Target Discrimination Against Clutter Based on Unsupervised Clustering and Sequential Monte Carlo Tracking

by

Matthew Gregory Freeman

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2016 by the
Graduate Supervisory Committee:

Antonia Papandreou-Suppappola, Co-Chair
Daniel Bliss, Co-Chair
Chaitali Chakrabarti

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

The radar performance of detecting a target and estimating its parameters can deteriorate rapidly in the presence of high clutter. This is because radar measurements due to clutter returns can be falsely detected as if originating from the actual target. Various data association methods and multiple hypothesis filtering approaches have been considered to solve this problem. Such methods, however, can be computationally intensive for real time radar processing. This work proposes a new approach that is based on the unsupervised clustering of target and clutter detections before target tracking using particle filtering. In particular, Gaussian mixture modeling is first used to separate detections into two Gaussian distinct mixtures. Using eigenvector analysis, the eccentricity of the covariance matrices of the Gaussian mixtures are computed and compared to threshold values that are obtained *a priori*. The thresholding allows only target detections to be used for target tracking. Simulations demonstrate the performance of the new algorithm and compare it with using k-means for clustering instead of Gaussian mixture modeling.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation

Estimation theory is fundamental to many fields ranging from economics to engineering. The theory itself can be divided into two broad categories: classical estimation theory and Bayesian estimation theory. In classical estimation theory, the parameter to be estimated is assumed a fixed value embedded in the noisy observations. In Bayesian estimation theory, the parameter to be estimated is considered a random variable. These two distinct categories result in two different approaches to estimation. Experiments performed under the classical approach essentially draw samples from the density that represents the uncertainty and combine it with the parameter. Experiments performed under the Bayesian paradigm will draw the parameter to be estimated from it's own distribution and then combine it with the noise drawn from it's distribution. The Bayesian approach to estimation will always result in a minimum mean-squared error estimator, unlike the classical approach [1]. In the classical approach, a minimum unbiased estimator (MVU) may not necessarily exist for all values of the parameter. This tends to be a problem in classical estimation due to the dependence of the estimator on the parameter to be estimated in order to minimize the mean squared error. The ability of radar processing to separate a target from clutter is fundamental to target tracking. In a radar track mode, the tracking engine uses observations from the signal processor in order to maintain a valid track on a target. The signal processor has to have the ability to separate the target from unwanted signal returns with a high degree of confidence. In target recognition and

classification, the signal processing has to discriminate between various objects in the scene such as different targets and clutter. Adverse affects such as misclassification can result if the discrimination yields poor performance. The purpose of this work is to investigate the use of unsupervised clustering with sequential Montel Carlo methods, such as particle filtering, to increase the discrimination performance between targets and clutter. In particular, a clustering algorithm based on Gaussian mixture modeling is used to separate a target from clutter for the purpose of providing reliable track observables to a particle filter. Gaussian mixture modeling has previously been applied to the problem of target tracking in clutter. In [2], Gaussian mixture models learned from recorded data are used to classify radar tracks. A Gaussian mixture probability hypothesis density filter is used in [3] to estimate the number of targets and their unknown parameters in the presence of noisy measurements and clutter. In [4], measurement origin and target model uncertainty due to maneuvering target tracking in clutter by combining a multiple hypothesis tracker and a multiple model algorithm based on Gaussian mixture reduction. This reduction approach is used as it helps to reduce the exponentially increasing number of measurement association possibilities and target model trajectories. In [5], the problem of a single target tracking in clutter using a high pulse repetition frequency radar is considered by approximating each track trajectory probability density function as a Gaussian mixture.

1.2 Proposed Work

Estimation is central to many engineering problems. In this work, we propose two types of estimation techniques applied to two different problems in a common context. Unsupervised clustering will be applied to the problem of association where data samples have an underlying hidden grouping that needs to be estimated. This

grouping is due to the association of the data to a common source or sources. The clustering approach taken in this paper is the Gaussian Mixture Model (GMM-EM) and will be used to separate a target from clutter for the purpose of providing samples whose states can be tracked. These samples will be fed into a particle filter that will track the state of these samples as they evolve over time. As a means of comparison, a K-means (KM) clustering algorithm will be used as a secondary clustering technique with the results evaluated against the GMM-EM. The proposed algorithm uses the resulting detections from the RDM as input to GMM-EM for cluster separation. The purpose of the separation is to cluster detections that belong to the target from detections that belong to clutter. Both the target and clutter have range and Doppler extent but are separated by some distance in range and Doppler space. The clustering algorithms work on these data points to separate the target from clutter. As a post processing procedure for differentiating between the mixture corresponding to the target and the mixture corresponding to clutter, an eigen-decomposition of the resulting Gaussian mixture covariances is performed. The eccentricity of the covariances associated with the two clusters is used to identify which is the target and which is the clutter. These target associated detections are then used to compute the centroid of the target in range and range-rate space (which are proportional to time delay and Doppler, respectively) . This centroid sample, which represents the target's position in range and range rate space at this collection time, is then used as an input to a particle filter tracker. The particle filter is responsible for tracking the state of the target as the target moves and provides estimates of the target's position and range rate at each time step.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2 and Chapter 3, we provide background information on the particle filter sequential Monte Carlo approach and on Gaussian mixture models, respectively. Our proposed method of separating target detections from clutter using Gaussian mixture modeling before tracking is described in Chapter 4. In Chapter 5, we provide simulation results to demonstrate the performance of the new approach and compare it with k-means clustering.

Chapter 2

PARTICLE FILTER

Particle filters are part of a larger class of filters known as Bayesian filters. Under the Bayesian philosophy, prior knowledge about a statistical process will tend to yield a more accurate estimator. This can be understood intuitively by considering the reduction of the sample space and re- weighting of random variables given prior knowledge. If it is known that a random variable of a distribution can assume only a subset of its original range, then the subset range(s) must be re-weighted in order to maintain a valid density function. This re-weighting will increase the probability of occurrence to the values that can occur and rule out the possibility of those that cannot, given this prior knowledge. The estimator, in essence, only generates estimates based on this reduced sample space, whereas the estimator that does not consider this sample space reduction will continue to produce estimates over the entire sample space, even though these values do not actually occur. The Bayesian approach does produce a biased estimator; however, the bias will improve the estimator performance [1]. The Bayesian mean square error (BMSE) of an estimator $\hat{A}$ can be expressed as

$$\text{BMSE}(\hat{A}) = \int < \int (A - \hat{A})^2 p(A|\mathbf{z}) dA > p(\mathbf{z}) d\mathbf{z} \tag{2.1}$$

where A is the parameter to be estimated and $\mathbf{z}$ are the measurements. To find the estimator that minimizes the BMSE, we take its gradient and set the result to zero and solve for the parameter estimator $\hat{A}$. Noticing that $p(\mathbf{z}) \geq 0$ and minimizing the portion of the equation in brackets results in

$$\frac{\partial}{\partial \hat{A}} \int (A - \hat{A})^2 p(A|\mathbf{z}) dA = 0 \tag{2.2}$$

Solving for the estimator $\hat{A}$ yields

$$\hat{A} = E(A|\mathbf{z}) \tag{2.3}$$

This equation implies that the optimal estimator that minimizes the BMSE in the Bayesian sense is simply the mean of the posterior PDF. The posterior PDF is the PDF of A after the data has been taken into consideration [1]. It should be noted that the Bayesian estimator $\hat{A}$ is treated as a random variable and as such is represented by a probability distribution. This is a completely different approach to estimation from the classical approach used in such techniques as Maximum Likelihood Estimation. The recursive Bayes filter is used extensively in state estimation in a variety of forms including the Kalman filter and the Particle filter. The Bayes filter is a framework for recursive state estimation with the Kalman filter and Particle filter as specific instances. In state estimation, the true state of a system is estimated given a set of noisy observations and a set of control inputs. By using Bayes theorem it is possible to determine the current state of the system if we have an estimate for the previous state and are given a current observation and a control command. An underlying assumption in the Bayes filter is that the system process is Markovian. A Markov process is a random process where the current state depends only on the prior state. In this type of process, the past and future state of the system are independent with the current state transitioning to the next state based on some rule. The probability of the true state, using the Markov assumption can be expressed as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, ..., \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.4}$$

with $\mathbf{x}$ defined as the true state at some time index t. This assumption shows that the current state is conditionally dependent only on the previous state and no other. The Bayes filter also assumes that the current observation depends only on the current

state and is independent of any prior states. This can be expressed as

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{x}_{t-1}..., \mathbf{x}_0) = p(\mathbf{z}_t|\mathbf{x}_t) \tag{2.5}$$

By denoting our belief in the current state of the system as $b(\mathbf{x}_t)$, the Bayes filter, given the assumptions above, can be expressed as

$$b(\mathbf{x}_t) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) b(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \tag{2.6}$$

where η is a normalization factor. This states that if we have an estimate for the previous state and are given a current observation and a control command $\mathbf{u}_t$, then we can determine the current state of the system. As can be seen above the prior belief is used in recursive form to compute the current belief. The Bayes filter can be be written as a two step process [6]. The two steps in this process are the prediction step and correction step. The prediction step uses the control command and prior state to predict the current state. This can be expressed as

Algorithm 1 Repeat for each incoming sample

Prediction Step: $b(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) b(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}$

Correction Step: $b(\mathbf{x}_t) = \eta p(\mathbf{z}_t|\mathbf{x}_t, b(\mathbf{x}_t)$

This algorithm is used for the Kalman filter and the Particle filter. Both filters make use of these prediction and correction cycles to refine the state estimate over time, albeit in different ways. This paper will focus on the Particle filter. The Particle filter is a Bayesian filter that can be used to represent arbitrary density functions. Representation of arbitrary densities is performed by using samples to represent the densities and iteratively refining these samples so that the densities are approximated. The Particle filter uses a non-parametric approach unlike other estimation techniques such as Maximum Likelihood Estimation. In the Particle filter algorithm a proposal distribution is iteratively refined to estimate the target distribution. This is done

through iterative prediction and correction cycles. The prediction of the state is accomplished by sampling from the proposal's distribution. The differences between the proposal and target distributions are accounted for by using corresponding weights that represent how likely a given sample is to come from the target distribution. When a sample from a proposal distribution has a low enough weight it will be eliminated from the proposal's sample set. These low likelihood samples are then replaced with samples that are more likely to come from the target distribution through a process known as resampling. The purpose of the resampling stage is to replace unlikely samples by more likely samples resulting in the focusing the particles into the correct state space region. The observations are used to determine the likelihood as will be seen. These prediction-correction cycles conform the proposal distribution to the target distribution. Then the expectation of the posterior density will result in the Bayesian estimate. Denote the target distribution as $p(x)$ and the proposal distribution as $\pi(x)$, then the Particle filter algorithm is as follows:

Algorithm 2 Particle Filter Algorithm

Initialization: Set prior distribution

while (state estimating) **do**

Prediction: Sample $\mathbf{x}_t^{(j)} \sim \pi(\mathbf{x}_t)$, that is draw from proposal distribution

Correction: Compute correction weights $\mathbf{w}_t^{(j)} = \frac{p(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)})}$

Resample: Draw j=1:J with probability $\mathbf{w}_t^{(j)}$

for $j = 1$ to J **do**

Add $\mathbf{x}_t^{(i)}$ to $\mathbf{x}_t$

end for

end while

Particle filters solve the filtering problem through simulation. The simulation iteratively refines the density estimate by considering the observations as indicated

in the algorithm steps above. The Particle filter's discrete density approximation is expressed as

$$p(\mathbf{x}_t|\mathbf{z}) = \sum_{i=1}^{N} \omega_t^i \delta_{\mathbf{x}_t^{(i)}} \tag{2.7}$$

where N is the number of particles and ω are the weight assignments to the corresponding particle. This density approximation, as discussed, is refined over the course of the observations with the weights continually being updated at each time step resulting in the estimation of the underlying true state density. When the Particle filter is first initialized the proposal distribution is typically selected as a uniform density, since there is no preferential knowledge of where the state is located in the state space. To emphasize how the Particle filter develops over time consider the following conceptual example of the time refinement of the proposal density. In figure 2.1 an initial prior distribution is selected as a uniform distribution over a state space ranging from -50 meters to 50 meters. This could represent the position of an object along a line. The bounded nature of the initial prior is whatever our state space constraints are. If, for example, we had a sensor that could measure range from 0 meters to 10 kilometers, then this prior would be constrained between those limits. The uniform density is used to indicate that the object could be anywhere in this state space with equal chance. Considering subsequent proposals, at latter points in time the proposal densities are extracted and displayed for this conceptual figure. A one dimensional Gaussian distribution is the representation of the state space at these times. The value of the true state is fixed at 25 meters and measurement noise Gaussian. As can be seen in the figure, as the Particle filter refines it's estimate of the true state over time, the state distribution is concentrated about the true value of the state. This conceptual figure illustrates the time evolution of the proposal density in the Particle filter using a simple one dimensional Gaussian. With each new measurement, the uncertainty in the state estimate is reduced and the belief in the

estimated state converging to the true state is increased illustrated by the reduction in the variance over time. The expectation of each density is the estimate of the state.

