent of the particle cloud. As these estimated positions are already stored for use with bulk repopulation methods should they become necessary, this is an "easy" fix. The increased accuracy of this different estimated position is shown below. The red platform is using a particle track consisting of general contextual particles to track the blue platform passively. The estimated position calculated from the current state of the particle filter is shown in white as usual. An estimated position based on the detection estimated position moved forward in time is shown in magenta.

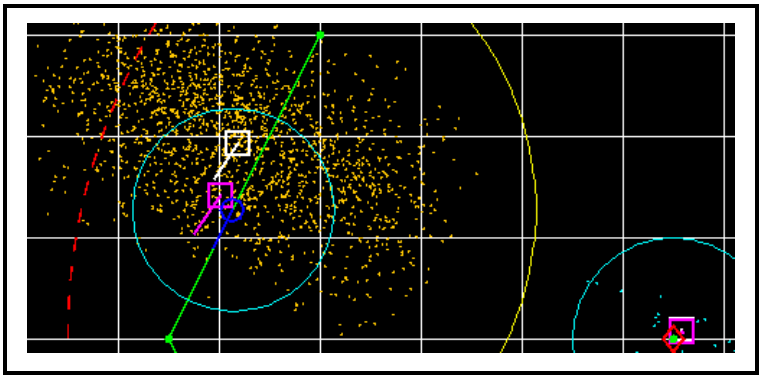


Figure 55. Difference in Estimated Position Types

While both types of estimated positions can be useful, there are many instances where due to infrequent detections and corresponding updates to the particle track the estimated position can fail to provide an accurate result. One of these instances is shown below. In (a), the red platform is operating its radar resulting in a track on the blue

platform. The blue platform has a passive track on the red platform. In (b), the red platform has turned off its radar and reversed course. In (c), the red platform continues to the south while the blue platform has turned to approach the estimated position of the red platform. In (d), the blue platform has brought the estimated position of the red platform under visual observation but has not acquired the red platform (which is out of range to the south).

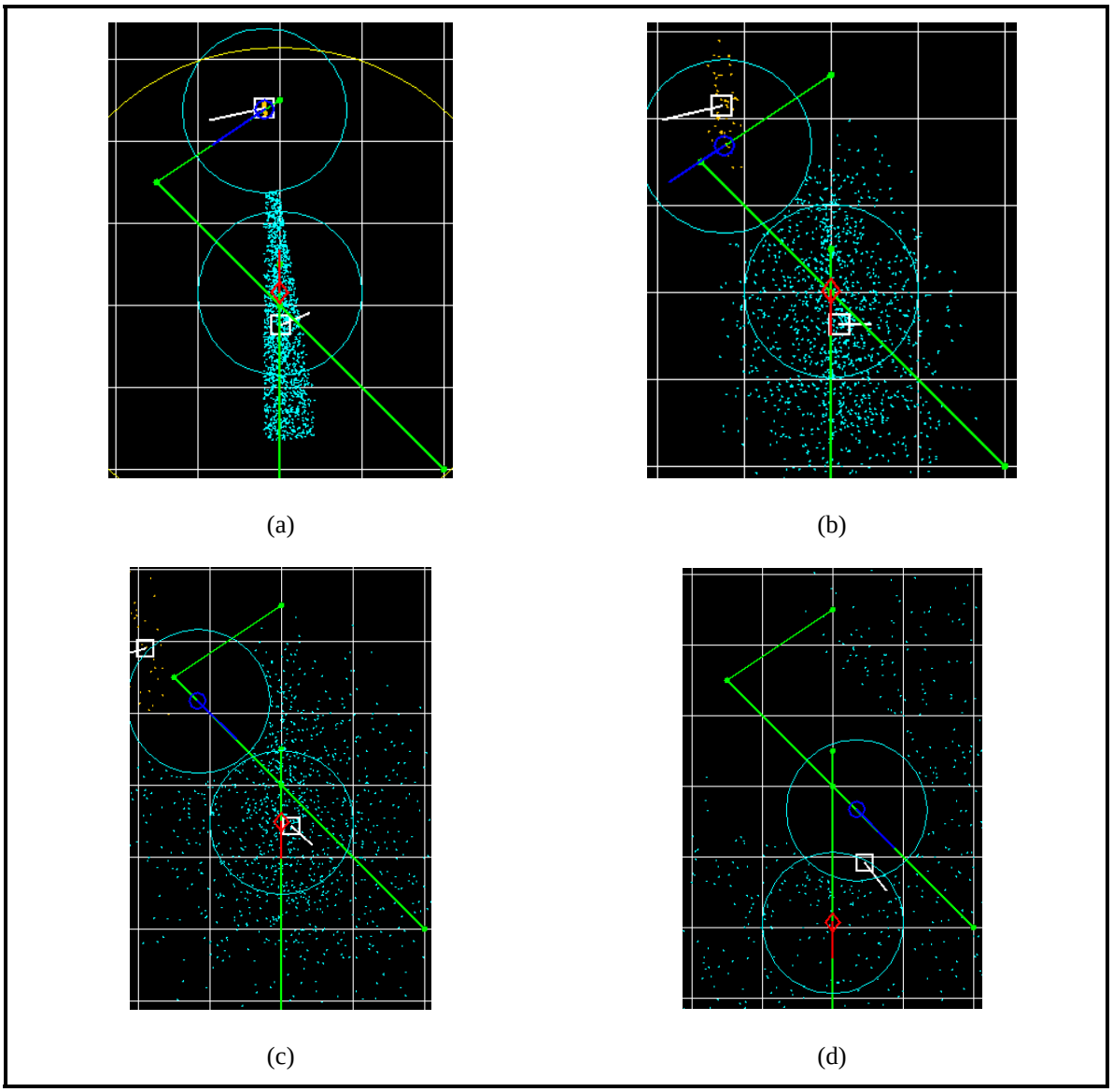


Figure 56. Estimated Position Failure

While a logical course of action for the blue platform in the situation above would be to turn on its radar upon bringing the estimated position of the red platform to an

appreciably close distance (which would result in detection in this case), there are many instances where this would be undesirable. In these cases, the treatment of the particle track as a large number of samples as opposed to an estimated position provides the means to continue the search. Although estimated positions can fail, they provide a starting point for decision making, and in cases where the track in question has benefited from frequent detection events and can provide highly accurate information. An example showing the use of particles as a number of samples to plan paths through the environment is provided in the next section.

2. Large Number of Samples

Using the particles in a given track individually can have benefits over attempting to utilize the estimated positions shown above. For such tasks as path planning or searching, the individual particles in a track can be used in cost functions or as metrics to trigger a certain behavior. For example, by "binning" particles in a way similar to occupancy maps in (Isla, 2006), an agent could plan a path through an environment to either avoid or force detection. By counting the number of particles in a given track which are within a specified sensor footprint, an agent could decide when to use long range active sensors to accomplish sanitization or detection.

A simple path planning scenario was implemented in this work as a proof of concept for using the particle from a track in this method. In this scenario, a blue platform must maneuver from one location to another while attempting to avoid detection. Waypoints in the environment were defined as the centers of the 10x10 display grids. A number of red platforms were placed in the environment in patrol areas with pseudo-random radiation plans. The start of one such scenario is show below. The blue platform is at the lower left corner of the display and has planned an initial path to reach the upper right corner. There are three red platforms, all stationary, with overlapping sensor arcs.

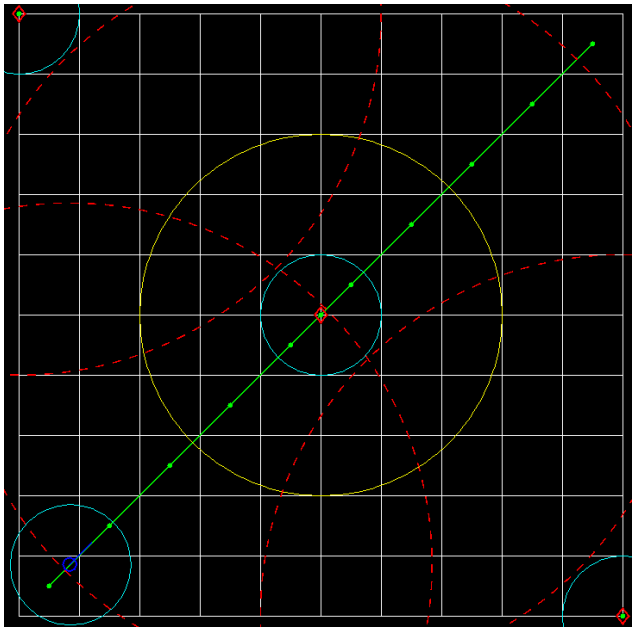


Figure 57. Initial Setup in Path Planning Example

Changes to the blue platform's path will occur when it obtains tracks on any of the three entities. While initial detections will always result in a new path plan, continued detections of entities will only result in a new plan if the new track is significantly different than a previously held track. This difference is calculated in the same way that sporadic communication is scheduled in (Klaas *et al.* 2005). Additionally, when the sum of the un-normalized weights of particles in a track drops below a threshold, the path will be re-planned ignoring the lost track.

Track information is taken into account in a cost function used during path planning. The A^* algorithm is used to find a path over the grid when a new plan is triggered. The estimate of remaining distance in the path is the straight line distance from a given grid node to the goal position. Cost incurred is a sum of the distance already traveled to reach a given point and the number of particles which fall into the grid square. The first change to the plan path which occurs when the blue platform acquires a passive track on the middle red platform is shown below.

