A Semi-Markov Multiple Event Filter for Maneuvering Targets

Peter Abeles
Advanced Technology Center
Lockheed Martin Space Systems Company
Sunnyvale, CA, U.S.A.
Email: peter.j.abeles@lmco.com

Mike Kovacich
Advanced Technology Center
Lockheed Martin Space Systems Company
Sunnyvale, CA, U.S.A.
Email: mike.kovacich@lmco.com

Abstract— Tracking maneuvering targets is a difficult problem due to unpredictable maneuvers which change the target’s state and/or dynamics. To ensure track accuracy a filter needs to model the target correctly and quickly respond to maneuvers. A new sequential filter is proposed which attempts to improve upon existing algorithms in several areas. A more flexible internal model is used to describe effects of maneuver events. Maneuver hypotheses have improved temporal accuracy. A semi-Markov process is used to describe the probability of an event occurring as a function of time. In simulated test scenarios the new algorithm performs as well as or significantly better than Interacting Multiple Model filter.

Keywords: Filtering, Markov, semi-Markov, estimation, multiple-model, maneuvering, MTT

I. INTRODUCTION

Maneuvering target tracking (MTT) is an important problem, which has been the subject of intensive research in missile defense, air traffic control, surveillance, robotics, and other fields. A target can perform a controlled maneuver for a variety of reasons; such as to avoid detection, evade an interceptor or to reach its destination.

Filters are used to estimate a target’s state and predict where it might be in the future. Observations are assigned to the target’s track using these predictions, which the filter then uses to update its state. It is not uncommon for a tracking system to track hundreds of targets with numerous false alarms. Filters capable of generating accurate state and covariance estimates are essential for creating an effective tracking system.

To reduce the peak computational and memory requirements sequential filters are commonly used. Perhaps the most popular sequential filter is the Kalman filter, which is often sufficient for non-maneuvering targets. However because of the additional unknowns in MTT other approaches are needed to achieve a high level of accuracy. A popular strategy is multiple-model (MM) filtering [1]–[3].

The MM approach uses a Bayesian framework. The target is modeled as a system that obeys one of a finite number of models [2]. In other words the system is constrained to be in a single mode \(M(k) \in \{m_1, \ldots, m_n\}\) at any time and there is a known probability \(Pr(m_j|m_i, t_k, t_{k+1})\) of transitioning from mode \(m_i\) to \(m_j\) between time \(t_k\) and \(t_{k+1}\). In this framework a maneuver is defined as change in the target’s mode. A mode is a set of dynamics equations which describes the motion of the target. While the MM approach has been applied successfully to a wide variety of problems there are several areas for improvement.

An example of a maneuver that is not accurately modeled by the MM approach is the angular velocity of a turning airplane. A common approach is to limit the angular velocity to a discrete set of values. This works well when the target’s true angular velocity is one of the hypothesized values, but its performance is degraded if it is not. While increasing the number of models is an obvious solution, it can in fact degrade the performance both in terms of accuracy and runtime [4]. An alternative approach is to model the maneuver as an instantaneous event after which the velocity is initially unknown. This allows the filter to converge towards an arbitrary value.

Multi-sensor tracking of maneuvering targets poses additional challenges. The observation rate and type of observation can both vary with time. For example, a moving target might become detectable or undetectable to a sensor, each sensor can have a different accuracy/frequency, and a sensor’s tasking can cause it to have a variable update rate. For a filter to function in such an environment it must correctly model the target’s maneuverability for a wide range of update rates.

The theoretical upper limit for the accuracy of a MM filter is set by the accuracy of its individual models, that are Kalman filters. A Kalman filter is an optimal filter in the least-squares sense only when it has an accurate model of the system. If an event occurs and the change is not reflected in any of the models, then all the internal filters will be suboptimal. It is a common practice to only hypothesize events when a new observation arrives. This couples the individual filter’s accuracy to the sensor’s observational frequency. For systems/sensors with a relatively slow observation rate this can significantly impact the track accuracy. To improve the filter’s upper limit for accuracy the filter needs to be able to hypothesize events between observations.

In MM filters the probability for when an event occurs as a function of time is most often described using a Markov process. Given the present state the future and past states are independent in a Markov process. This is known as the Markov property and is of practical importance since it enables filters to be “memoryless”. While simple and elegant it is a poor representation of the sojourn time distribution for many real
world problems of interest.