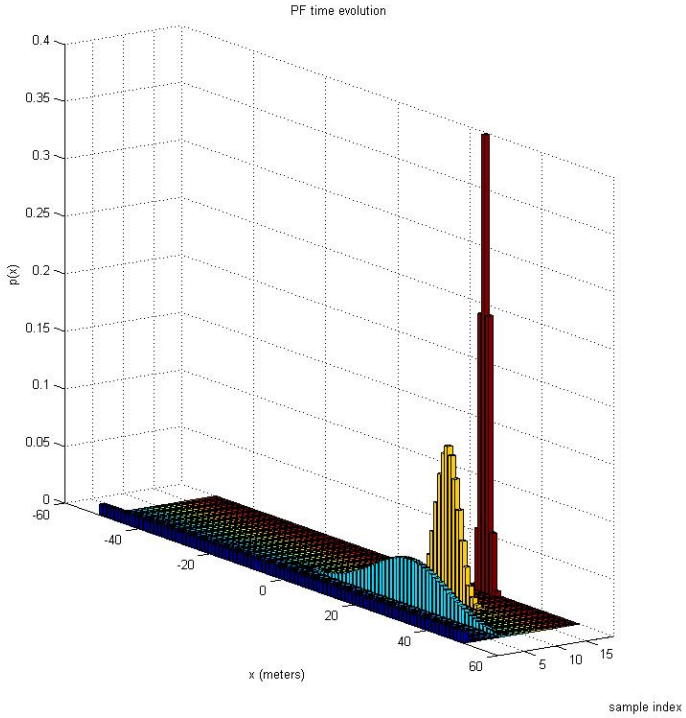

Figure 2.1: Evolution of Proposal Densities Given Gaussian Noise Densities.

Because the Particle filter uses a discrete density estimate represented by the position in the sample space and associated probability weights, it can be used to represent arbitrary densities and non-linearities in the parameter estimation. This allows usage of the Particle filter in a wider set of problems than filters that assume linearity in the model, such as the Kalman filter.

Chapter 3

GAUSSIAN MIXTURE MODEL

Clustering is an unsupervised learning technique that finds structure in a set of unlabeled data based on similarities or differences in the data. A set of criteria are used to provide ways to make these cluster associations. These criterion could be based on the Euclidean distance each data point in the data set is from a set of centroids placed in the sample space. An example algorithm that uses this approach is k-means. Other approaches may produce associations to clusters based on a probabilistic weight. The probabilistic weight could be based on the likelihood that a given data point is associated to a specific cluster. In this latter case, a data point could be associated with multiple clusters with the weight representing a measure of the strength of the association to the cluster in question. Gaussian mixture models can be used for this type of clustering.

There are two broad categories of clustering: hard and soft clustering. Hard clustering is any clustering technique where elements in a cluster belong to only that cluster. K-means is an example of a hard clustering algorithm. In this algorithm, data points are assigned to a centroid based on a minimum distance from the centroid to the data point in question. The centroid's positions are then updated based on this association and the process repeated until no new associations are made or until a threshold is reached. In contrast, soft clustering is any technique where elements can belong to multiple clusters. Soft clustering allows for data points to have multiple associations by ranking the strength of the association of the data point to all the clusters in the cluster set. Gaussian mixture models can be used to perform soft clustering and will be explored in more detail in this section.

Clustering can be broken up into a number of classifications. According to [7], these categories are exclusive, overlapping, hierarchical, and probabilistic clustering. Exclusive clustering focuses on grouping data based on assignments to only a single group and not allowing data points to have multiple group assignments. Overlapped clustering allows data points to have multiple group assignments with a score based on the strength of that membership. According to [7] hierarchical clustering is "based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted." Probabilistic clustering is completely statistical. Gaussian mixture models fall within the class of probabilistic clustering. The approach used in this paper is probabilistic clustering, specifically using the Gaussian mixture model to perform soft clustering. Gaussian mixture models are an application of the Expectation-Maximization (EM) algorithm. The EM algorithm is considered a meta algorithm that needs to be adapted to a particular application [8]. The EM algorithm is an extension of the maximum likelihood estimation procedure for estimating the parameters of a distribution [9]. The motivation for the extension is due to the mathematical tractability of either poorly behaved density functions or the feasibility of direct maximization given high dimensional density functions. The way EM deals with this is by introducing a latent random variable with the intent of simplifying the maximization of the log likelihood function. A latent variable is one that influences the data but is not directly observed [8]. In the context of EM, the latent random variable is the variable that assigns a data point to an underlying component density. This random variable is distributed according to a multinomial distribution. The EM algorithm is an iterative algorithm and is performed until convergence is achieved. The steps of the algorithm are as follows:

Algorithm 3 EM Algorithm

while (not converged) **do**

E step: Compute expectation of log-likelihood evaluated using the current estimate for the parameters, $Q(\theta|\theta^{(t)}) = E_{z|x,\theta^{(t)}}[logL(\theta; x, z)]$

M step: Compute density parameters that maximize the expected log-likelihood found using the E step. That is, $\theta^{(t+1)} = argmax_\theta < Q(\theta|\theta^{(t)}) >$

end while

where θ is the parameter under estimation, E is the expectation operation, t is the iteration step, and $L(\theta; x) = p(x|\theta) = \sum_z p(x, z|\theta)$ is the marginal likelihood of the data. The estimates computed from the M step are used to compute the next E step, and so on. Essentially a lower bound for the likelihood function is found and this lower bound is maximized by finding the partial derivatives with respect to the parameters and setting these to zero and solving for the parameter of interest. A GMM is a weighted sum of component Gaussian densities. There can be as many component Gaussian densities as needed based on the application. The GMM equation is given by

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{N} w_i g(\mathbf{x}|\mu_i, C_i) \tag{3.1}$$

where $\mathbf{x}$ is an M dimensional data vector, w_i are the mixture weights and $g(\mathbf{x}|\mu_i, C_i)$ are the component densities with a mean vector of μ and covariance C, and θ is the collection of these parameters in order to simplify the notation. It should be noted that the GMM is completely defined by the mixture weights, the mean vector, and the covariance matrix. The multivariate Gaussian density can be expressed as

$$g(\mathbf{x}|\mu_i, C_i) = \frac{1}{(2\pi)^{N/2} det^{1/2}(C)} e^{-\frac{(\mathbf{x}-\mu)^T C^{-1}(\mathbf{x}-\mu)}{2}} \tag{3.2}$$

What the EM algorithm seeks to achieve using GMM is the association weight of each data point to each component density in the density set [9]. Essentially

an association weight is given to each data point based on it's likelihood that the associated density generated that data point. The association weights are then used to adjust the parameters of the underlying component densities. The weights of the GMM reflect the strength of an underlying association to a particular component density. Samples in the data can have multiple associations with the weight showing how strong that association is. The N dimensional data $\mathbf{x}$ is partitioned by the GMM into groups called clusters where a cluster is a distribution. The EM algorithm allows us to infer the parameters of the GMM. The EM algorithm is needed for GMM because in order to find the sources of the data points the means and variances of the underlying Gaussian densities need to be known, but these parameters are unknown [9]. If the source assignments of the data points were known we could estimate the means and variances, but these assignments are not known either. The EM algorithm solves this problem by iteratively refining the assignments of data points to densities using a soft assignment process. By setting the initial Gaussian set at random centers and iteratively assigning data points to the given density and adjusting the centers and uncertainties for each iteration, data points are assigned to clusters based on how likely they are to belong to that density. The application of EM to GMM results in the following algorithm

Algorithm 4 GMM-EM Algorithm

Initialization: Randomly place clusters

while (not converged) **do**

E step: For each point compute $P(b_k|x_i)$, that is the likelihood that x_i came from the k^{th} Gaussian

M step: Adjust means and variances of densities to fit point assignments using the Maximum Likelihood Estimator for that parameter

end while

The probability $P(b_k|x_i)$ is computed from Bayes rule and is expressed as

$$P(b_k|x_i) = \frac{P(x_i|b_k)P(b_k)}{\sum_{k=1}^{N} P(x_i|b_k)P(b_k)} \tag{3.3}$$

for N component Gaussians at the i^{th} data point where i represents the data point index. The probability $P(x_i|b_k)$ is simply

$$P(x_i|b_k) = \frac{1}{\sqrt{2\pi\sigma_{b_k}^2}} e^{\frac{-(x-\mu_{b_k})^2}{2\sigma_{b_k}^2}} \tag{3.4}$$

These probabilites together compute the likelihood that a data point x_i came from the k^{th} component density. Once the soft assignments are made, then the parameters of the component densities are computed based on the data point assignments and the densities updated for the next iteration. The mean and variance of the k^{th} component density is computed using the weight generated from the Bayesian posterior $P(b_k|x_i)$. The mean can be computed as

$$\mu_{b_k} = \frac{w_1x_1 + w_2x_2 + ... + w_Nx_N}{w_1 + w_2 + ... + w_N} \tag{3.5}$$

where $w_i = P(b_k|x_i)$ is the likelihood probability that the data point x_i comes from the k^{th} component density. Likewise, the variance of the k^{th} component density is computed as

$$\sigma_{b_k}^2 = \frac{w_1(x_1 - \mu_{b_k})^2 + w_2(x_2 - \mu_{b_k})^2 + ... + w_N(x_N - \mu_{b_k})^2}{w_1 + w_2 + ... + w_N} \tag{3.6}$$

These updated parameters are then applied to their corresponding component densities for the next E step. The k-means algorithm is almost identical to the GMM/EM algorithm above but differs in that the assignments are binary in nature. The k-means algorithm simply assigns a data point to a component density without replacement. The data point is only allowed one cluster to be a member of and no other. The assignment of a data point to a cluster is based on the minimum Euclidean

distance of that data point to the cluster centroid. The centroid cluster that is closest to the data point gains the membership of that data point. The k-means algorithm is

Algorithm 5 K-means Algorithm

Initialization: Randomly place clusters

while (not converged) **do**

E step: Set $c_i = argmin_j|x_i - \mu_j|$ for a cluster centroid c_i

M step: Update the centroid based on E step, $\mu_i := \frac{\sum_{i=1}^{M} 1_{<c_i=j>} x_i}{\sum_{i=1}^{M} 1_{<c_i=j>}}$ for M centroids

end while

Both the k-means and GMM algorithms work with unlabeled data, that is, data that has no cross labels. Unsupervised learning algorithms, of which k-means and GMM belong, work with data of this form. It should be noted that because k-means uses a cost function that is not convex, it is susceptible to local minimums. This can be seen by examining the E step of the k-means algorithm and noticing that it is not a quadratic function but rather a distance function. What this implies is that since there is no global minimum for the cost function, the k-means algorithm can yield a non-optimal solution in the mean squared error sense. Figure 3.1 shows an example output of the k-means clustering algorithm. The super-imposed X's in the plot are the computed centroid centers which are the average of the data points assigned to that cluster. For this example, k-means successfully matched the data points to the correct cluster.