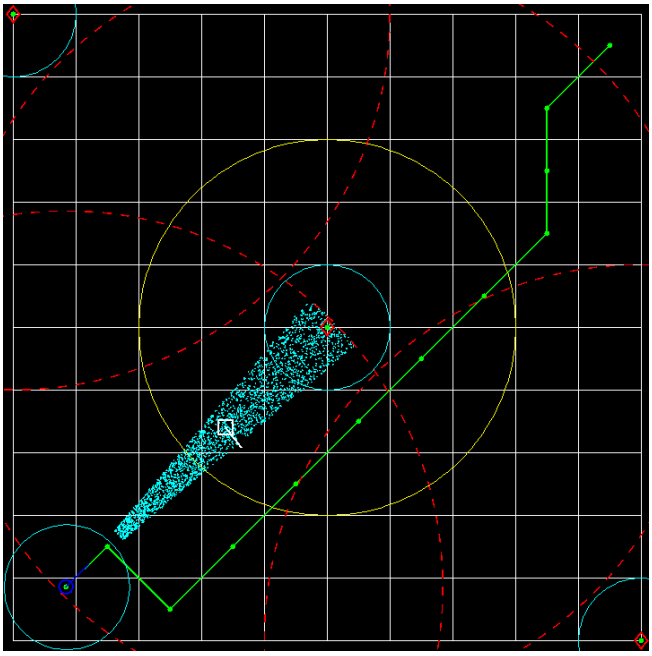


Figure 58. Path Planned Based on “Binned” Particle Positions

As the blue platform continues along this planned path, it will eventually gain a passive track on the red platform in the lower right corner. This will cause another change in the planned path as shown below. Due to the extremely simplistic cost function, the blue platform will blunder into the center red platform’s radar range.

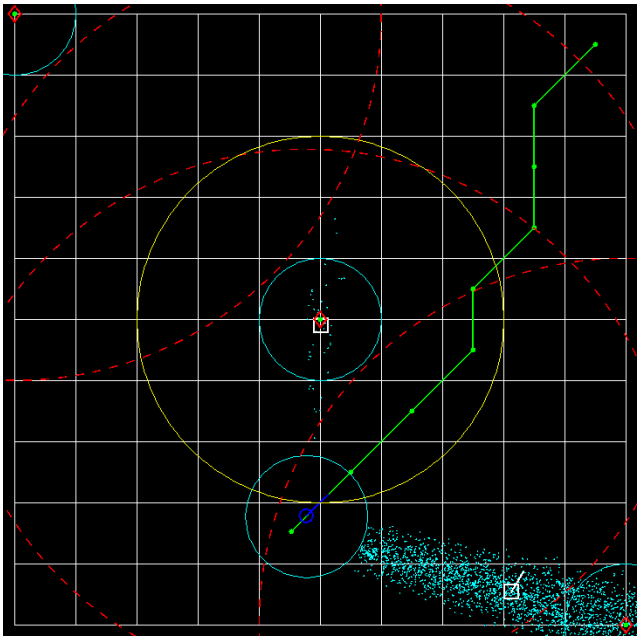


Figure 59. Altered Path Planned Following Additional Detection

While the above example of path planning is extremely simple in nature, it demonstrates the usefulness of the particle tracking method in representing areas of uncertainty as opposed to a discrete estimated position. For planning methods which are more complex, it has already been shown that the particle filter itself provides a metric for deciding when to re-plan using an expensive algorithm versus continuing to carry out the old plan. In a large environment with many entities having shared sensor states (not an uncommon occurrence in military simulations) this ability could significantly reduce the amount of wasted re-planning computation.

If the decision making involves determining the likelihood of a tracked entity being near a spot in the environment at a given time, a kernel method could be employed to test the probability of that spot being from the particle distribution. The Parzen-windows approach to density estimate which has been used to visualize the state of particle filters at certain points is a simple example of a kernel method. One of the advantages of working with particle filters as described in this work is that they can be advanced in time. This would allow a current track to be used to take into account the travel time of an entity in question to reach a certain point and the likelihood of the opposing entity being near the same point.

IV. ANALYSIS OF DIFFERENT TRACKING METHODS

A. BASIS FOR TRACKING ANALYSIS

Through the course of implementing and tweaking the particle tracking techniques described above, it became evident that particle tracking techniques were capable of capturing track uncertainty and computing accurate estimated positions in a wide variety of circumstances. The tracks obtained using the described techniques "looked right" to the eyes of both those with extensive and those with non-existent surface warfare experience. Unfortunately, an agent in a simulation would not have the ability to "look" at the state of its particle track in the manner of those watching of taking part in a simulation. The extent of an agent's knowledge about the state of a tracked entity would consist entirely on the estimated positions and particle distributions resultant from continued application of the particle track update algorithm.

To that effect the relative accuracies of the different tracking methods were analyzed in four different scenarios which presented tracking problems of varying types and difficulties. Two scenarios were designed to test the particle tracks' accuracy at determining target location, course, and speed through the use of active sensors. Those scenarios and resulting analysis of the different track types are contained in the *active tracking* section below. Two scenarios were designed to test track estimated position, course, and speed accuracy on tracks obtained wholly from passive means. Those scenarios and corresponding analysis can be found in the *passive tracking* section below.

B. ACTIVE TRACKING

The active tracking capabilities of the methods described above all have comparable performance. Due to the low bearing and range ambiguity of active sensors in the simulation, entities are able to obtain an accurate position with the first sensor sweep. Accurate courses and speeds of tracked entities take a series of sensor sweeps to obtain, but are readily available after a short period of time. The accuracy of the particle tracking techniques described in this work in an active tracking context was tested in two simple scenarios. The first scenario, the *run to the south*, tested these abilities on a non-

maneuvering target. The second scenario, *southern zig-zag*, tested the active tracking capabilities of the particle tracking methods on a maneuvering target. The results from these tests are detailed below.

1. Run to the South Scenario

As an initial test of active tracking capabilities a scenario was constructed in which a blue and red platform start abreast and proceed to the south with constant headings and speeds. The blue platform continually operates its radar during the run with the red platform being inside detection range. The scenario continues until the blue track of the red platform satisfies several criteria. These criteria are a distance between the estimated and actual location of the red platform of less than 0.1 simulation units, estimated heading within 2.5 degrees of actual heading, and estimated speed within one speed unit. A visualization of this scenario is shown below.

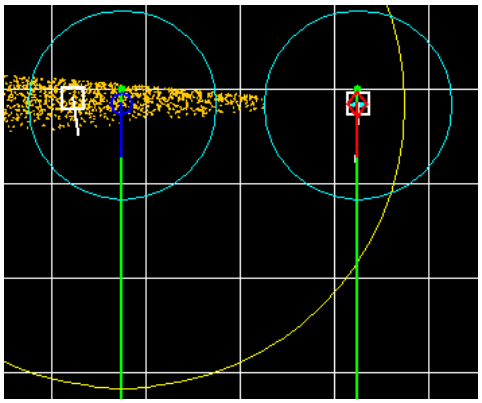


Figure 60. Run to the South Scenario Visualization

This scenario was run with six different track configurations, three using naïve particles and three using general contextual particles. The parameters of the particle tracks during these runs are shown in the tables below.

Naïve Particle Track Parameters			
$N_{\max}$	r_P	r_B	R_B
2500	0.75	0.25	Weighted Position Estimated Hdg/Spd Combined

Table 5. Naïve Particle Track Parameters for Scenarios

General Contextual Particle Track Parameters			
$N_{\max}$	r_P	r_B	R_B
2500	0.75	0.10	Weighted Position Estimated Hdg/Spd Combined

Table 6. General Contextual Particle Track Parameters for Scenarios

Both the red and the blue platforms were given an identical course and speed of 180 degrees at a speed of twenty-two. The scenario was run 100 times for each particle track variety. The location aspect of the criteria was satisfied on the first sensor sweep for every track variety. The performances of the tracks in satisfying all three criteria at the same time are shown in the table below. While the average times vary somewhat due to several outliers, the median times for satisfying all of the criteria are all around 1.3 simulation minutes. All the tracking methods perform similarly in this active tracking context ($p = 0.19$). With a radar sensor sweep cycle of 0.02 simulation minutes this equates to around sixty-five radar sweeps to meet all the track accuracy criteria mentioned above.

	Bulk Repopulation Method	Average Time	Median Time
Naïve	Weighted Position	4.85	1.34
	Estimated Heading and Speed	9.07	1.32
	Combined	3.87	1.34
Context	Weighted Position	8.06	1.34
	Estimated Heading and Speed	4.61	1.30
	Combined	1.90	1.34

Table 7. Average and Median Criteria Satisfaction Times in Run to the South Scenario

A time-lapsed Parzen-windows approximation of the particle track heading and speed distribution shows how the continued active detections drive the particles in the track to correspond to the actual target's course and speed. The visualizations shown below were taken from one run of a naïve particle track using the combined bulk replacement method. In this particular run the accuracy criteria were met in 1.25 simulation minutes. Recall that the actual target heading and speed in this scenario is 180 at a speed of twenty-two simulation knots.