The sojourn time is the elapsed time between two consecutive events. For a Markov process, the effective sojourn time is an exponential distribution that decays with time. As a result filters will tend to heavily weigh events that occur early on, see Figure 1. However, as is discussed in [5] events are more likely to occur later for the MTT problem.

The semi-Markov (SM) process [6], or Markov renewal process, represents the sojourn time as a stochastic variable and has a fixed matrix of event probabilities. The future is dependent upon the current state and current sojourn time. This process is capable of describing a wider range of systems than the Markov process.

Several different approaches have been taken in the past to implement a filter using a SM process. Some of the earlier attempts to use a SM process are presented in [7]–[9]; however because the sojourn time is set to the observation period, which is constant, a Markov process is actually implemented. An approach is presented in [10], [11] that merges hypotheses in a manner similar to the Interacting Multiple Model (IMM) algorithm [12]. A second order shaping filter was derived from a SM process with a Gamma distribution in [7]. A more detailed summary of these and other algorithms is presented in [1], [13], [14].

Hypothesis maintenance is an important issue in MM filters since it allows them to run in real time. There are three different maintenance strategies: merging, pruning, and a hybrid of both. Pruning is the easiest strategy to implement in code, but merging tends to yield better results [1]. All strategies converge towards the same level of accuracy as the maximum number of hypotheses increases. When merging hypotheses that use a SM process, a complexity arises. Hypotheses need to have similar sojourn times; which reduces the number of hypotheses that can be merged and increases the complexity.

A new sequential filter is proposed that addresses the issues previously raised. It breaks from the traditional MM formalism by allowing an event to cause a discontinuous change in the system’s state and change the system’s mode. Events are hypothesized at a predefined frequency independent of the sensor’s observation frequency. The target’s sojourn time is modeled as a SM process. The computational complexity is controlled by saving only the n-best hypotheses for the system. This algorithm is referred to as the SMME-n filter, which stands for “semi-Markov Multiple Event n-Best” filter.

In section II the problem is formally defined. In section III the SMME-n algorithm is described. Section IV describes the simulation environment and configuration of the filters. Section V contains simulation results and explanation.

---

1 Traditionally a SM process describes the probability of transition from one mode to another mode. Here it describes probability of an event given the mode.

2 A merging/pruning hybrid method was originally developed for SMME. It’s not discussed in this paper because of its complexity and the small performance gain.

---

**Fig. 1.** Comparison of the sojourn probability density function (PDF) from an exponentially decaying (Markov) and pseudo-Gaussian distribution.

**II. PROBLEM FORMULATION**

The state of the system evolves from time $t_k$ to time $t_{k+1}$ according to:

$$x(k+1) = \Phi[M(k), k]x(k) + \Lambda[M(k), k] + w[M(k), k]$$

(1)

where $x \in \mathbb{R}^r$ is the system state vector, $M(k) \in \mathbb{M}$ is the mode, $r$ is the state vector dimension is mode dependent, and the plant noise $w(k)$ is a zero-mean white Gaussian process. The matrices $\Phi \in \mathbb{R}^{r \times r}$ and $\Lambda \in \mathbb{R}^{r \times r}$ are assumed to be known. Sensor measurements are described by:

$$z(k) = H[M(k), k]x(k) + v(k)$$

(2)

where $z \in \mathbb{R}^l$ is the measurement vector, $H \in \mathbb{R}^{l \times r}$ is assumed to be known and $v(k)$ is a zero-mean white Gaussian process.

Let $E \triangleq \{e_1, \ldots, e_b\}$ be a set of events which represents the effects of maneuvers. An event $e_i \triangleq \{G_i, m_i\}$ is defined by a state transformation function $G_i$ and a mode $m_i \in M$. When the event occurs the mode of the system will be set to $m_i$, and the system state is transformed by function $G_i$ as is shown below:

$$x^+(t) = G_i[x^-(t)]$$

(3)

where $x^-$ and $x^+$ is the state of the system immediately before and after the event respectively. The dimension of the state vector can be changed by the transformation function.