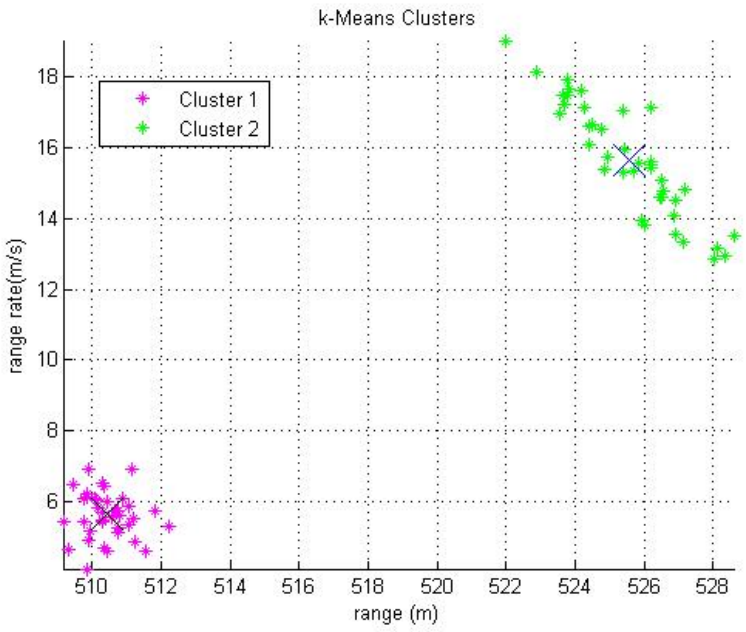

Figure 3.1: Plot of K-means Clustering.

In figure 3.2 the convergence of the algorithm is shown. In these plots the error between the actual cluster center, as measured as a magnitude from the origin, and the computed cluster center are shown.

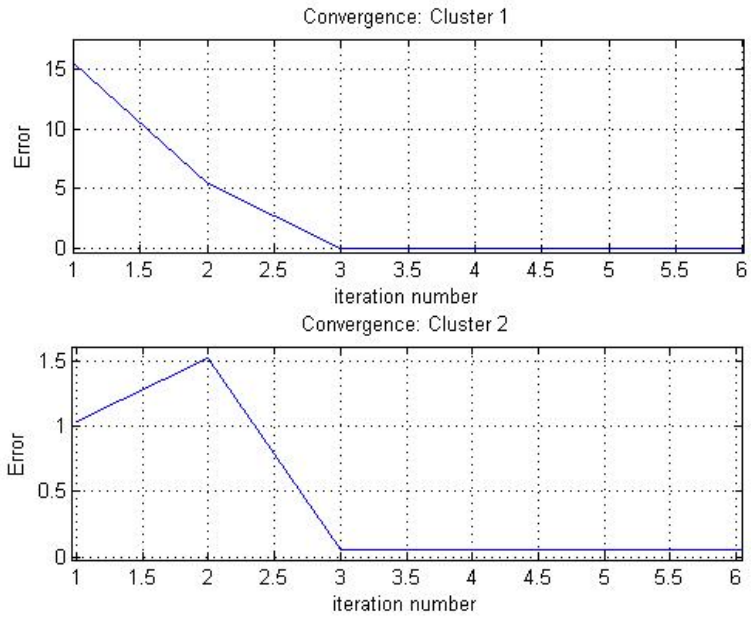

Figure 3.2: Plot of K-means Convergence to True Centroid Magnitude as Measured from Origin.

In figure 3.3 an example output of the GMM/EM algorithm is plotted. The algorithm successfully found the cluster centers.

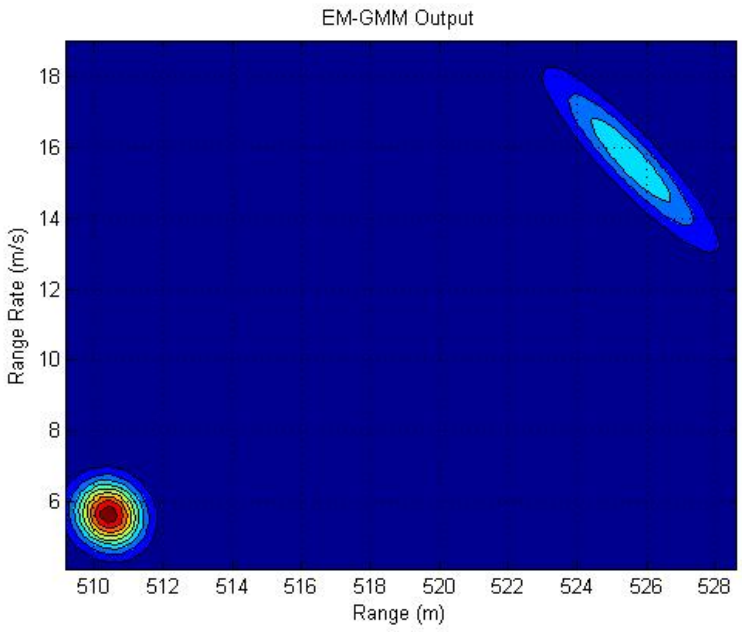

Figure 3.3: Plot of GMM/EM.

To get a baseline idea of the relative performance between K-means (KM) and the Gaussian Mixture Model (GMM-EM), consider figure 3.4.

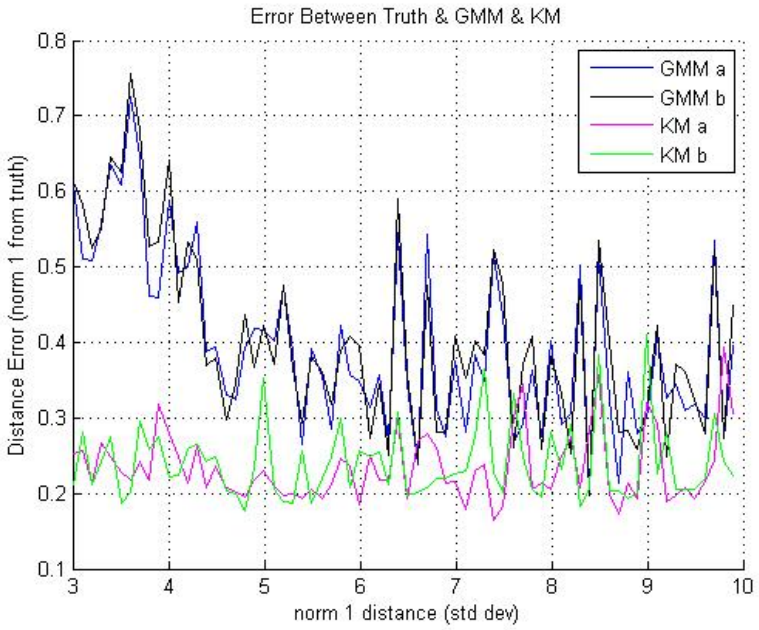

Figure 3.4: Error Between Truth and GMM vs.K-means.

This shows the performance difference between the two algorithms given two circular symmetric two-dimensional Gaussian distributions with unit variance one centered at (0,0) and the other adjusted based on a Euclidean distance of 3 sigma to 9 sigma in steps of .1 sigma from the Gaussian centered at (0,0). The error is the distance from each cluster and its corresponding truth centroid. Each generated cluster has 80 samples. The monte carlo was iterated 50 times at each distance point and the result averaged to get the sample mean of the performance. The error metric for this example is the norm 1 distance between each of the centroids and truth. As can be seen from figure 3.4, K-means on average outperforms GMM-EM considering performance as a function of centroid distance. Figure 3.5 shows that the GMM-EM algorithm is almost twice as noisy as the K-means algorithm as shown in the numbers in the legend. As discussed, K-means is a hard clustering algorithm whose cluster membership is not shared among other clusters. This has the benefit of forcing the computation of

the centroid to only include data that is closest to it thereby not allowing far off data points from affecting the estimated centroid position. GMM-EM allows all the data to be used in the computation, albeit based on a weighting inversely proportional to the distance from the centroid. This type of membership allows far off data points to have a weight on the centroid estimate and, as seen in the figures, to bias the centroid off the true value.

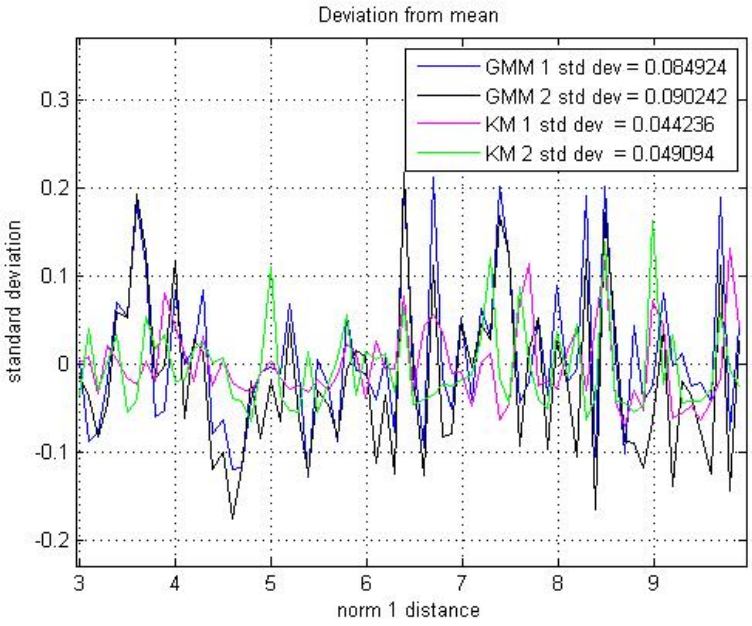

Figure 3.5: Variation of GMM-EM and KM about Mean.

Given the baseline case above it is predicted that K-means would perform better in separating the clusters.

Chapter 4

CLUTTER SUPPRESSION AND TARGET DISCRIMINATION TRACKING ALGORITHM

4.1 Introduction

In order to have a reliable target track, it is necessary to separate target detections from background noise and clutter. As clutter differs from thermal noise, it must be processed differently in order to reduce the rate of false target detections. Clutter can consist of backscatter of the transmit radar signal from land, sea or other surfaces, or it can be due to changes in the atmosphere, such as precipitation [10]. The signal-to-clutter ratio depends on many factors, including the amount of clutter illuminated, the clutter reflectivity, and the clutter that falls in the same range-angle resolution cell as the target [11]. Different approaches exist to discriminate target against surface clutter, including the moving target indication (MTI) operation mode and pulse-Doppler processing. However, if the signal-to-clutter ratio is very low, additional processing methods can be used to further increase target tracking performance. This work considers a new algorithm for separating target detections from clutter by integrating unsupervised clustering using Gaussian mixture modeling and sequential Monte Carlo methods. The block diagram in Figure 4.1 summarizes the main steps of the proposed algorithm, as described in this chapter. The range-Doppler map (RDM) measurements are processed for detection, expecting high false detections in the presence of high clutter. The clustering algorithm is then used to separate the detections into two Gaussian distinct mixtures. Post-processing of the covariance matrices of

the Gaussian mixtures is then applied using eigenanalysis to discriminate clutter and provide the particle filter tracker with clutter-suppressed detections.