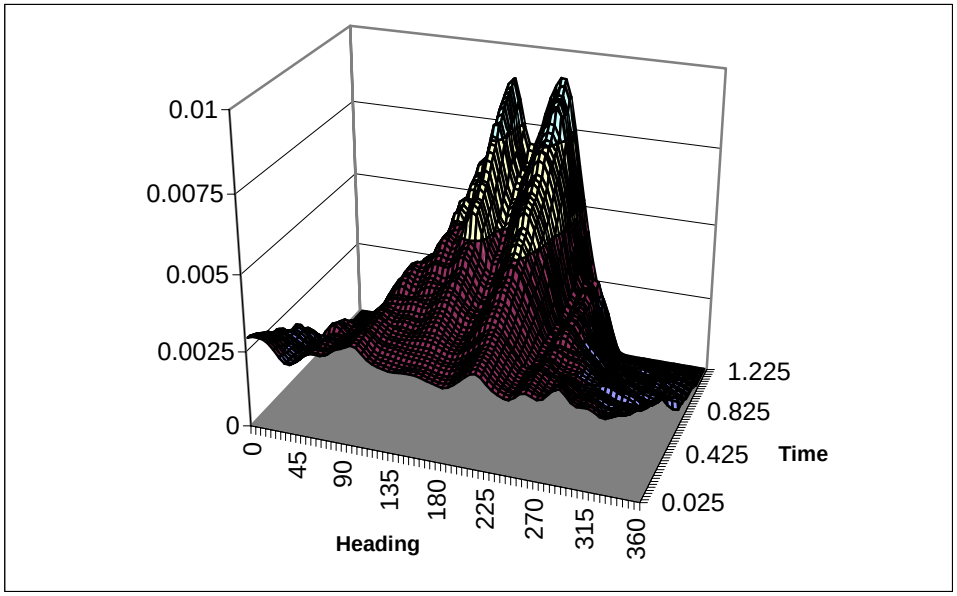


Figure 61. Parzen-Windows Approximation of Time Lapsed Particle Track Heading Distribution (Window Width 7.2)

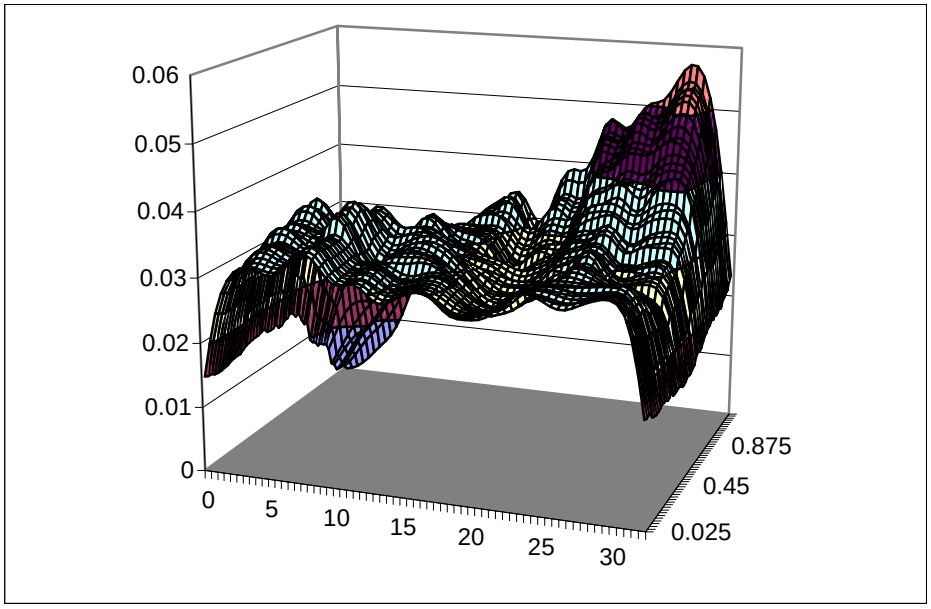


Figure 62. Parzen-Windows Approximation of Time Lapsed Particle Track Speed Distribution (Window Width 1.0)

2. Southern Zig-Zag Scenario

The above scenario, while showing that the different repopulation methods perform similarly in an active tracking context, is not very interesting because the target being tracked is not maneuvering. A scenario similar to the run to the south was created to test the particle tracks' abilities to accurately track a maneuvering target. The red

platform was given a series of southerly heading changes and an increased speed to allow it to remain within the radar footprint of the blue platform. A visualization of the southern "zig-zag" scenario is shown below.

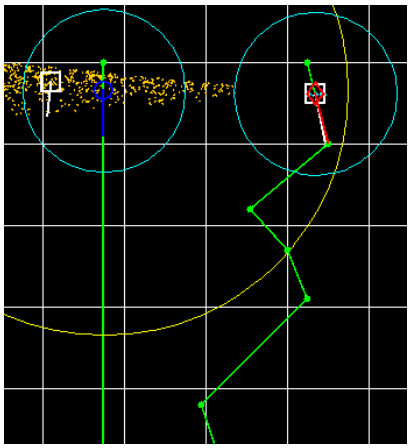


Figure 63. Southern Zig-Zag Visualization

The southern zig-zag scenario was run for two hours (120 minutes) of simulation time. The charts and tables below are based on the averaged results from 100 runs of this scenario for each tracking method. Track parameters were the same as those in the run to the south scenario. Comparisons between the estimated and actual state of the red platform were calculated every five minutes of simulation time. The estimated locations of the red platform were extremely accurate due to the low bearing and range ambiguity of the sensor in this simulation. The heading and speed estimates were accurate while the red platform was steady on a course and inaccurate in the periods immediately following a course change.

The chart below shows the course accuracy of the blue platform's track of the red platform. The performance of the different track types appears similar in performance. The peaks of inaccuracy in estimated heading correspond to the red platform's course changes. Following these peaks, the estimates get progressively more accurate as the red platform maintains its course and speed for longer periods of time.

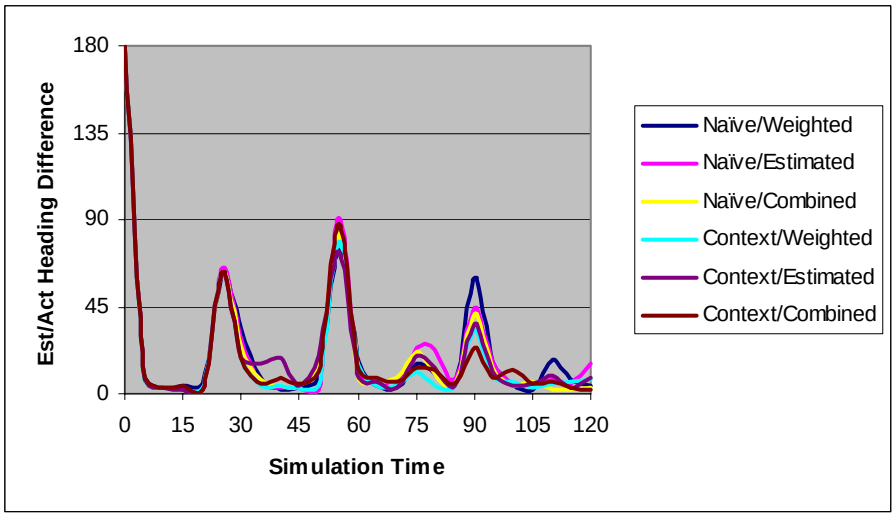


Figure 64. Difference between Estimated and Actual Red Platform Heading in Southern Zig-Zag Scenario

A table showing the average difference in estimated target course throughout the scenario illustrates that some of the tracking methods are better than others at capturing the heading of a maneuvering target. The letters denote membership in a group of tracking techniques which share statistically similar performance ($p > .05$). Thus all the tracking methods performed relatively similarly in estimating heading in this scenario with the exception of the general contextual particle track utilizing the weighted position bulk replacement method.

	Bulk Repopulation Method	Average Hdg Difference	Group
Naïve	Weighted Position	15.78	A
	Estimated Hdg/Spd	16.18	A
	Combined	14.43	A
Context	Weighted Position	12.70	B
	Estimated Hdg/Spd	14.99	A
	Combined	14.08	A

Table 8. Average Heading Accuracy of Different Track Types in Southern Zig-Zag Scenario

A table showing the average difference in estimated target speed throughout the scenario indicates that there was some difference in speed estimation capability between different tracking types. While all the track types maintained an average speed inaccuracy of around two speed units, one group of methods achieved slightly better

results. The group letters on the table below show the types of tracking methods which performed with similar accuracy ($p > .05$).

	Bulk Repopulation Method	Average Spd Difference	Group
Naïve	Weighted Position	1.79	A
	Estimated Hdg/Spd	1.84	A
	Combined	1.90	A B
Context	Weighted Position	1.85	A
	Estimated Hdg/Spd	2.08	B
	Combined	2.22	B

Table 9. Average Speed Accuracy of Different Track Types in Southern Zig-Zag Scenario

A time-lapsed parzen-windows approximation of the particle track heading and speed distribution through one run of the simulation are shown below. The visualizations shown are taken from a simulation run in which the track is a naïve particle track using the combined bulk repopulation method. The heading visualization is shown "flipped" so that the start of simulation is at the top of the display and the end of the simulation is at the bottom. This facilitates comparison to the red platform's patrol plan in the simulation window. The "gaps" in the distribution correspond to the red platform's course changes through the course of the scenario, and illustrates the time periods during which old particle track information is being adjusted to fit the new heading of the tracked target.

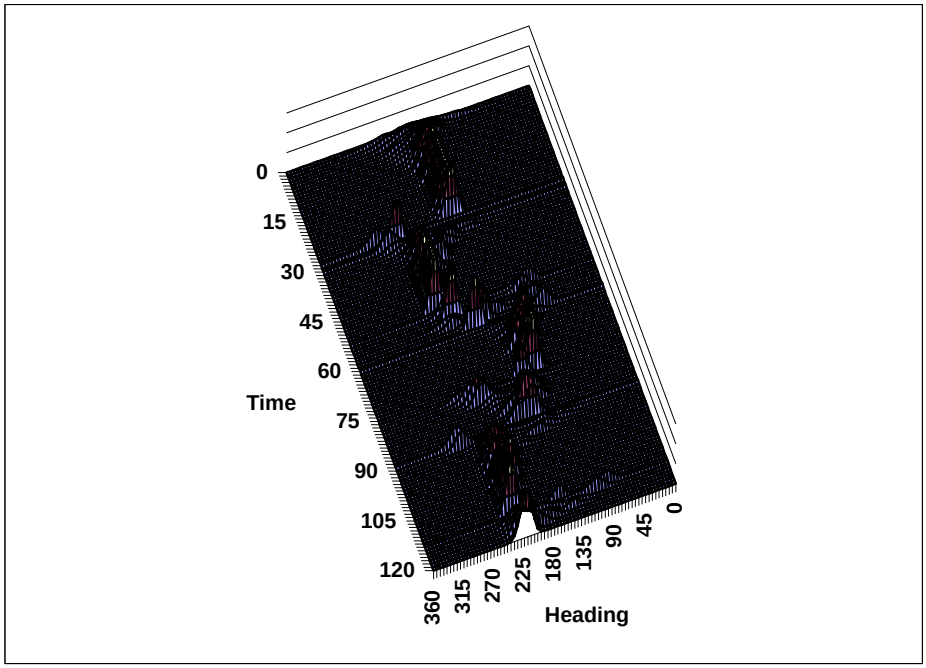


Figure 65. Time Lapsed Parzen-Windows Approximation of Particle Track Heading Density (Window Width 7.2)

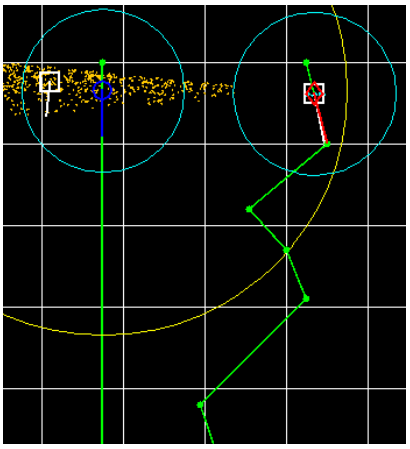


Figure 66. Southern Zig-Zag Scenarion Visualization for Comparison with Parzen-Windows Approximation of same Scenario

The time-lapsed parzen-windows approximation of the speed distribution in the particle track also shows these periods of track adjustment. This display has also been flipped so that the start of the simulation is at the top of the display. The periods of track speed adjustment correspond to those in the heading display above. Prior to and after these periods, the track displays an accurate estimated of the course's actual speed of twenty-seven (shown at the left due to the orientation of the graph).

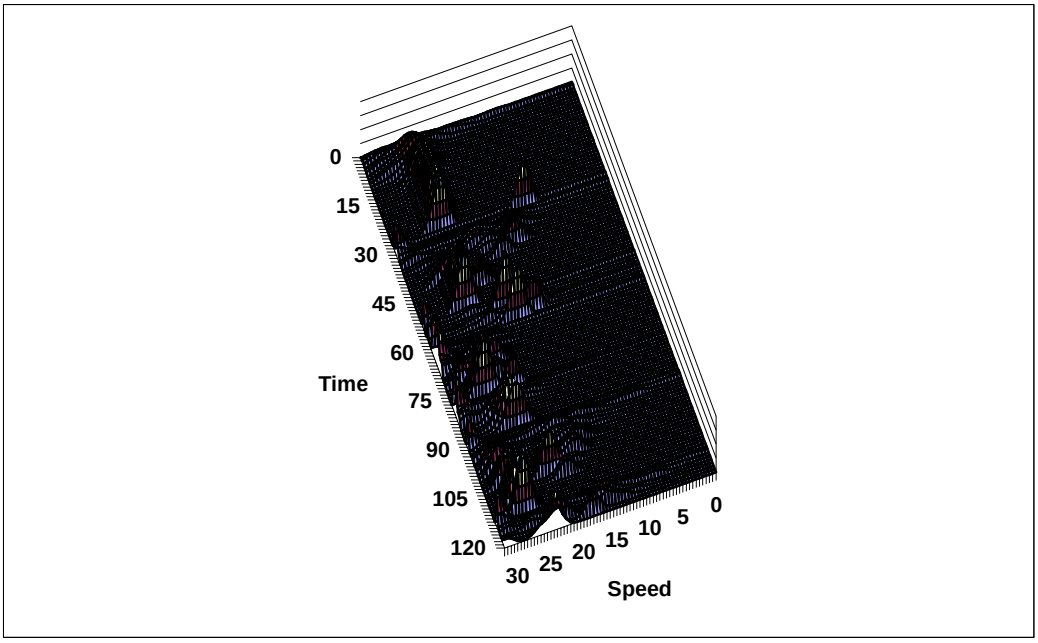


Figure 67. Time Lapsed Parzen-Windows Approximation of Particle Track Speed Density (Window Width 1.0)

While there were some minor differences in the ability of the different tracking types to estimate the course and speed of maneuvering targets, all of the techniques demonstrate the ability to accurately track targets through active means. The sections below will explore the ability of these same techniques to accurately portray a series of passive detections.

C. PASSIVE TRACKING

Tracking another entity through passive means presents significant challenges to the tracking platform due to the high ambiguity of the associated detections. The inexact nature of passive detections can be offset to a certain extent by intelligent maneuvering of the tracking platform to leverage the more exact bearing information of these detections as compared to the lack of accurate range information. However, complimentary maneuvers by the tracked platform can negate the effect of these disambiguation course changes. If the tracked platform can induce a situation where there is no bearing shift from the tracking platform's point of view, passive tracking becomes very difficult. The first scenario used to test passive tracking, the *closing* scenario, tests passive tracking on

a target with no bearing shift. The second scenario, the *triangulation* scenario, tests passive tracking in a situation with a large amount of bearing shift.

1. Closing Scenario

The first scenario evaluated the ability of these particle tracking techniques to discern accurate estimated target state from a situation with no discernable bearing shift. In this scenario a red platform will move towards a stationary blue platform operating its radar at preset time intervals. The red platform started out of counter-detection range of the blue platform eventually entering visual detection range of the blue platform. A visualization of this closing scenario is shown below.

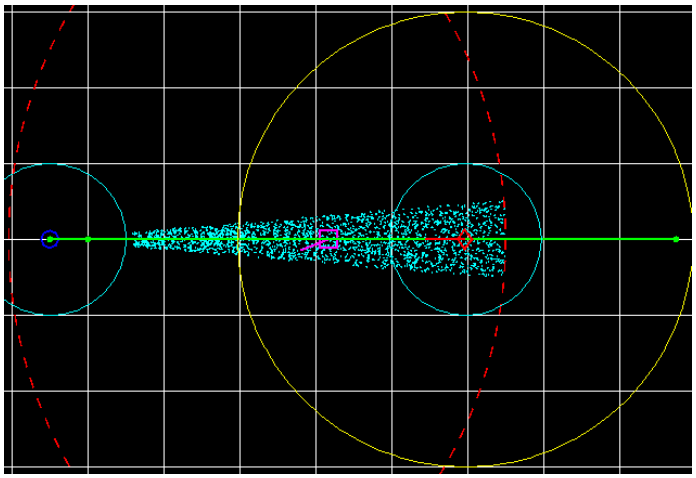


Figure 68. Closing Scenario Visualization

The actual course and speed of the red platform was 270 degrees at a speed of twenty. The red platform operated its radar for five simulation minute intervals, beginning at the start of the scenario and with fifteen minute periods of radar silence in between subsequent radiation periods. The scenario was run for four simulation hours with track accuracy statistics collected every five minutes. All of the charts and tables below with the exception of the parzen-windows approximations present the average data from 100 simulation runs with each track type. The red platform entered the blue platform's counter-detection range eighty-five minutes into the scenario and entered visual range of the blue platform two-hundred-fifteen minutes into the scenario. The scenario was run with ten different track types. The first three track types were naïve particle tracks utilizing the weighted position, estimated heading and speed, and

combined bulk repopulation methods. The second three track types were general contextual particle tracks utilizing the same three bulk replacement methods. The last four were four varieties of a transitioning contextual particle filter. The parameters for those track types are shown below. The transitioned behavior used in the closing scenario was a "seeker" behavior. When a contextual particle transitions to this behavior, it will maintain its current speed while changing course to intercept the position of the platform owning the track.

Transitioning Contextual Particle Track Parameters

$N_{\max}$	r_P	r_B	R_B	p_{chg}	p_{trans}	$Behavior_{trans}$
2500	0.75	0.10	Weighted Position Estimated Hdg/Spd Combined	0.45 0.35	0.10 0.30	Seeker

Table 10. Transitioning Contextual Particle Track Parameters for Scenarios

While the positional accuracy of the different tracking types varied, they all displayed a similar pattern of changing estimates. This pattern is shown in the figure below. The four tracking types displayed all begin with an inaccurate position estimate when the red platform first comes into counter-detection range. Although this estimate appears to improve steadily until just after time 120, this is a reflection of the tracked platform moving from the outer portion of the counter-detection window to the middle portion. At these ranges the limited range disambiguation of the radar detector as modeled in this simulation cannot effectively reduce the size of the detection distribution used to disqualify particles. Therefore the estimated position is in the middle of the particle cloud, and as the platform moves to this location the estimated position gets very "accurate." This is followed by a general decreasing of positional accuracy as the red platform moves past the midpoint of the detection distribution towards the blue platform. At around simulation time 160, the range accuracy of the radar detector begins to have an effect on the size of the range distribution used for particle disqualification and this is the cause of the jagged appearance of the position estimates following this time. The portions with increasing accuracies correspond to the red platform's radar operation intervals while the increasing portions correspond to the intervals where the red platform

has silenced its radar. At the far right chart the red platform has entered visual detection range of the blue platform, and the track transitions to an active track.