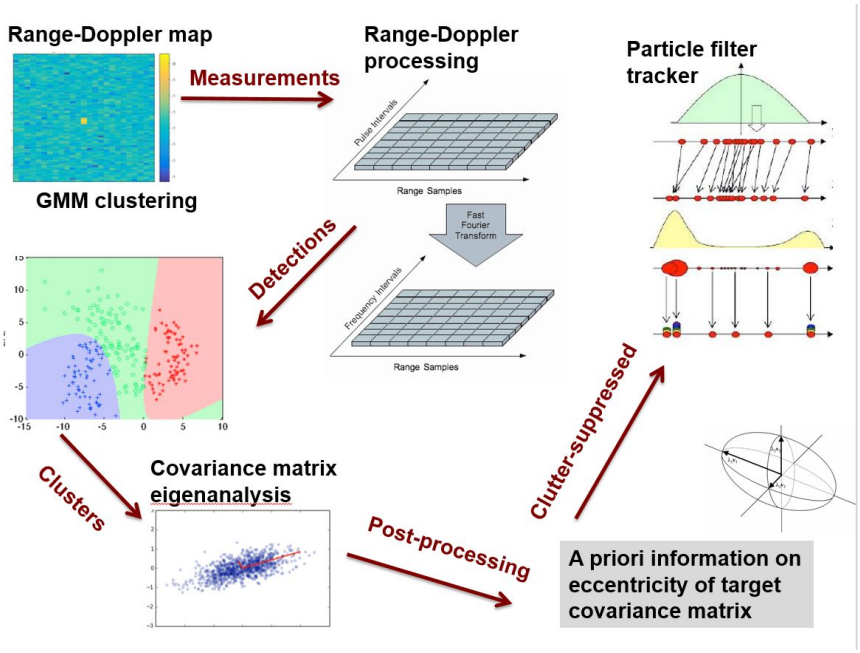

Figure 4.1: Block diagram of Proposed Algorithm for Target Discrimination and Tracking.

4.2 Pulse-Doppler Processing

We consider a pulse-Doppler radar that transmits N_p pulses over a coherent processing interval (CPI) . The baseband received signal corresponding to the mth transmitted pulse, $m_1, ..., N_p$, is given by

$$z_m(t) = \sqrt{P_r}\, s(t - \tau_0 - mT_{\text{PRI}})\, e^{-j2\pi\nu_0 mT_{\text{PRI}}} + x_{c,m}(t) + w_m(t), \quad t \in (0, T_{\text{PRI}})$$

where $s(t)$ is the transmit waveform, T_{PRI} is the pulse repetition interval (PRI), P_r is the radar return power, τ_0 and ν_0 are the time-delay and Doppler shift, respectively, that are assumed to be constant over the CPI, $x_{c,m}(t)$ is due to the presence of clutter, and $w_m(t)$ is additive white Gaussian noise (AWGN). Assuming a sampling period

T_s, the discrete-time received signal $z_m[n] = z_m(nT_s)$ is given by

$$z_m[n] = \sqrt{P_r}\, s(nT_s - \tau_0 - mT_{\rm PRI})\, e^{-j2\pi\nu_0 mT_{\rm PRI}} + x_{c,m}[n] + w_m[n], \quad n = 1, \ldots, N_s \tag{4.1}$$

where $N_s = T_{\rm PRI}/T_s$ is the largest number of samples less than $T_{\rm PRI}/T_s$. In vector form, the received signal

$$\mathbf{z}_{r,m} = [z_{r,m}[1]...z_{r,m}[N_s]]^T \tag{4.2}$$

is given by

$$\mathbf{z}_{r,m} = \sqrt{P_r}\mathbf{s}_r(\tau_0; m)e^{-j2\pi\nu_0 T_{PRI}} + \mathbf{x}_{c,m} + \mathbf{w}_m, \quad m = 1, ..., N_p \tag{4.3}$$

where T denotes vector transpose and $\mathbf{z}_{r,m} \in C^{N_s \times 1}$. The vectors $\mathbf{s}_r(\tau_0; m) \in C^{N_s \times 1}$, $\mathbf{x}_{c,m} \in C^{N_s \times 1}$, and $\mathbf{w}_m \in C^{N_s \times 1}$ are defined, respectively, as

$$\mathbf{s}_r(\tau_0; m) = [s_r(T_s - \tau_0 - T_{PRI}), s_r(2T_s - \tau_0 - T_{PRI})...s_r(N_sT_s - \tau_0 - T_{PRI}]^T \tag{4.4}$$

$$\mathbf{x}_{c,m} = [x_{c,m}[1], x_{c,m}[2], ...x_{c,m}[N_s]]^T \tag{4.5}$$

$$\mathbf{w}_m = [w_m[1], w_m[2], ...w_m[N_s]]^T \tag{4.6}$$

Considering N_p pulses over the CPI, the overall radar received signal is

$$z_{CPI}[n] = \sum_{m=1}^{N_p} z_{r,m}[n] \tag{4.7}$$

where $z_{r,m}[n]$ is given in (4.1).

In matrix form, the N_p received signals form

$$\mathbf{Z}_r = [\mathbf{z}_{r,1}, \mathbf{z}_{r,2}, ...\mathbf{z}_{r,N_p}] \tag{4.8}$$

for $\mathbf{Z}_r \in C^{N_s \times N_p}$, where $\mathbf{z}_{r,m}$ is defined in (4.3).

This matrix can also be given by

$$\mathbf{Z}_r = \sqrt{P_r}\mathbf{S}_r(\tau_0)\mathbf{D}^H(\nu_0) + \mathbf{X}_c + \mathbf{W} \tag{4.9}$$

where H denotes vector Hermitian transpose, and $\mathbf{S}_r(\tau_0) \in C^{N_s \times N_p}$, $\mathbf{X}_c \in C^{N_s \times N_p}$, $\mathbf{W} \in C^{N_s \times N_p}$ are defined as

$$\mathbf{S}_r(\tau_0) = [\mathbf{s}_r(\tau_0; 1), \mathbf{s}_r(\tau_0; 2), ...\mathbf{s}_r(\tau_0; N_p] \tag{4.10}$$

$$\mathbf{X}_c = [\mathbf{x}_{c,1}, \mathbf{x}_{c,2}, ...\mathbf{x}_{c,N_p}] \tag{4.11}$$

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_c, ...\mathbf{w}_{N_p}] \tag{4.12}$$

The Doppler term $\mathbf{D}(\nu_0) \in C^{N_p \times N_p}$ is a diagonal matrix, $\mathbf{D}(\nu_0) = \text{diag}(\mathbf{d}(\nu_0))$, with zero-valued off-diagonal entries and diagonal entries given by $\mathbf{d}(\nu_0) \in C^{1 \times N_p}$,

$$\mathbf{d}(\nu_0) = \left[e^{j2\pi\nu_0 T_{\text{PRI}}}, e^{j2\pi\nu_0 2T_{\text{PRI}}}, ...e^{j2\pi N_p \nu_0 T_{\text{PRI}}}\right] \tag{4.13}$$

Note that, if the columns, $\mathbf{z}_{\text{CPI}} = [z_{\text{CPI}}[1], z_{\text{CPI}}[2], ...z_{\text{CPI}}[N_s]]^T$, $\mathbf{z}_{\text{CPI}} \in C^{N_s \times 1}$, of the radar received matrix $\mathbf{Z}_r$ in (4.9) are stacked into a single column vector, then the vector represents the time-domain received signal over the CPI. Also, the columns of matrix $\mathbf{X}_c$ in (4.9) correspond to the communications interference symbols over each PRI. Using the above pulse-Doppler approach, range and Doppler data are collected in a receiver using the approach outlined in the next section.

4.3 Range-Doppler Map Measurements

Using the above pulse-Doppler approach, we can construct radar Doppler map (RDM) data that can be used as input to the clustering algorithm. The construction of the RDM consists of collecting range and Doppler data in a column-wise and

row-wise series known as fast time and slow time, respectively. Fast time samples are collected along the range dimension and are collected at a rate proportional to the pulse compression factor. Slow time samples are collected by taking the Fourier Transform (FT) along the range aligned rows of the range columns. This approach allows for a two dimensional representation of the scene data that can be processed by a detection algorithm. The processing at the receiver involves the correlation of the received signal at the mth PRI in (4.1) with a time-delayed version of the transmitted signal to estimate the corresponding target range. Note that slow-time processing involves the PRI time step m, whereas fast-time processing involves the time sample n. Thus, at the mth slow-time PRI time step, we compute the correlation

$$a_{\ell,m} = \sum_{n=1}^{N_s} z_m[n] s^*(nT_s - \tau_\ell - mT_{PRI}) = \mathbf{z}_{r,m}^H \mathbf{s}_r(\tau_\ell; m) \tag{4.14}$$

where τ_ℓ, $\ell = 1, ...N_\tau$, denotes the ℓth time-delay or range bin, $z_{r,m}[n]$ is given in (4.1), and $\mathbf{z}_{r,m}$ and $\mathbf{s}_r(\tau_\ell; m)$ are given in (4.3) and (4.4), respectively. The domain $[T_r, T_{\text{PRI}}]$ of τ_ℓ represents the domain of unambiguous target returns, where T_r is the duration of the transmit radar signal $s_r(t)$. This correlation, known as range correlation, is the process of taking pulse returns and applying the fast time samples to a matched filter. The reason why this dimension is referred to as fast time is because the chip rate of the signal is much larger than the pulse repetition frequency. The chip rate is the modulation rate of the bi-phase technique used in this simulation. The chip rate is selected to give a desired range resolution. Range resolution can be computed as

$$\Delta R = \frac{c\tau}{2} \tag{4.15}$$

where the value of c is the speed of light in meters per second and τ is the chip period. As can be seen when the chip period, which is the reciprocal of the chip rate, is decreased the range resolution is improved. This implies that if two or more

reflectors are separated by at least the range resolution then they will be separated at the output of the matched filter. The top plot in figure 4.1 is an example plot showing what two targets separated by the range resolution at the output of the matched filter look like. The bottom plot is another example plot that shows what two targets separated by twice the range resolution look like at the output of the matched filter. As can be seen two distinct peaks can be resolved in the bottom plot, whereas in the top plot two consecutive samples are at the same value. The output of the matched filter will have a single maximum at the time where the input to the matched filter and the matched filter's impulse response are time aligned. This information can be used to detect two distinct peaks in the top plot.

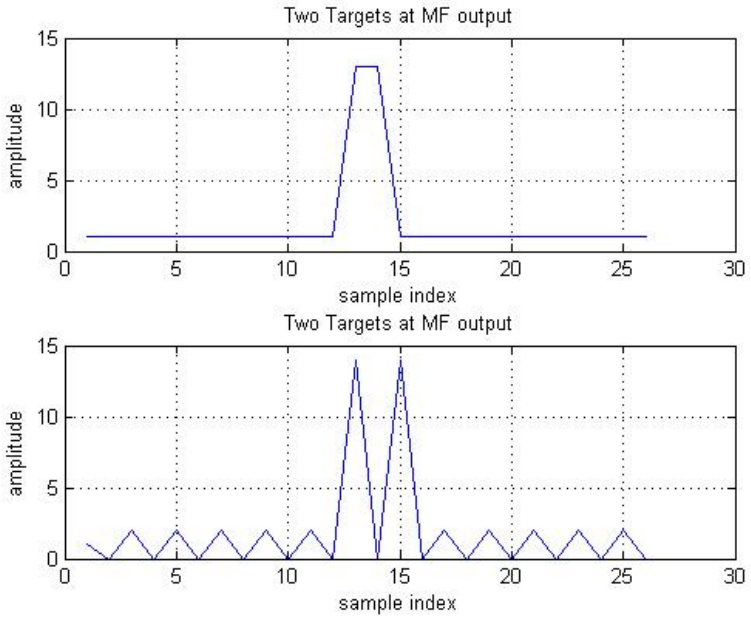

Figure 4.2: Plot of Output of Matched Filter.