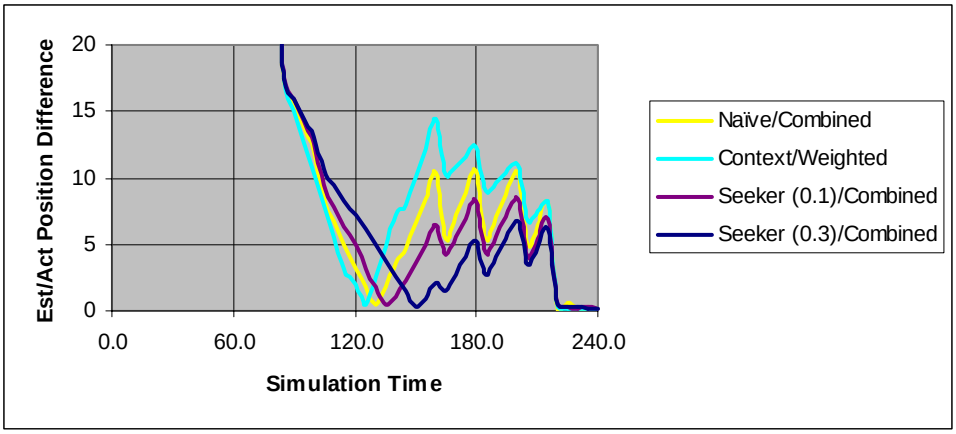


Figure 69. Difference Between Estimated and Actual Position in Closing Scenario

Average positional accuracy of the different tracking methods during the period of the scenario in which the red platform was being passively tracked (85.0 – 215.0) are presented in the table below. The transitional particle filter performed the best due to the accurate depiction of the red platform's movement. The general contextual particles performed the worst due to the ability for particles to choose random courses and speeds throughout the tracking problem while the naïve particle filter tends to repopulation only with particles that somewhat reflect the movement of the red platform. As the chart shows, track types performed similarly based on the type of particle in the tracks and that the bulk repopulation methods did not have a great effect on track accuracy.

	Bulk Repopulation Method	Avg Posit Difference	Group
Naïve	Weighted Position	7.60	A
	Estimated Hdg/Spd	7.70	A
	Combined	7.52	A
Contex	Weighted Position	8.94	B
	Estimated Hdg/Spd	8.91	B
	Combined	9.29	B
Seeker	Weighted Position	6.64	C
	Estimated Hdg/Spd	6.64	C
	Combined	6.13	C

Table 11. Average Position Accuracy of Different Track Types in Closing Scenario

While all three track types performed similarly with regards to positional accuracy through the closing scenario heading accuracy was heavily dependent on the type of particles used in the blue platform's track. The chart below shows the pattern of estimated heading accuracy through the course of the closing scenario. With the exception of the transitioning particle track, the tracking techniques are unable to obtain an accurate heading of the red platform until the limited range disambiguation capabilities of the radar detector are able to affect the size of the detection distributions around time 170.0.

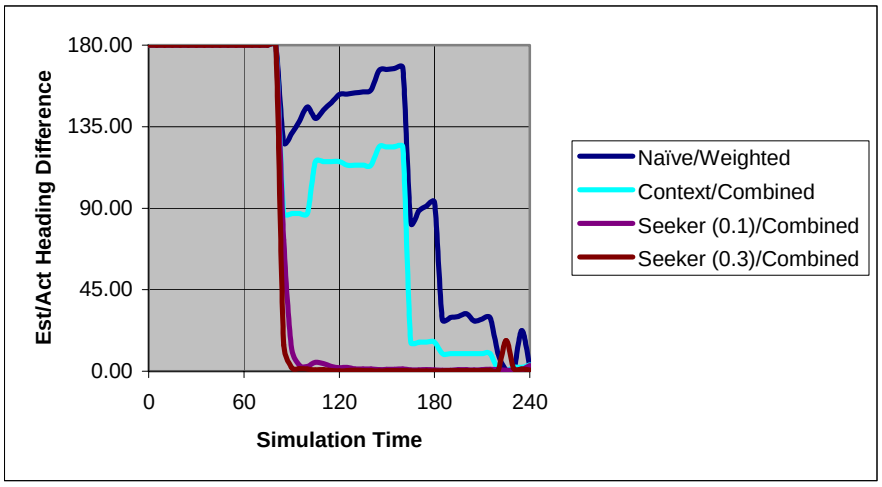


Figure 70. Difference Between Estimated and Actual Heading in Closing Scenario

A table containing the average heading accuracy of the different track types through the passive tracking portion of the closing scenario is shown below. Once again the bulk repopulation method use by the different track types did not have a significant effect on track accuracy. The transitioning particle track far outperformed the other two track types with the naïve particle track averaging more than ninety degrees off in estimated heading accuracy.

	Bulk Repopulation Method	Avg Hdg Difference	Group
Naïve	Weighted Position	110.02	A
	Estimated Hdg/Spd	116.38	A
	Combined	116.30	A
Context	Weighted Position	82.43	B
	Estimated Hdg/Spd	77.62	B
	Combined	70.17	B
Seeker	Weighted Position	3.64	C
	Estimated Hdg/Spd	4.47	C
	Combined	4.56	C

Table 12. Average Heading Accuracy of Different Track Types in Closing Scenario

The time-lapsed parzen-windows approximation of track heading distribution density below shows the difficulties encountered by the naïve and general contextual particle tracks in estimating accurate course information. In early portions of the scenario there are larger numbers of particles with headings of the actual course (270) and the reciprocal course (90). Due to the large range ambiguity of the radar detector particles on the reciprocal heading are retained in the track until the red platform is close enough for range-disambiguation to disqualify these reciprocal heading particles.

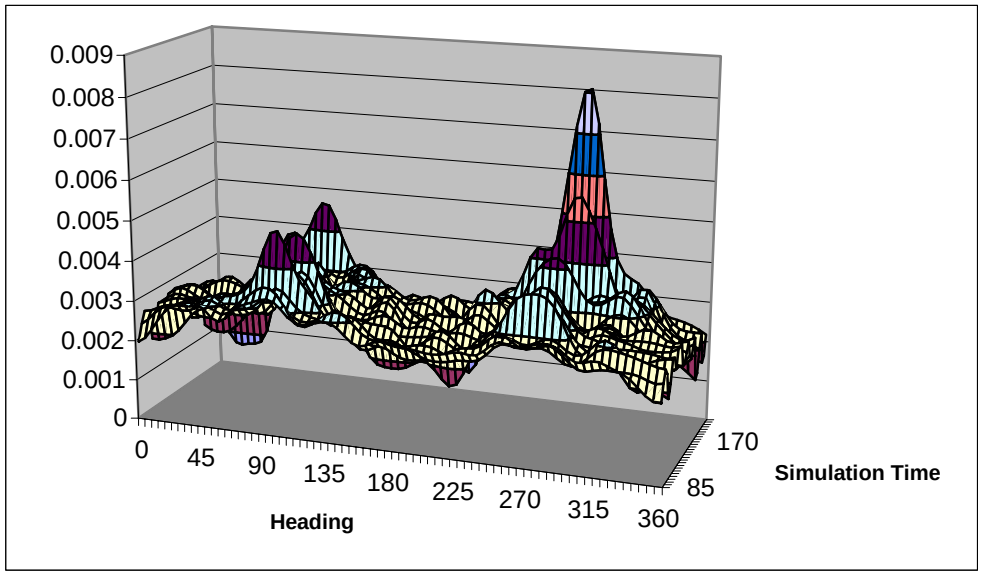


Figure 71. Time-Lapsed Parzen-Windows Approximation of General Contextual Particle Filter Heading Density (Window Width 7.2)

The transitioning particle track does not have the same problem due to the movement model of transitioned particles. The intercept course of transitioning particles and the small probability of disqualifying these particles results in a heading distribution density resembling that shown below.

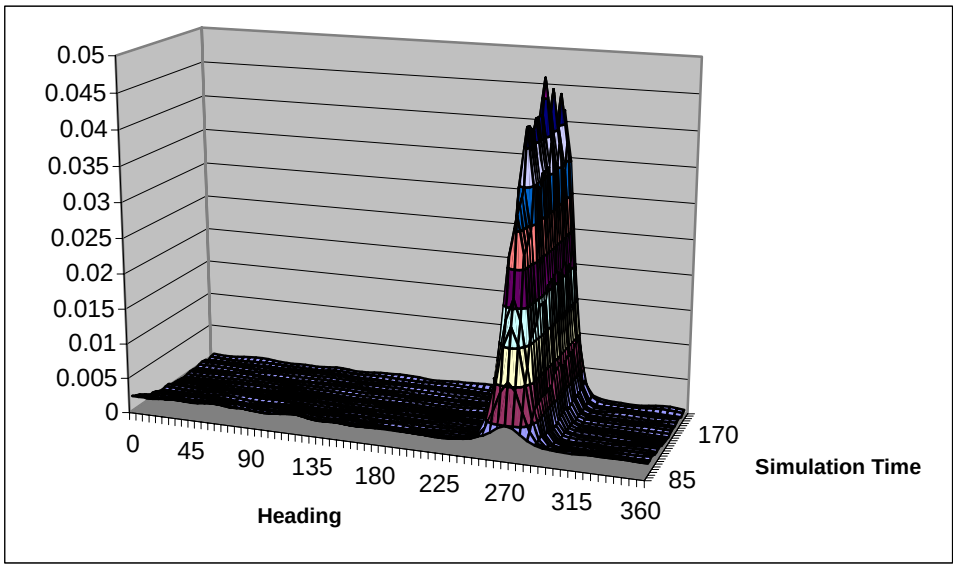


Figure 72. Time-Lapsed Parzen-Windows Approximation of Transitioning Contextual Particle Filter Heading Density (Window Width 7.2)

While the two contextual particle tracks exhibited similar performance in estimating the speed of the red platform, the naïve particle track performed dismally. As the chart below shows, initial speed estimates of the red platform during the passive tracking phase of the closing scenario are reasonably accurate. While the two contextual track types maintain this level of accuracy through the rest of the passive tracking problem, the naïve particle track gets progressively less accurate until the red platform enters the blue platform's visual detection range. This was due to the large number of particles with "slow" speeds which were duplicated during partial replacements. Slow particles were able to stay within the successive detection distributions regardless of their heading while faster particles were disqualified when their heading took them out of the detection distributions. Continued partial replacements duplicated the slow particles with higher and higher probabilities due to their increasing number in the track. This resulted in worsening speed estimation from the track as the tracking problem continued.