In pulsed Doppler radar, a series of fast time pulses are collected at a slower rate known as the pulse repetition frequency with each pulse collection processed through a matched filter. The number of fast time samples collected is dependent on the

coherent processing interval and the number of samples that will be processed in the frequency domain by the fast Fourier transform (FFT). A coherent processing interval is a time interval over which pulses are assumed coherent and can be integrated as such using the FFT. The time interval between each pulse is the pulse repetition interval and is selected based on avoiding Doppler ambiguity. Doppler ambiguity is avoided by time spacing pulses in such a way that when the pulses are processed that the Doppler content in the underlying signals don't alias in the frequency domain. Consider the vector

$$\mathbf{x} = [Ae^{\frac{j2\pi f_d 0}{PRF}} Ae^{\frac{j2\pi f_d 1}{PRF}} ... Ae^{\frac{j2\pi f_d (N-1)}{PRF}}] \tag{4.16}$$

where A is the signal amplitude, f_d is the Doppler frequency, and N is the number of pulses in a coherent processing interval. This vector represents the slow time sampling of the fast time matched filter outputs at the matched filter peaks, that is the samples at the output of the matched filter at a given range. As can be seen, the Doppler frequency is ambiguous when f_d is greater than half of the pulse repetition frequency (PRF). The Doppler frequency is computed as

$$f_d = \frac{2vcos(\theta)}{\lambda} \tag{4.17}$$

where v is the maximum radial velocity of the target and θ is the angle between the target's velocity vector and the observer's velocity vector. For an approaching target the Doppler frequency is positive and for a receding target the value is negative. Figure 4.2 is a pictorial representation of the collection of fast time samples processed through a matched filter. In this plot four matched filter fast time samples were collected and placed in a matrix where the rows represent the matched filter samples (fast time or range) and the columns represent the coherent processing interval (slow time or Doppler). The radar signal processing will perform an FFT on each row in the matrix. The result of this will be a range doppler map (RDM).

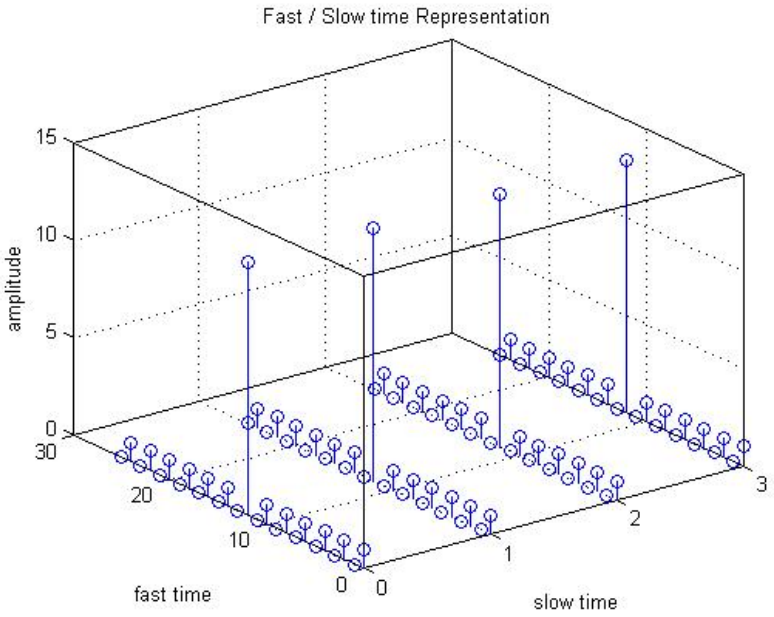

Figure 4.3: Plot of Fast and Slow Time Data Collections.

An example RDM is shown in Figure 4.3. The bright red spot is the target and the lighter blue spots are range sidelobes. The RDM is the data representation of the radar sensor output. This data representation is what the radar signal processing uses to extract range and range rate information. This range and range rate information is obtained by applying a detection algorithm to the RDM using a constant false alarm criteria. This essentially applies a threshold to each pixel in the RDM. The threshold is obtained from computing the noise power in a pixel and applying a detector structure to obtain a test statistic from the pixel data and applying this threshold. The detection strategy will be discussed in more detail in the following paragraphs.

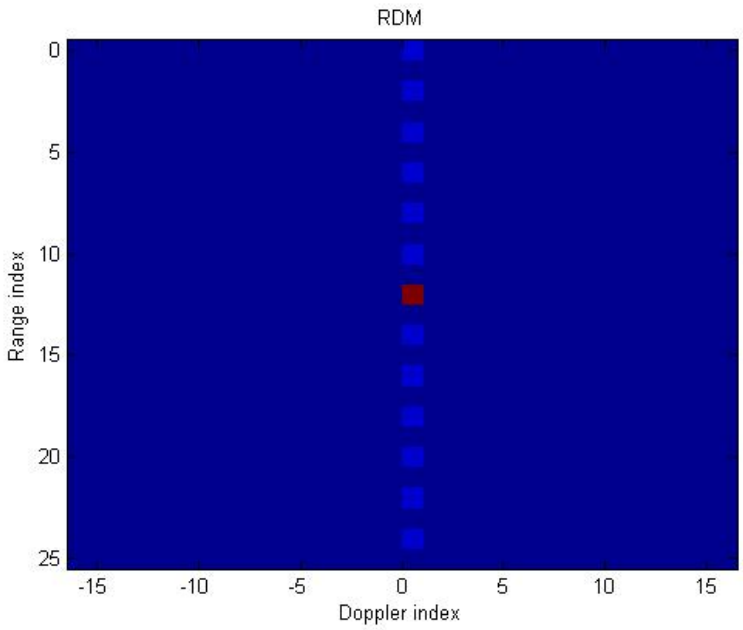

Figure 4.4: Notional Plot of RDM.

4.4 Detection Processing

A detector is used to extract range and range rate information from the RDM. The development of the detector assumes that the complex noise in the system is zero-mean circular symmetric white Gaussian noise at the input to the matched filter. Since the matched filter is a weighted linear sum of Guassian random variables then the noise at the output of the matched filter is also Gaussian, albeit with a reduction in the variance. The signal to noise ratio improvement at the output of the matched filter can be shown to be equal to the number of chips in the pulse sequence and in decibel space can be computed as

$$\mathrm{SNR}_{\mathrm{MF}} = 10log_{10}(\frac{NA^2}{\sigma^2}) \tag{4.18}$$

where N is the number of chips in the pulse sequence, using a bi-phase modulation scheme, A is the amplitude of the signal and σ^2 is the variance of the noise. The

output of the matched filter is complex and, as discussed, collected over the slow time coherent processing interval. The fixed range samples in this interval are the inputs to the FFT. The form of the samples into the FFT are of the column vector form

$$\mathbf{x} = [Ae^{\frac{j2\pi f_d 0}{PRF}} + \omega_0, Ae^{\frac{j2\pi f_d 1}{PRF}} + \omega_1, ..., Ae^{\frac{j2\pi f_d (N-1)}{PRF}} + \omega_{N-1}]^T \tag{4.19}$$

where f_d is the Doppler frequency, A is the amplitude, N is the number of pulses along the slow time, and ω_n is the complex circular symmetric Gaussian noise sample. The FFT is computed as

$$X(k) = \sum_{n=0}^{N-1} x(n)w(n)e^{\frac{-j2\pi kn}{N}} \tag{4.20}$$

where $x(n)$ are the complex time samples at a given range, $w(n)$ is a window function, N the FFT size, and k the number of bins in the FFT with a range from 0 to N-1. The magnitude of the complex vector $X(k)$ is computed prior to detection and results in two hypothesis to the detector. Since the input to the magnitude function is complex Gaussian with zero-mean for the noise and non-zero mean for the signal, the resulting probability density functions are Rayleigh and Rician, respectively. The variance of the Gaussian noise samples prior to the magnitude function are computed from the noise only RDM pixels and the Rayleigh and Rician variances are computed from this. The relationship of the Rayleigh variance to the Gaussian variance is given by

$$var(\sqrt{X^2 + Y^2}) = \frac{4 - \pi}{2}\sigma^2 \tag{4.21}$$

where X is the real component of the FFT output at a given bin and Y is the imaginary component of the FFT output at the same bin, both which are zero-mean Gaussian random variables. The parameter σ^2 is the variance of the Gaussian random variable. The relationship of the Rician variance to the Gaussian variance is given by

$$var(\sqrt{X^2+Y^2}) = 2\sigma^2 + \nu^2 - \frac{\pi\sigma^2}{2} L_{1/2}^2(\frac{-\nu^2}{2\sigma^2}) \tag{4.22}$$

where X is redefined here to be the real component of the FFT output at a given bin and Y is redefined here to be the the imaginary component of the FFT output at the same bin, both which are non-mean Gaussian random variables. The parameter σ^2 is the variance of the Gaussian random variable. The parameter ν is the center point of the distribution. The function $L_{1/2}$ is the Laguerre polynomial and is defined as

$$L_{1/2}(\alpha) = e^{\alpha/2}[(1-\alpha)I_0(-\alpha/2) - \alpha I_1(-\alpha/2)] \tag{4.23}$$

where I_0 is a modified Bessel function of the first kind, order zero, and I_1 is a modified Bessel function of the first kind, order one. Computing Bessel functions can be avoided by noticing that the natural logarithm of the Bessel function is monotonically increasing and the same detection results can be achieved by comparing the argument of the Bessel function of the Rician probability density to a modified threshold. The Rician probability density is defined as

$$p(x|\nu,\sigma) = \frac{x}{\sigma^2} e^{-\frac{x^2+\nu^2}{2\sigma^2}} I_0(x\nu/\sigma^2) \tag{4.24}$$

The Rayleigh probability density is defined as

$$p(x|\sigma) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \tag{4.25}$$

where the random variable x is greater than or equal to zero. The false alarm probability can be computed by integrating the Rayleigh probability density from the threshold T to positive infinity as

$$P_{FA} = \int_T^\infty \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx = e^{\frac{-T}{2\sigma^2}} \tag{4.26}$$

Solving for the T, the threshold, as a function of the probability of false alarm results in a threshold of

$$T = \sqrt{-2\sigma^2 ln P_{FA}} \tag{4.27}$$

By knowing the detector's sampling rate a false alarm rate can be set. The probability of detection is obtained by computing a similar integral to the P_{FA} calculation but using the Rician probability density instead. The integral can be expressed as

$$P_D = \int_T^\infty \frac{x}{\sigma^2} e^{-\frac{x^2+\nu^2}{2\sigma^2}} I_0(x\nu/\sigma^2) dx \tag{4.28}$$

According to Fundamentals of Radar Signal Processing by Mark Richards [11] this probability of detection P_D can be expressed as

$$P_D = Q_M(\sqrt{2SNR}, \sqrt{-2ln P_{FA}}) \tag{4.29}$$

where Q_M is the Marcum's Q function.