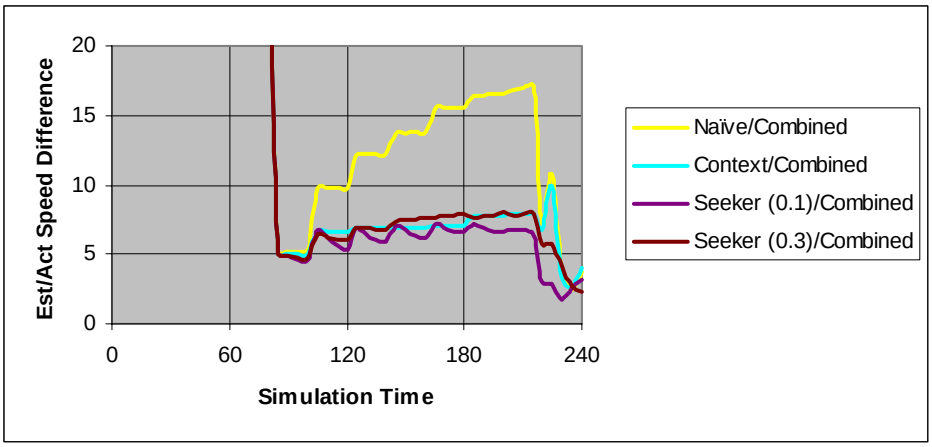


Figure 73. Difference Between Estimated and Actual Speed in Closing Scenario

A table showing the average speed accuracy of the different track types through the passive tracking portion of the scenario is provided below. Although the general and transitioning contextual particle filters performed similarly, the differences in overall performance were significant enough to yield statistically different performances. The naïve particle filter was unable to estimate the target speed with any degree of accuracy.

	Bulk Repopulation Method	Avg Spd Difference	Group
Naïve	Weighted Position	12.56	A
	Estimated Hdg/Spd	12.60	A
	Combined	12.69	A
Context	Weighted Position	6.92	B
	Estimated Hdg/Spd	6.91	B
	Combined	6.85	B
Seeker	Weighted Position	6.26	C
	Estimated Hdg/Spd	6.28	C
	Combined	6.26	C

Table 13. Average Speed Accuracy of Different Track Types in Closing Scenario

A time-lapsed Parzen-windows approximation of the speed distribution density of a naïve particle track through the passive tracking phase of the closing scenario shows a consistently bad estimated speed figure (actual target speed 20.0). This is due to the low range disambiguation ability of the radar detector mentioned above. All of the bulk replacement methods rely on prior estimated positions to some extent. As the estimated position remains pinned in the center of the line-of-bearing through most of the passive tracking phase, the estimated speed of the target will be close to zero. Any use of bulk

repopulation methods will render this bad speed estimate significantly hard to overcome without a means of radically altering the track configuration.

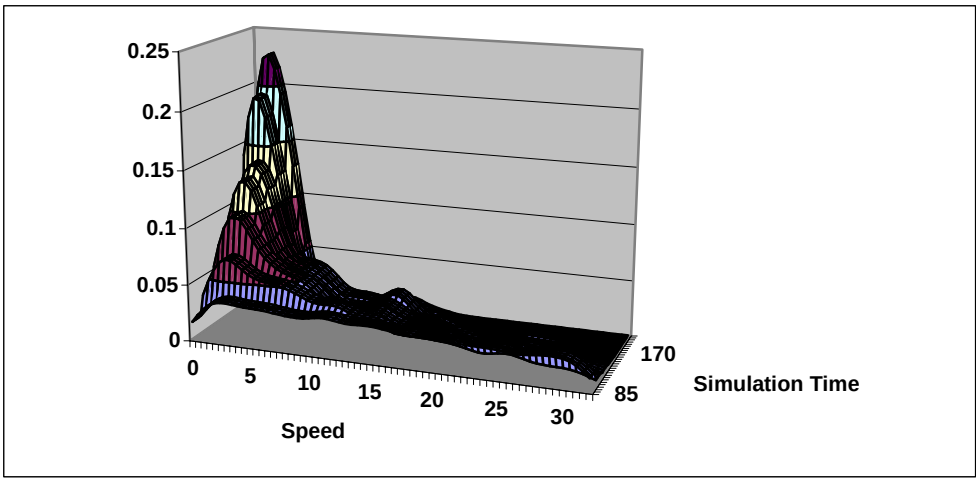


Figure 74. Time-Lapsed Parzen-Windows Approximation of Naïve Particle Filter Speed Density (Window Width 1.0)

By contrast, both of the contextual particle track types provide means to alter the nature of the particle track regardless of prior estimates. The general contextual track accomplishes this through random course changes to select particles and the transitioning contextual track accomplishes this through both random and directed course changes. A time-lapsed Parzen-windows approximation of the speed distribution density of a transitioning contextual track through the passive portion of the closing scenario is shown below. The random course and speed changes resulted in a more diverse speed distribution through the track with a "bad" mode at slower speeds due to the same circumstances described above and a "good" mode closer to the actual speed of the contact made possible by these two tracks' abilities to change the composition of the particle track.

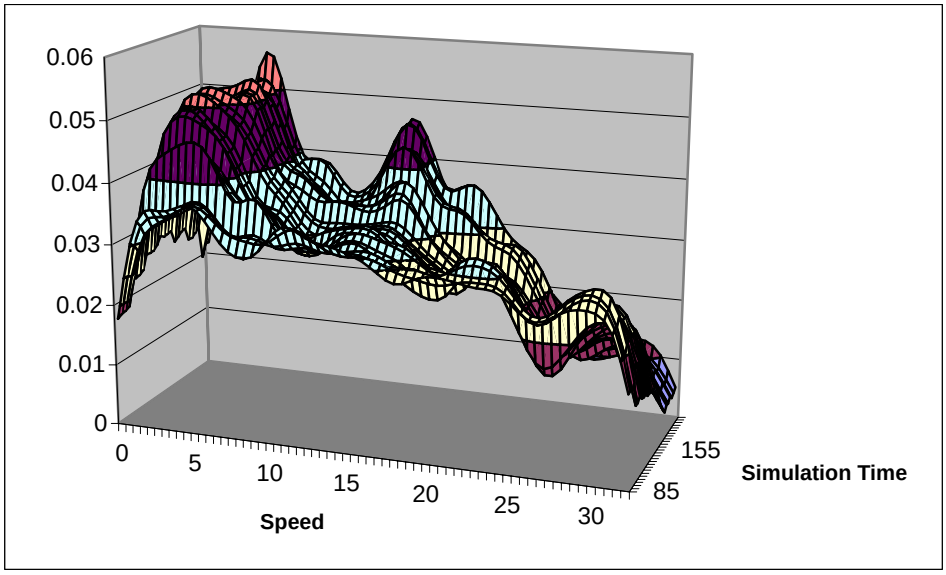


Figure 75. Time-Lapsed Parzen-Windows Approximation of Transitioning Contextual Particle Filter Speed Density (Window Width 1.0)

2. Triangulation Scenario

The results above show that even with prior knowledge of target behavior (as in the transitioning contextual particle track) the level of accuracy attainable using these particle tracking techniques in a zero-bearing-shift passive tracking problem is limited. To a certain extent this is comparable to real-life passive tracking problems in which an emphasis is placed on *imposing* bearing shift in similar situations through maneuvering to make the tracking problem easier. As the closing scenario examined the ability of the particle tracking techniques to achieve accurate estimates through passive tracking in a *worst case* scenario, another scenario was created to test the performance of the tracking techniques in the *best case*.

The triangulation scenario consists of a blue platform attempting to obtain a passive fix on a radiating red platform maintaining a constant course and speed. The same radiation plan was used for the red platform as in the closing scenario (five minute radiation periods with fifteen minute intervals of radar silence). The blue platform has a patrol plan which induces bearing shift while varying the range to the target. The course of the red platform is 90 degrees at a speed of five. A visualization of this scenario is shown below.

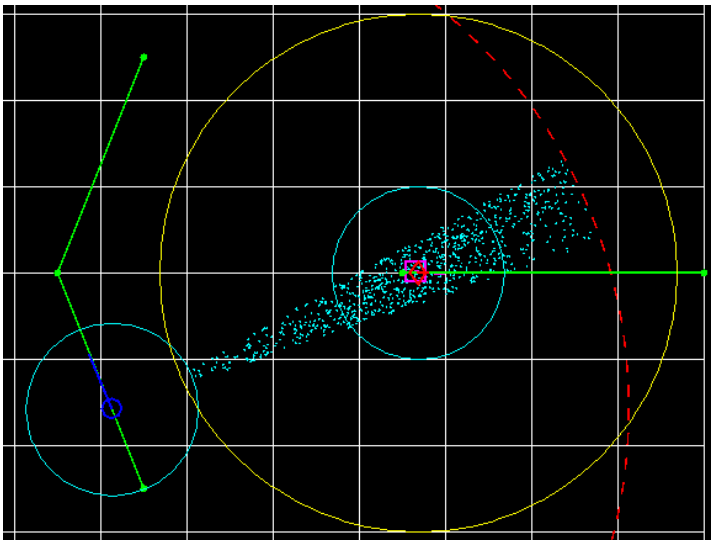


Figure 76. Triangulation Scenario Visualization

The triangulation scenario was run for 240 simulation minutes with track accuracy statistics gathered every five minutes. The charts and tables presented below display the average results from 100 simulation runs with each track type. The scenario was run with ten different track types. Three naïve and three general contextual tracks were used with identical parameters to shown above. The final four track types were transitioning contextual particle tracks with change and transition probabilities equal to those used in the closing scenario. Instead of a seeking behavior, a hiding behavior was used for transitioned particles. This behavior consists of a transitioned particle maintaining its speed while adopting a course which is within thirty degrees of the reciprocal bearing of the tracker.