4.5 Clustering and Post-processing for Target Discrimination

After detection is performed on the RDM, the proposed algorithm is applied to those detections. The algorithm consists of several processing steps before the final target centroid is given to the particle filter for tracking. The clustering algorithm is discussed in this section. After detection is performed on the RDM, the resulting data is sent to the GMM-EM algorithm (and as a comparison, the K-means). This algorithm, as discussed, is used to separate detections into target and clutter. The determination of which cluster is the target is determined by an eigen-decomposition of the resulting covariance matrices. For this simulation, the covariance of the cluster with the minimum eccentricity is the determining factor in choosing the target. The eccentricity here is defined as the ratio of the maximum eigenvalue to the minimum eigenvalue for the covariance matrix in question.The cluster that is most correlated

to the expected eccentricity is the target of interest. The expected eccentricity can be an a priori information vector provided to the algorithm from a separate source. It is assumed in this algorithm that clutter is more elliptical than the target. A block diagram of the algorithm is provided below

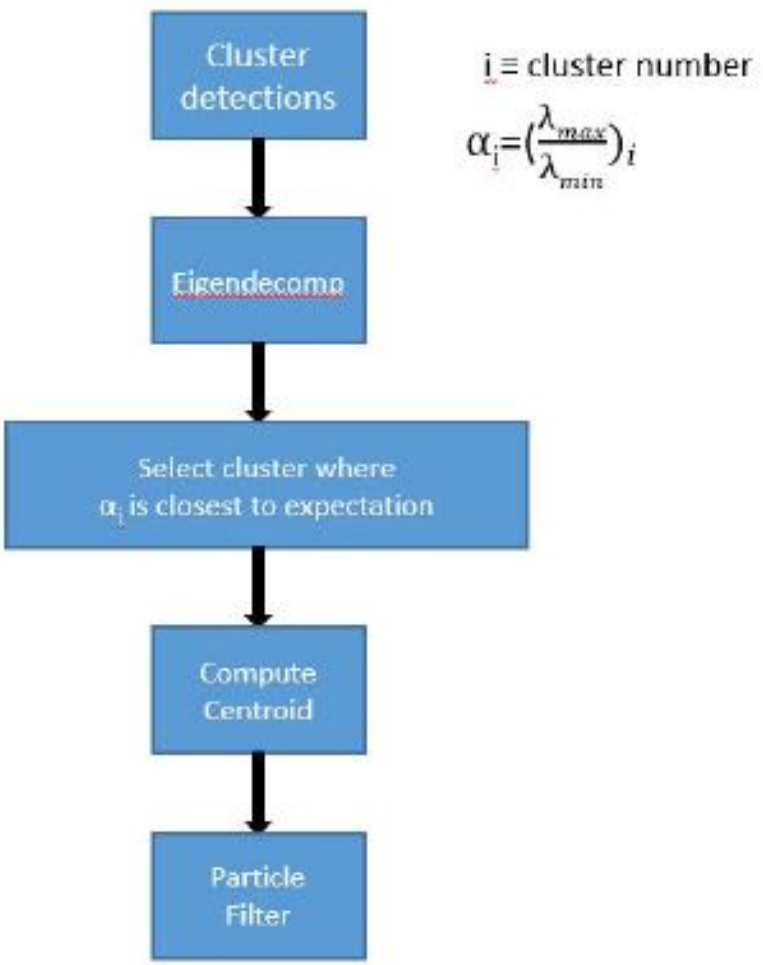

Figure 4.5: Block Diagram of Proposed Algorithm.

As discussed in chapter 3, the GMM-EM is a weighted sum of component densities. Each component density is jointly Gaussian and can model differing multi-modal

distributions, depending on the number of component densities in the sum. Since the component densities are Gaussian, each component can be fully described by its mean and covariance. In the clustering problem, detections are assigned to a component density based on a weighting assignment. This weighting assignment can be thought of as a membership strength, that is, how likely that detection is associated with the cluster in question. In this algorithm, the covariance is used to find the eccentricity of the component Gaussian. This information is used to determine which cluster is most like what is expected. Without loss of generality, consider the two dimensional covariance matrix

$$R_x = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

where ρ is a value between -1 and 1.The value of ρ determines how eccentric the covariance matrix is. Performing the eigen-analysis on the normalized covariance matrix and using the quadratic formula to find the eigenvalues results in the eigenvalues of

$$\lambda_1 = 1 + \rho \tag{4.30}$$

$$\lambda_2 = 1 - \rho \tag{4.31}$$

The eccentricity of the covariance matrix is found by taking the ratio of the largest to smallest eigenvalue for each resulting cluster. The target selection rule is

$$i = min(\frac{\lambda_{max}}{\lambda_{min}}|i = 1, \frac{\lambda_{max}}{\lambda_{min}}|i = 2) \tag{4.32}$$

where i is the cluster number. Equation 4.18 states that the lowest eccentricity is identified as the target. The centroid of that cluster, which is the sample mean of the cluster, is passed to the particle filter for tracking.

4.6 Particle Filter

As described above, the centroid with the minimum eccentricity is designated as the target cluster. The centroid of that cluster is sent to the particle filter for tracking. The particle filter is responsible for fusing measurements obtained from the clustering algorithm with the state predictions. As discussed in the second chapter, the optimal estimator that minimizes the BMSE is the mean of the posterior PDF. This can be expressed by (2.3) and is interpreted as the expectation of the state estimate given the data. The particle filter is a recursive estimator that, given a prior set of particles (realizations of a state distribution), will process this prior to generate a new set of transitioned particles. The selected target cluster's centroid measurement is passed to the particle filter which uses the measurement to correct the predicted state estimate. Weights are generated for each of the transitioned particles based upon the probability of the given measurements for each particle in the particle set. Given this new distribution, a random sampling is performed to generate a new set of estimated particles.

Chapter 5

SIMULATION RESULTS

The simulation was ran under five scenarios and the results collected. The five scenarios consisted of a fixed range model, a linear transition model, and a non-linear transition model with the Monte Carlo varying the distance between the clutter and the target for the non-linear transition model at varying eccentricities. This section will explain these scenarios in the order listed above and provide the corresponding results. The first three simulations were to produce a baseline expectation of the more strenuous scenarios of 4 and 5. Scenario 4 will explore how the algorithm performs in discrimination given separation in range and range rate between the target and clutter but with clutter eccentricity as a Monte Carlo parameter. Scenario 5 will explore how the algorithm performs in discrimination given that the target and clutter centroids are close in range and range rate with clutter eccentricity as a Monte Carlo parameter.

5.1 Scenario 1: Fixed Range and Stationary Target

Figure 5.1 shows the error between the true range and range rate and the centroids computed by the KM and GMM-EM algorithms after the eccentricity is used to associate the cluster under test to the target. This data represents a single run of the simulation. As this single data point indicates, the KM algorithm has a better average performance than the GMM-EM algorithm for this run. The spikes in the data for the GMM-EM are due to the incorrect association of the cluster under test to the target due to the eccentricity calculation. K-means also uses an eigen-decomposition of the covariance of the cluster to compute eccentricity. The GMM-EM algorithm, as discussed, uses a soft clustering technique where all data points in all clusters

contribute to the computation of the centroid. The weight of each data point in the cluster is inversely proportional to the distance of that data point from the centroid. Since the number of data points is limited, there will be some finite weight associated even with data points far from the cluster centroid. This can cause an increase in the eccentricity of both clusters depending on how the cluster samples are distributed in range and range rate space. This simulation ran centroids at 15 meters apart.

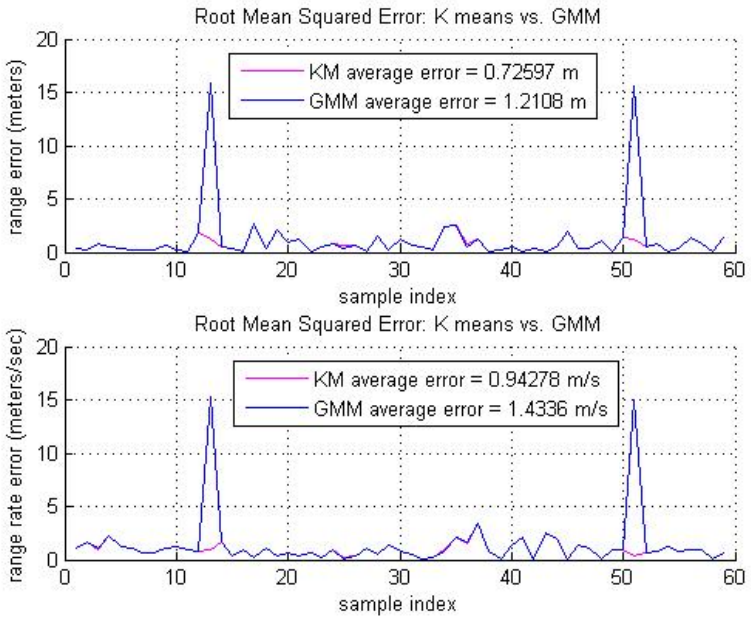

Figure 5.1: Plot of Root Mean Squared Error for KM and GMM-EM for Estimated Target Centroid.

Figure 5.2 shows the filtered centroids around the truth point. This is the target cluster and as seen in the figure the GMM-EM algorithm has cluster centroid samples that are farther from the truth point than the KM algorithm which are the result of including cluster samples from the clutter.

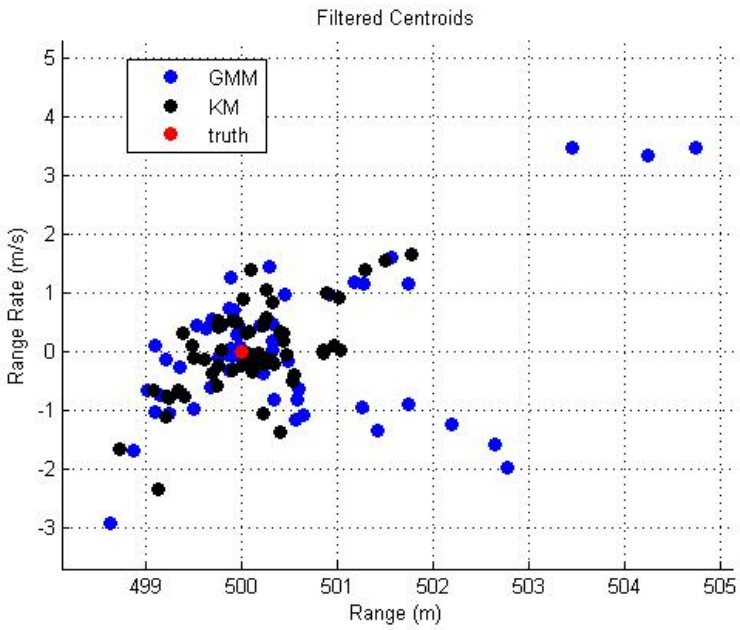

Figure 5.2: Plot of Filtered Target Centroid Estimates.

Figure 5.3 shows the error between truth and the filtered GMM-EM and KM centroids associated with the target. The Particle filter has reduced the variance of the centroid estimates and has lessened the effect of incorrect cluster association. The average error for both clustering algorithms has been reduced after the particle filter was applied. The peak range error associated with the GMM-EM algorithm was reduced from 15 meters to less than 5 meters and the peak range rate error associated with the GMM-EM algorithm was reduced from 15 meters per second to less than 4 meters per second.

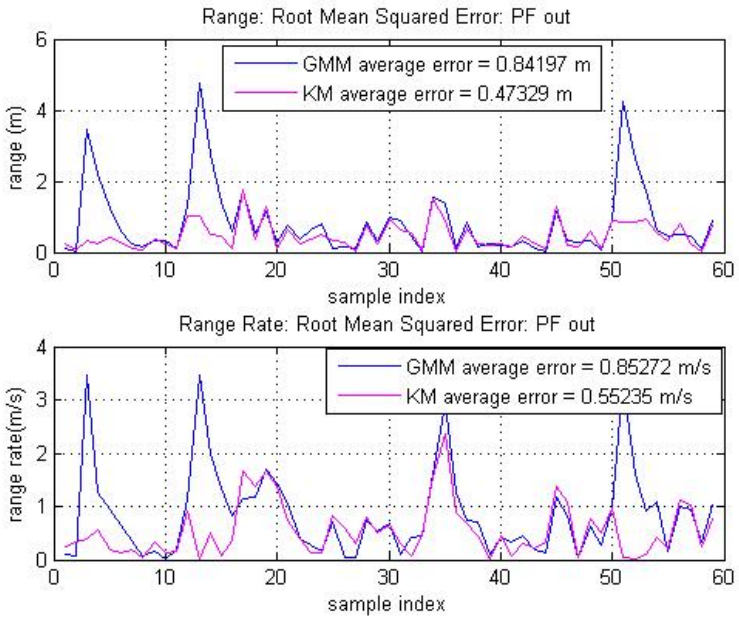

Figure 5.3: Plot of RMS Error at Output of Particle Filter for Fixed Range Scenario.