The blue platform's position estimates of the red platform were generally accurate in this scenario. Like the position estimates in the closing scenario, the estimates in the triangulation scenario showed similar behavior across all track types. Initial estimates of the red platform's location were very accurate. These estimates became less accurate as the simulation progressed and the red platform reached the edge of counter-detection range. These estimates also displayed the saw-tooth pattern seen in the results of the closing scenario with low points corresponding to red platform radar operation and the increasing periods of inaccuracy corresponding to red platform radar silent periods. A chart showing the positional accuracy behavior of the different track types in the

triangulation scenario is shown below. Note that maximum end of the range scale is set at a distance of ten. In this simulation that distance corresponds to the visual range of platforms. Therefore all of the tracking techniques are sufficiently accurate to allow a tracking platform to get within visual range of the track platform.

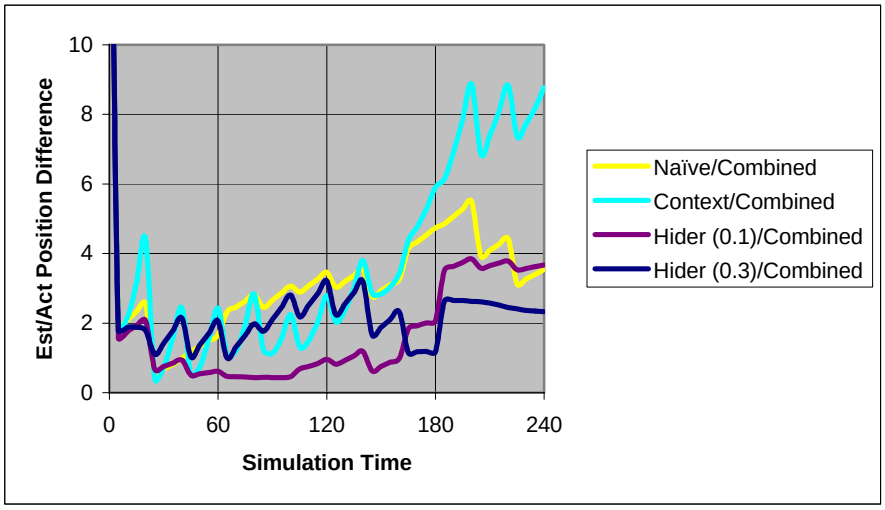


Figure 77. Difference Between Estimated and Actual Position in Triangulation Scenario

A table showing the average accuracy of the different track types through the triangulation scenario is shown below. The transitioning particle filter performed the best while the naïve and general contextual particle tracks performed similarly. The estimated heading and speed bulk replacement method perform significantly worse when pared with the naïve particle track. This was most likely due to the sub-par estimated positions obtained through passive tracking in the initial phases of the scenario.

	Bulk Repopulation Method	Avg Posit Difference	Group
Naïve	Weighted Position	3.10	A
	Estimated Hdg/Spd	3.59	A B
	Combined	3.03	A
Context	Weighted Position	3.75	B
	Estimated Hdg/Spd	3.79	B
	Combined	3.74	B
Hider	Weighted Position	1.67	C
	Estimated Hdg/Spd	1.73	C
	Combined	1.64	C

Table 14. Average Position Accuracy of Different Track Types in Triangulation Scenario

The heading accuracy of the naïve and transitioning particle tracks in the triangulation scenario were outstanding. The general contextual particle filter resulted in reasonably accurate initial heading estimates but suffered from increasingly inaccurate results through the course of the scenario. This was due to the random movement factor of the general contextual particles. While this movement feature was an asset in the closing scenario, in the triangulation scenario it resulted in inaccurate estimates due to the increasing size of the detection distributions allowing particles with inaccurate headings to remain in the track. The relative heading performance of the different track types through the course of the triangulation scenario are shown in the chart below.

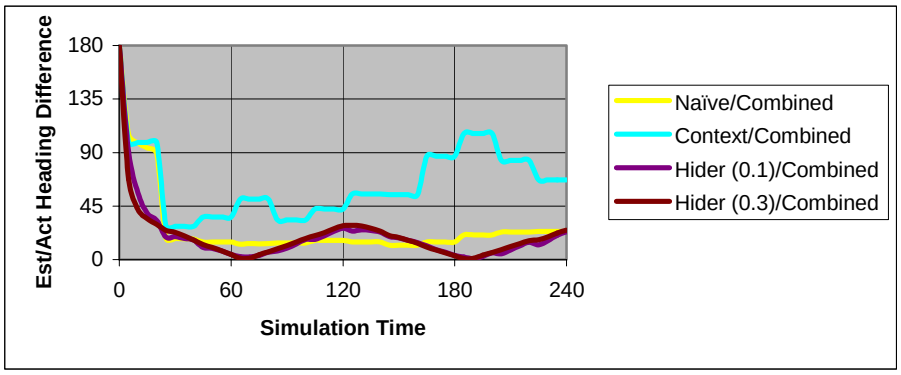


Figure 78. Difference Between Estimated and Actual Heading in Triangulation Scenario

The bow shape of the transitioning particle track is due to the imperfect reflection of the red platform's motion in the transitioned behavior of the hiding particles. The transitioned particles update their course to reflect the current bearing of the tracking platform from the particle. While this is a close approximation in this scenario, it is not perfect, particularly at the extreme ends of the patrol plan. A time-lapsed Parzen-windows approximation of the heading distribution density of a transitioning particle filter which illustrates this behavior is shown below.

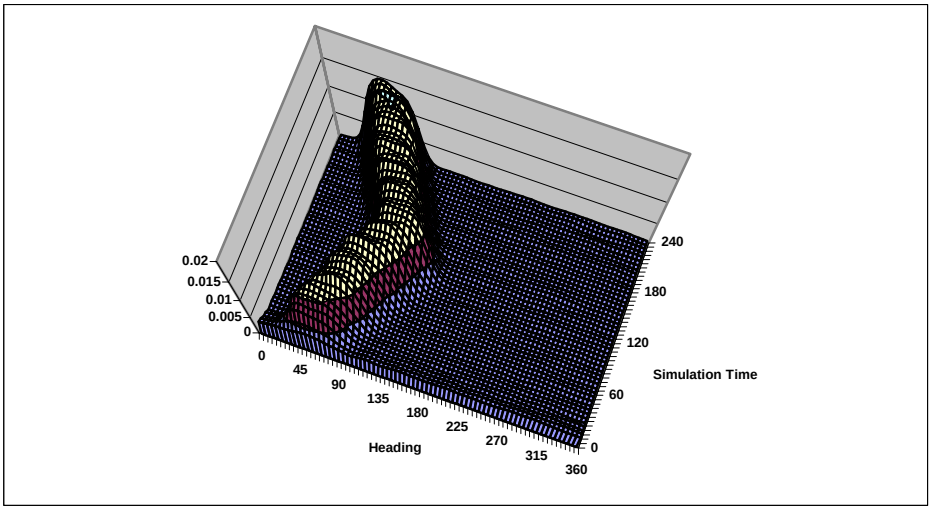


Figure 79. Time-Lapsed Parzen-Windows Approximation of Transitioning Contextual Particle Filter Heading Density (Window Width 7.2)

The behavior of a naïve particle track in the same scenario is much "noisier" but arrives at an accurate estimate nonetheless. A time-lapsed Parzen-windows approximation of the speed distribution density in a naïve particle track through the triangulation scenario is shown below. While the naïve particle track contains a number of particles on a reciprocal heading, the number of particles near the actual heading of the red platform is sufficient to result in accurate heading estimates.

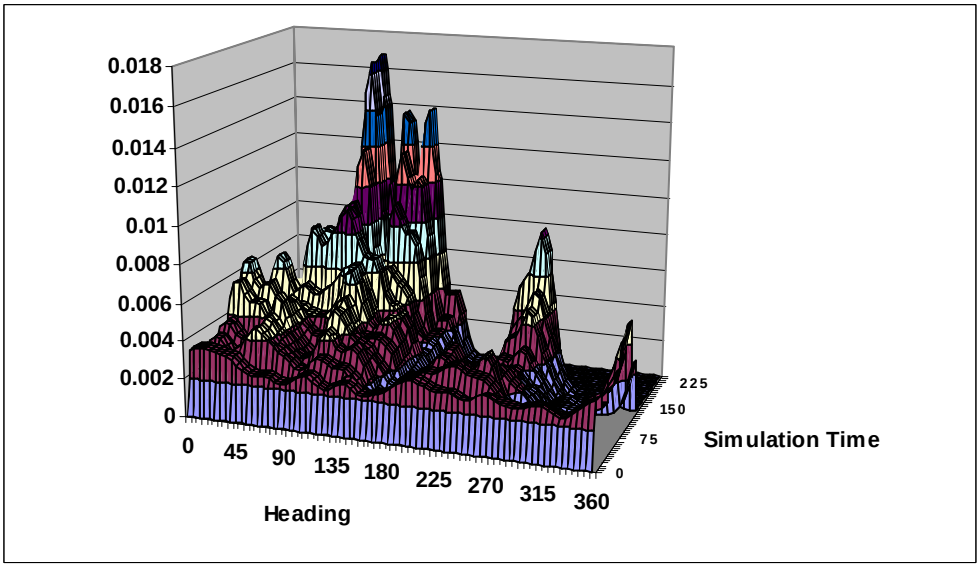


Figure 80. Time-Lapsed Parzen-Windows Approximation of Naive Particle Filter Heading Density (Window Width 7.2)

A table showing the average heading accuracy of the different track types in the triangulation scenario is provided below. Tracks consisting of like particle types performed similarly regardless of the bulk replacement method used. This was due to the radiation interval of the red platform which allowed partial repopulation to occur at most particle track update cycles. This resulted in use of the bulk repopulation methods in a very limited number of trials. The transitioning particle tracks performed with the most accuracy followed by the naïve particle tracks. The general contextual particle tracks performed with the least heading accuracy due to circumstances already described above.