An average performance expectation was generated by running ten simulations and averaging the results with a uniform weighting. This gives a better idea of how the algorithm will perform on average. A total of ten runs were averaged together to find a sample average of the performance. Figure 5.4 shows the averaged error at the output of the GMM-EM and KM algorithms after centroid to target association is performed using the eccentricity as the discriminating feature.

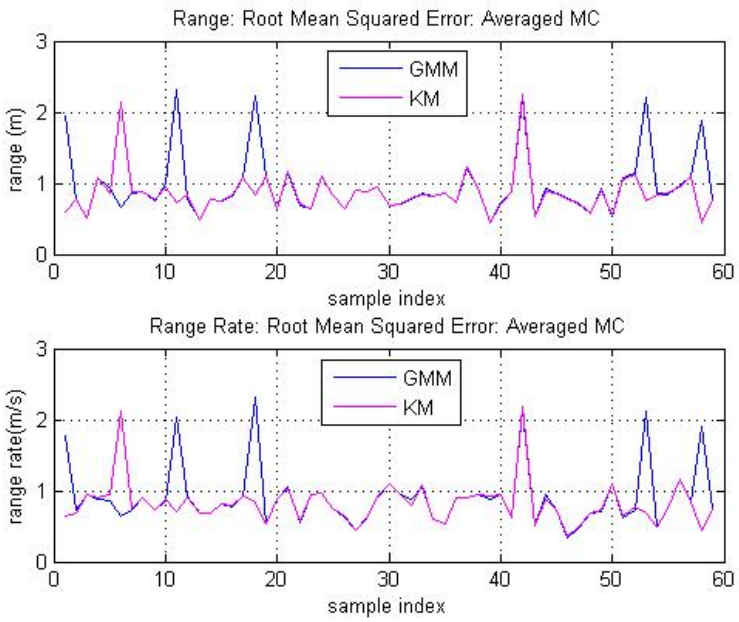

Figure 5.4: Plot of RMS Error Between Truth and the Output of the Clustering Algorithms. An Average of Ten Runs was Performed. Estimates are Target Centroids Over Tracking Time Span.

Figure 5.5 shows the averaged error at the output of the particle filter for both the GMM-EM and KM cases.

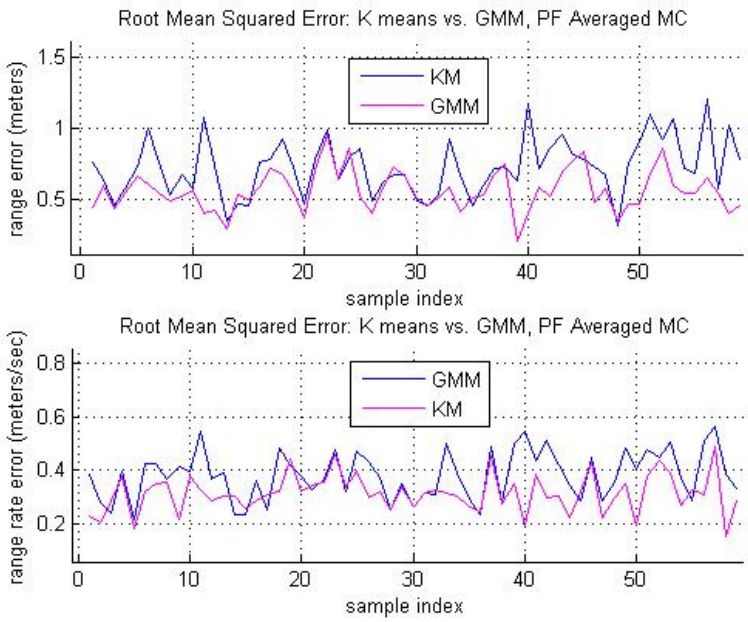

Figure 5.5: Plot of RMS Error Between Truth and Output of the Particle Filter for the Clustering Algorithms. An Average of Ten Runs was Performed. Estimates are Target Centroids Over Tracking Time Span.

Below is a table that shows the average across the Monte Carlos for both clustering algorithms for cases before and after the application of the particle filter and represents the error between where the target actually is and where the algorithm believes it is. These point estimates are computed across the time dimension of the averaged Monte Carlo. Each Monte Carlo run consisted of 60 time samples with 10 Monte Carlo iterations. Each Monte Carlo run was averaged along the time dimension and the resulting 60 samples averaged. The averaging took place with the deviation from the truth mean and so these results represent the root mean square errors relative to truth.

Table 5.1: Fixed Range

	Range Error	Range Rate Error
GMM-EM No PF	.96 m	.91 m/s
KM No PF	.86 m	.82 m/s
GMM-EM with PF	.73 m	.38 m/s
KM with PF	.57 m	.31 m/s

As the table indicates the KM clustering algorithm performs better on average than the GMM-EM clustering algorithm, but not by much. The pre-particle filter GMM-EM and KM range errors were just shy of 1 meter with 1/10 meter difference between them. The pre-particle filter GMM-EM and KM range rate errors were just shy of 1 meter also with about the same 1/10 meter difference between them as in the range error case. The biggest difference was at the output of the particle filter where the range error difference between the clustering algorithms totaled .16 meters in favor of the KM algorithm. The range rate error difference was less than 1/10 meters. The larger difference in the post particle filtered range error can be attributed to the increased variance of the GMM-EM samples.

5.2 Scenario 2: Constant Velocity Kinematic Model

Figure 5.6 shows the error between the true range and range rate and the centroids computed by the KM and GMM-EM algorithms after the eccentricity is used to associate the cluster under test to the target for the constant velocity kinematic model. The simulation Monte Carlo and resulting averages were computed the same way as in scenario 1 with the exception that the transition model is linear.

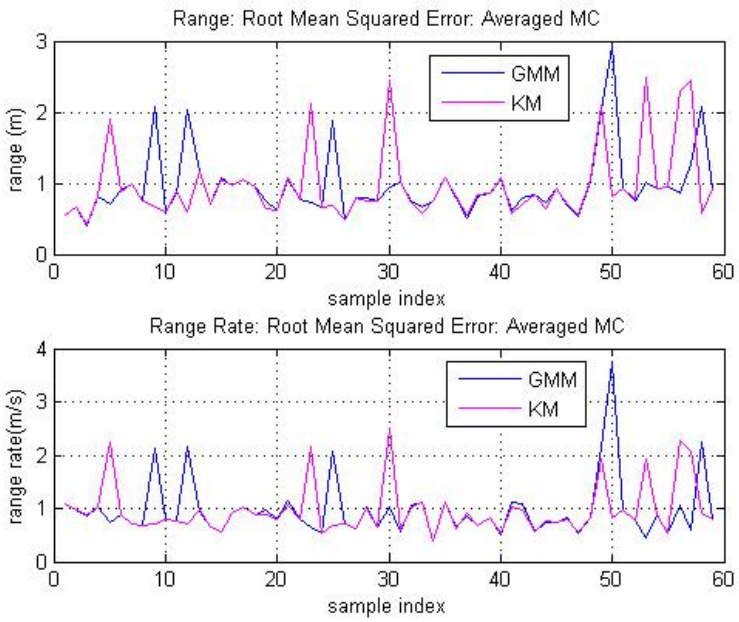

Figure 5.6: Plot of RMS Error for KM and GMM-EM for Constant Velocity Kinematic Model. Estimates are Target Centroids over Tracking Time Span.

Figure 5.7 shows the root mean squared error for the clustering algorithms at the output of the particle filter. As discussed, the Monte Carlo simulations were identically performed as in scenario one with the exception that the transition model is linear.

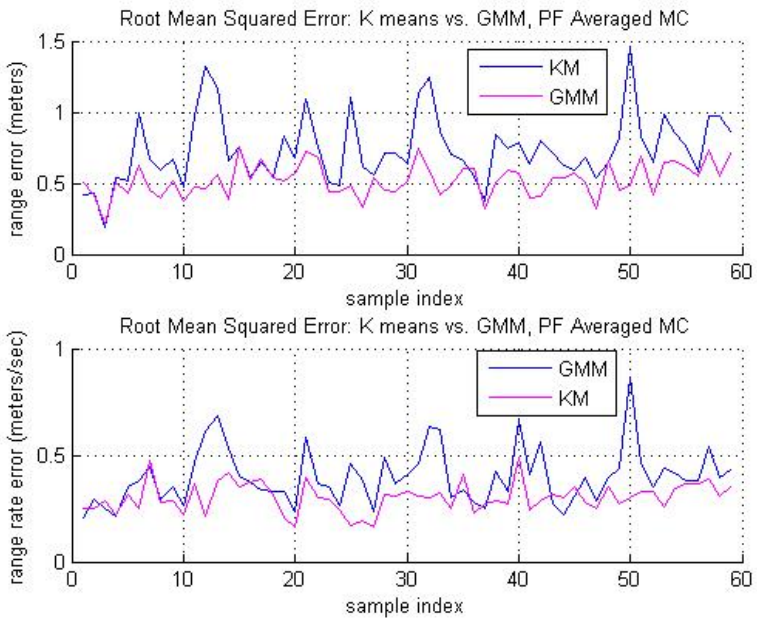

Figure 5.7: Plot of RMS Error for Filtered KM and GMM-EM for Constant Velocity Kinematic model. Estimates are Target Centroids over Tracking Time Span.

The table below is computed the same as table 5.1.

Table 5.2: Constant Velocity Kinematic Model

	Range Error	Range Rate Error
GMM-EM No PF	.96 m	.97 m/s
KM No PF	.96 m	.96 m/s
GMM-EM with PF	.73 m	.40 m/s
KM with PF	.52 m	.30 m/s

As the table indicates the KM clustering algorithm and the GMM-EM clustering algorithm before the application of the particle filter performed identically on average. The biggest difference is after the particle filter is applied. The GMM-EM range error dropped by about 2/10 of a meter while the KM range error dropped almost by half.

The GMM-EM range rate error dropped by over half and the KM range rate error dropped almost over 1/3. As a comparison to table 5.1 of scenario 1, the GMM-EM algorithm performed almost the same for the fixed range stationary target as in the linear transition model case. The KM algorithm performed better in the fixed range stationary target scenario by about 1/10 meter in error improvement before the particle filter and about the same after the application of the particle filter.

5.3 Scenario 3: Constant Acceleration Kinematic Model

Figure 5.8 shows the error between the true range and range rate and the centroids computed by the KM and GMM-EM algorithms after the eccentricity is used to associate the cluster under test to the target for the non-linear state transition model. The simulation Monte Carlo and resulting averages were computed the same way as in scenario 1 with the exception that the transition model is non-linear.

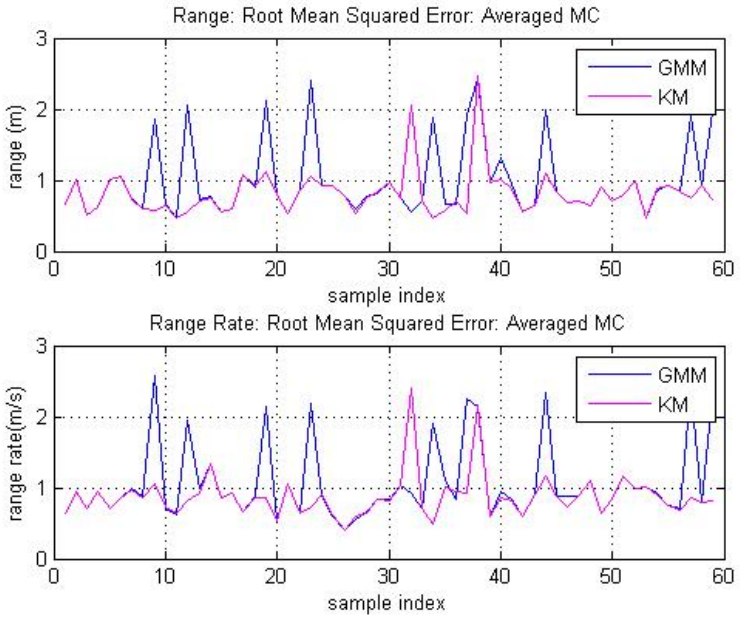

Figure 5.8: Plot of RMS Error for KM and GMM-EM for Constant Acceleration Kinematic Model. Estimates are Target Centroids over Tracking Time Span.

Figure 5.9 shows the root mean squared error for the clustering algorithms at the output of the particle filter. As discussed, the Monte Carlo simulations were identically performed as in scenario one with the exception that the transition model is non-linear.

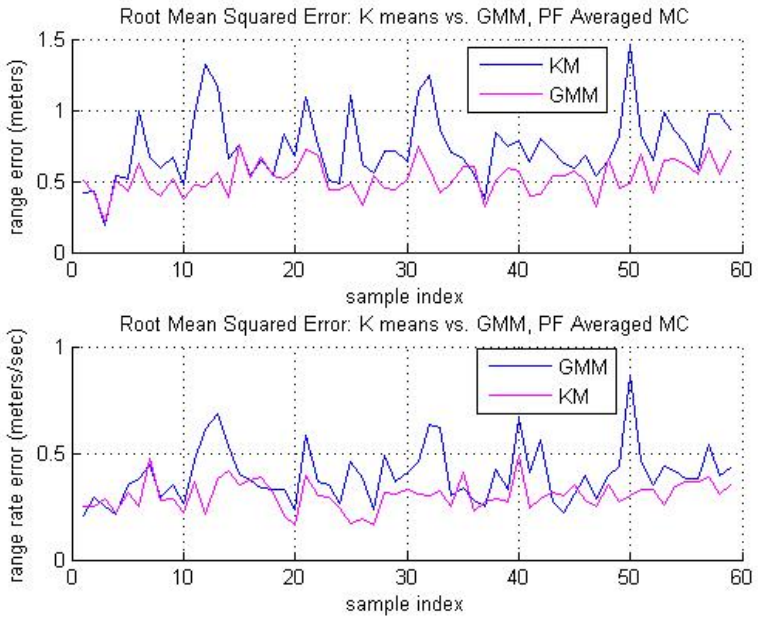

Figure 5.9: Plot of RMS Error for Filtered KM and GMM-EM for Constant Acceleration Kinematic Model. Estimates are Target Centroids over Tracking Time Span.

The table below is computed the same as table 5.1.

Table 5.3: Constant Acceleration Kinematic Model

	Range Error	Range Rate Error
GMM-EM No PF	.99 m	1.06 m/s
KM No PF	.73 m	.42 m/s
GMM-EM with PF	.53 m	.33 m/s
KM with PF	.52 m	.30 m/s

As the table indicates the KM clustering algorithm prior to applying the particle filter performs better on average than th GMM-EM. However, after the application of the particle filter the clustering algorithms performed identically on average.

5.4 Scenario 4: Constant Acceleration Kinematic Model, Function of Eccentricity

The error between the true range and range rate and the centroids computed by the KM and GMM-EM algorithms after the eccentricity calculation is used to associate the cluster under test to the target for the non-linear state transition model with varying eccentricities. The simulation Monte Carlo and resulting averages were computed the same way as in scenario 1 with the exception that the transition model is non-linear. This scenario set is intended to capture the average performance of the algorithm as a function of eccentricity at three centroid distances. There were three trials ran as a function of centroid distance and eccentricity with the centroid distance initialized at a range of 30 meters, 20 meters, and 10 meters with a corresponding range rate of 30 meters per second, 20 meters per second, and 10 meters per second. The trials are set at two different clutter eccentricities $\rho = .99$ and $\rho = .25$. The results are captured in this subsection. Results where $\rho = .99$ Table 5.4 and 5.5 below are collected at the output of the GMM-EM algorithm and the KM algorithm, respectively.

Table 5.4: Constant Acceleration Kinematic Model Range only with $\rho = .99$.

	GMM-EM Range Error	KM Range Error
30 m	1.17 m	.87 m
20 m	.73 m	.74 m
10 m	.83 m	.86 m

The table indicates that at the 30 meter range the algorithm at the output of the clustering blocks (after the target has been selected using the eigenvalues as discussed earlier) has a poorer performance than at the 20 meter range. This is because when an incorrect cluster is selected, that centroid is farther away from the centroid that should have been selected.

Table 5.5: Constant Acceleration Kinematic Model Range Rate only with $\rho = .99$.

	GMM-EM Range Rate Error	KM Range Rate Error
30 m	1.22 m/s	.91 m/s
20 m	.85 m/s	.85 m/s
10 m	.69 m/s	.73 m/s

Table 5.6 and 5.7 below are collected at the output of the Particle Filter algorithm for both clustering algorithms.

Table 5.6: Constant Acceleration Kinematic Model Range only with $\rho = .99$ at Output of PF.

	GMM-EM Range Error	KM Range Error
30 m	.71 m	.54 m
20 m	.61 m	.55 m
10 m	.54 m	.53 m

Table 5.6 and 5.7 is the result of filtering the data that generated the errors of Table 5.4 and 5.5 with the Particle Filter.

Table 5.7: Constant Acceleration Kinematic Model Range Rate only with $\rho = .99$ at Output of PF.

	GMM-EM Range Rate Error	KM Range Rate Error
30 m	.43 m/s	.34 m/s
20 m	.33 m/s	.31 m/s
10 m	.35 m/s	.34 m/s

Results where $\rho = .25$ Table 5.8 and 5.9 below are collected at the output of the GMM-EM algorithm and the KM algorithm, respectively.

Table 5.8: Constant Acceleration Kinematic Model Range only with $\rho = .25$.

	GMM-EM Range Error	KM Range Error
30 m	1.04 m	1.04 m
20 m	.77 m	.77 m
10 m	.85 m	.80 m

The table indicates that at the 30 meter range the algorithm at the output of the clustering blocks has a poorer performance than at the 20 meter range. Again, this is the result of selecting the wrong centroid.

Table 5.9: Constant Acceleration Kinematic Model Range Rate only with $\rho = .25$.

	GMM-EM Range Rate Error	KM Range Rate Error
30 m	1.01 m/s	.99 m/s
20 m	.90 m/s	.93 m/s
10 m	.99 m/s	.93 m/s

Table 5.10 and 5.11 below are collected at the output of the Particle Filter algorithm for both clustering algorithms.

Table 5.10: Constant Acceleration Kinematic Model Range only with $\rho = .25$ at Output of PF.

	GMM-EM Range Error	KM Range Error
30 m	2.43 m	1.88 m
20 m	3.00 m	3.18 m
10 m	2.40 m	2.40 m

Table 5.10 indicates that the filtering samples at the output of the Particle Filter have an additional error.

Table 5.11: Constant Acceleration Kinematic Model Range Rate only with $\rho = .25$ at Output of PF.

	GMM-EM Range Rate Error	KM Range Rate Error
30 m	.80 m/s	.74 m/s
20 m	.87 m/s	.88 m/s
10 m	.83 m/s	.82 m/s

Summary Scenario Performance Table 5.12 is a summary of the scenario averaged across the ranges and range rates above and averaged across $\rho = .99$, $\rho = .5$, $\rho = .25$, and $\rho = .125$.

Table 5.12: Constant Acceleration Kinematic Model Across Performance Envelope.

	Range Error	Range Rate Error
GMM-EM No PF	.91 m	.94 m/s
KM No PF	.83 m	.86 m/s
GMM-EM with PF	2.67 m	.70 m/s
KM with PF	2.50 m	.66 m/s

As the table indicates, the range error increased at the output of the Particle Filter.

5.5 Scenario 5: Non-Linear State Transition Model, Function of Eccentricity

This scenario is exactly the same as scenario 4 with the exception that the centroids of the clusters are much closer. For this scenario, the centroid separation is at a range of 3 meters, 2 meters, and 1 meters with a corresponding range rate of 3 meters per second, 2 meters per second, and 1 meters per second. The goal of this scenario is to see how the algorithm performs with close clusters with differing eccentricities. For this scenario only the performance envelope summary is given below in Table 5.13.

Table 5.13: Constant Acceleration Kinematic Model Across Performance Envelope.

	Range Error	Range Rate Error
GMM-EM No PF	.77 m	.80 m/s
KM No PF	.77 m	.78 m/s
GMM-EM with PF	.73 m	.38 m/s
KM with PF	.81 m	.39 m/s

As the table indicates, the range and range rate error is less than 1 meter for all cases.

Chapter 6

CONCLUSION

Estimation theory is fundamental to many branches of science and technology. This paper used estimation theory to propose an algorithm for target discrimination and tracking in the presence of clutter. The GMM-EM and K-means algorithms are estimation techniques used to associate data with groups. The GMM-EM algorithm and the K-means algorithm were both used to provide the mechanism to separate a target from clutter, a process known as clustering. Once the clusters were formed, the eigenvalues of the resulting covariance matrices of the respective clusters were used to discriminate the target from the clutter. This information was an a priori piece of information provided to the algorithm and would represent some target feature, in the case of this paper, an extent in range and range rate space. This feature needs to be accentuated by the clusters so the algorithm will not confuse the clutter with the target. This can occur when the eccentricities, defined in this paper as the ratio of the maximum eigenvalue to the minimum eigenvalue, of the clusters are close enough where errors in the covariance exceed what this ratio would be. The result of this is that the incorrect cluster and it's centroid is selected for tracking. In this algorithm the Particle Filter was used to track the target state. The Particle Filter is another estimation algorithm used in a variety of applications. In state estimation, as used in this algorithm, the Particle Filter tracks the time evolution of a target's state. Future work will consist of augmentations of the algorithm to include using the angle of the target cluster as an additional discriminating feature. This can be useful when the eccentricities are close with the result of selecting the wrong cluster to track.

REFERENCES

[1] S. M. Kay, *Fundamentals of Statistical Processing: Estimation Theory.* Prentice Hall, N.J., 1993.

[2] L. P. Espindle and M. J. Kochenderfer, "Classification of primary radar tracks using Gaussian mixture models," *ET Radar, Sonar and Navigation*, vol. 3, pp. 559–568, 2009.

[3] K. Panta, D. E. Clark, and B.-N. Vo, "Data association and track management for the Gaussian mixture probability hypothesis density filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, pp. 1003–1016, 2009.

[4] J. Zhang and Y. Liu, "Single maneuvering target tracking in clutter based on multiple model algorithm with Gaussian mixture reduction," *Journal of Applied Research and Technology*, pp. 641–652, 2013.

[5] Y. F. Shi, T. L. Song, and D. Mušicki, "Target tracking in clutter using a high pulse repetition frequency radar," *IET Radar, Sonar & Navigation*, vol. 9, pp. 299–307, 2015.

[6] P. Abbeel, "Bayes filters." [Online]. Available: http://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

[7] M. Matteucci, "A tutorial on clustering algorithms." [Online]. Available: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/index.html

[8] E. Weinstein, "Expectation maximization algorithm and applications." [Online]. Available: http://cs.nyu.edu/~eugenew/publications/em-talk.pdf

[9] A. Ng, "Mixtures of Gaussians and the EM algorithm." [Online]. Available: http://cs229.stanford.edu/notes/cs229-notes7b.pdf

[10] D. K. Barton, *Radar Equations for Modern Radar.* Artech House Publishers, 2012.

[11] M. Richards, *Fundamentals of Radar Signal Processing.* McGraw-Hill Professional Engineering, 2014.