	Bulk Repopulation Method	Avg Hdg Difference	Group
Naïve	Weighted Position	30.67	A
	Estimated Hdg/Spd	25.47	A
	Combined	23.49	A
Context	Weighted Position	62.71	B
	Estimated Hdg/Spd	64.13	B
	Combined	61.86	B
Hider	Weighted Position	16.39	C
	Estimated Hdg/Spd	16.04	C
	Combined	16.40	C

Table 15. Average Heading Accuracy of Different Track Types in Triangulation Scenario

The speed accuracy of the particle tracks in the triangulation scenario tended to increase with the amount of time available for observation. The only exception of this pattern was the general contextual particle track which maintained the same level of speed accuracy throughout the scenario. The speed estimation behavior of the different particle tracks is shown in the chart below.

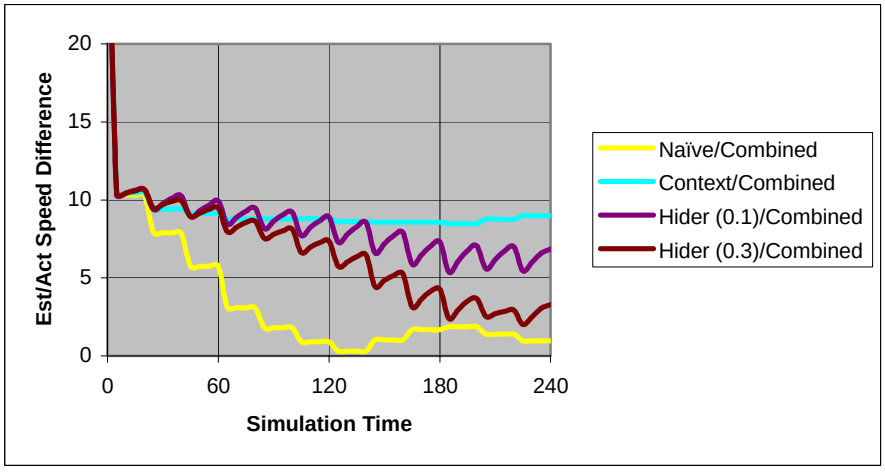


Figure 81. Difference Between Estimated and Actual Speed in Triangulation Scenario

A table showing the average speed accuracy of the different track types in the triangulation scenario is provided below. The two contextual particle filters performed similarly while the naïve particle filters performed the best by a significant margin.

	Bulk Repopulation Method	Avg Spd Difference	Group
Naïve	Weighted Position	3.03	A
	Estimated Hdg/Spd	3.20	A
	Combined	3.08	A
Context	Weighted Position	8.97	B
	Estimated Hdg/Spd	8.91	B
	Combined	8.95	B
Hider	Weighted Position	8.06	B
	Estimated Hdg/Spd	8.11	B
	Combined	8.08	B

Table 16. Average Speed Accuracy of Different Track Types in Triangulation Scenario

The results above show that the ability of the particle tracking techniques to obtain accurate state information on entities through passive means relies on the ability to induce bearing and/or range shift on the target. This requirement does not reflect a weakness of the particle tracks but the general difficulty of tracking other platforms through solely passive means. To this effect the particle tracks reflect the actual difficulties encountered in tracking real-world entities through inexact passive means.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

Through the course of this work it became evident that although particle filter tracking techniques present a novel and flexible means of representing target state in a variety of environments their immediate usefulness in general defense simulation pursuits has limitations. Although the naïve particle filter technique described in this work displayed an ability to accurately determine target heading and speed in both active and passive tracking contexts doing so required a large number of particles in the track. While large numbers of particles per track was not an issue in this simple simulation, in large-scale defense simulations with large entity counts and communication schemes this requirement would lead to a larger computational overhead if naïve particle schemes were implemented. This need could be addressed by using particle tracking techniques only for positional estimation and then applying simple target motion analysis procedures to the series of locations obtained from the tracks. Position estimation can be achieved through particle tracks with a significantly smaller particle count in individual tracks.

The transitioning particle filter tracking technique proposed in this work is also a possible method of alleviating the need for large particle counts to achieve accurate heading and speed information. In the two passive tracking scenarios detailed in the analysis chapter of this work the transitioning particle filters performed superbly in comparison to the naïve and general contextual particle tracks. In the closing scenario, which represented a worst case tracking problem, the transitioning contextual particle filter presented the only means to accurately predict target heading and speed. In the triangulation scenario, although the transitioning particles were outperformed by the tracks consisting of naïve particles, they were able to achieve usable levels of accuracy despite the transitioning particles not directly representing the movement model of the platform being tracked. This seems to indicate that intelligently chosen simple movement models for transitioned particles could result in highly accurate state estimations of tracked entities while requiring a substantially lower particle count.

Although the particle tracking technique in its current state is not ready for use in large-scale military situations, in smaller scale environments with a small number of entities the technique presents a means to represent both active and passive tracks. In addition, in smaller and more complex environments with small particle count requirements transitioning contextual particles can be provided with movement and/or behavioral models closely approximating those used by the actual entities in the simulation. In this way computer controlled entities can be provided with limited information yet still have the ability to construct very detailed and accurate tracks of opposing entities. The visualization of the particle tracking method also shows potential to more accurately display uncertainty data to human decision makers. The appearance of detection distributions and their corresponding increased in uncertainty over time offer more discerning visualization of target uncertainty than current datums (which are expanding circles centered at old estimated positions.

B. FUTURE WORK

While the tracking techniques performed admirably as implemented in this work several avenues of possible extensions to improve or alter the utility of these methods became evident. Some of the more challenging and possibly rewarding direction for future study of particle based tracking methods in simulation environments are discussed briefly below.

1. Use in Actual Simulation

Perhaps the first extension would be to use the particle tracking technique in an environment with a more realistic sensor and detection representation. The techniques presented in this work were constructed using a generalized detection representation and as such should support a wide variety of detection types. . Prior to actually using this tracking method in a simulation it would be necessary to test its use with sensor models which result in dirty or completely misleading data. Additionally, use of the particle filter technique should be tested in an environment without perfect correlation of sensor information. An examination of the particle track's response to this data would need to be conducted to ensure that the methods result in appropriate uncertainty representations and state estimations within the environment.

The use of the particle tracking method in an actual simulation would also require a more strenuous examination of the types of information which can be obtained from a particle track. While the position, heading, and speed were relatively well explored in this work and the use of the individual particles to represent an area of uncertainty was discussed briefly, use in an actual simulation would require a more detailed analysis of ways to convert this type of information into states which could be used by a simulation engine. Due to the wide array of simulation suites used in the defense industry, the feasibility of using a particle tracking technique in defense simulations would need to be made on a case-by-case basis with the needs of the individual simulation taken into account.

2. Contextual Particle Behaviors

Although the general and transitioning contextual particle behaviors presented in this work were simple in nature, their inclusion into several of the test scenarios resulted in very accurate tracking information. These results seem to indicate that the development of more detailed or widely applicable individual particle behaviors could result in better tracking results with a decreased need for high particle counts. A more detailed analysis of the effects of transitioning particles on track accuracy including the use of complex behaviors needs to be completed prior to embracing this technique for use in future particle tracking systems.

The possibility of classifying track behavior through the use of the transitioning particle technique also exists. Given no prior knowledge of track behavior and a particle track made up of several different transitioning particle types the rates of disqualification of certain particle types could be compared to arrive at an estimate not only of the position, heading, and speed of the platform in question but of the tracked entity's behavior. These rates could result in changed transition rates for the different particle types to not only classify the target in question but increase the ability of the particle track to accurately represent the estimated state of the target. Continued application of these techniques could result in the ability of a track to fine tune its uncertainty representation through a variety of behaviors on the part of the tracked entity.

3. Additional Dimension

The particle tracking techniques reviewed in this work were based in a two-dimensional environment. The use of such tracking techniques to represent target uncertainty in a three-dimensional environment would require little modification if the only state information of interest were the location of the tracked entity. The addition of heading and speed information to items of interest would present a significant challenge to the naïve particle filter track types implemented in this work. This is due to the large array of possible headings for naïve particles in a three dimensional environment. If the same movement model and estimated position techniques were used in this environment a prohibitively large number of particles would be required to accurately capture the tracked entity's heading and speed.

This weakness of the methods described in this work could be addressed through the use of the particle filter tracking technique only for positional estimation while relying on other algorithms to extract heading and speed information from a series of estimated positions. This method could also be addressed through the contextual particle filter model with the use of a more restrictive movement model for individual particles. In either case a more comprehensive study of these techniques would be required to determine the usefulness of the particle filter tracking method in a three-dimensional environment.

LIST OF REFERENCES

Arulampalam, S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/Non-gaussian bayesian tracking. IEEE Transactions on Signal Processing, 50(2), 174-188.

Bererton, C. (2004). State estimation for game AI using particle filters. AAAI Workshop on Challenges in Game AI,

Buss, A. (2002). Simkit: Component based simulation modeling with simkit. WSC '02: Proceedings of the 34th Conference on Winter Simulation, San Diego, California. 243-249.

Buss, A. H., & Sanchez, P. J. (2005). Simple movement and detection in discrete event simulation. WSC '05: Proceedings of the 37th Conference on Winter Simulation, Orlando, Florida. 992-1000.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). Pattern classification (2nd ed.). New York: Wiley.

Isla, D. (2006). Probabilistic target tracking and search using occupancy maps. In S. Rabin (Ed.), AI game programming wisdom III (379-387). Massachusetts: Charles River Media.

Klaas, M., Southey, T., & Cheung, W. (2005). Particle-based communication among game agents. Artificial Intelligence and Interactive Digital Entertainment Conference, Marina del Rey, CA. 75-80.

Stout, B. (2000). The basics of A* for path planning. In M. DeLoura (Ed.), Game programming gems (254-262). Massachusetts: Charles River Media.

Stroupe, A. W., Martin, M. C., & Balch, T. R. (2001). Merging gaussian distributions for object localization in multi-robot systems. ISER '00: Experimental Robotics VII, 343-352.

van der Sterren, W. (2002). Tactical path-finding with A*. In D. Treglia (Ed.), Game programming gems 3 (294-306). Massachusetts: Charles River Media.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California