The temporal distribution of events and their type is controlled by a SM process. Each mode $m_i \in M$ has a sojourn time distribution $p(e \in E | \tau, m_i)$ that specifies the likelihood of any event occurring $\tau$ seconds after the previous event. Given that an event has occurred and the current mode is $m_i$, the probability that the next event will be $e_j$ is specified using an event probability matrix below:

$$\pi_{ij} \triangleq Pr(e = e_j | m_i) \ \forall m_i, e_j$$

(4)

All noise processes are mutually independent with each other and in time.

3 A more common implementation of the SM process found in the literature would have a sojourn distribution for each model event pair. By having only having a single sojourn distribution for all events for a given mode the filter’s computational and memory requirements are significantly reduced.
III. ALGORITHM DESCRIPTION

![Flow chart of the update cycle.](image)

After the SMME-n algorithm has been initialized it recursively updates its state as new measurements are received. A functional diagram of the SMME-n algorithm is shown in Figure 2. The filter can have at most a total of \( N \) hypotheses at any given time.

A hypothesis \( H \) is the result of an initial mode \( m_0 \), and a sequence of events. The hypothesis’ likelihood is calculated using its initial probability, event history and sequence of observation residuals. The sequential state is defined as all the variables needed to fully describe a hypothesis at time \( t \geq t_k \) given measurements up to \( k \). In SMME-n, a sequential state is defined using six values, shown below:

\[
H = \{ \gamma(k), \tilde{\zeta}, t_e, m, x(t), P(t) \}
\]

(5)

where \( \gamma(k) \) is the cumulative measurement residual likelihood, \( \tilde{\zeta} \) is the cumulative event probability, \( t_e \) is the time of the previous event, \( m \) is the current mode, \( x(t) \) is the estimated state, and \( P(t) \) is the estimated covariance matrix. A super script above the preceding variables is used to indicate a specific hypothesis, for example \( \gamma^i(k) \).

The computational complexity of the SMME-n algorithm is approximately \( O(NM) \), where \( N \) is the max number of hypotheses and \( M \) is the average number of times that hypotheses are spawned in a single propagation step.

See Appendix D for a derivation of SMME-n.

A. Initialization

The filter is initialized using any prior information available. At a minimum there needs to be an initial state and covariance provided for each possible initial mode. Hypotheses should be created that cover the spectrum of possible sojourn times.

Each hypothesis’ sequential state is initialized with: \( \gamma(0) = 1 \), \( \tilde{\zeta}(0) = Pr(m, \tau_0) \), \( x(0) = \hat{x}_0 \), \( P(0) = P_0 \), \( m = m_0 \) and \( \tau_0 > 0 \) is the assumed initial elapsed time.

B. Update Cycle

When a measurement arrives at time \( t_k \), the hypotheses are updated in four steps: propagate, spawn, prune, and update. New hypotheses are created in the spawn step every \( T_e \) seconds (the event hypothesis period). The time of the next scheduled event hypothesis is symbolized as \( t_{sp} \).

1) Propagate: If the next scheduled event time is less than \( t_k \) then the hypotheses are propagated to \( t_{sp} \), otherwise they are propagated to time \( t_k \). The Kalman filter’s prediction equations are used to propagate the state and covariance associated with a hypothesis forward in time.

2) Spawn: When a hypothesis is propagated to time \( t_{sp} \), events are hypothesized, spawning new hypotheses. For each hypothesis \( i \) in \( \{1, \cdots, N\} \) and event \( e_r \) in \( E \) pair, a new hypothesis is spawned by transforming the sequential state of hypothesis \( H^i \) according to event \( e_r \). The transformation, resulting in the new hypothesis \( H^j \), is specified below for each variable in the sequential state:

\[
\gamma^j(k) = \gamma^i(k)
\]

\[
\tilde{\zeta}^j = \pi_{r}Pr(e \in E|t_{ab})\tilde{\zeta}^i
\]

\[
t_e^j = t_{sp}
\]

\[
m^j = m_r
\]

\[
[x^j(t), P^j(t)] = G_{k}[x^i(t), P^i(t), T_e]
\]

where \( H^j \) is in mode \( l, t_{ab} = \{ t_a | t_a < \tau < t_b \} \) is a set of hypothesized sojourn times, \( t_a = t_{sp} - T_e/2 \), and \( t_b = t_{sp} + T_e/2 \). Notice that the transformation function \( G_{k} \) has \( T_e \) as an input. This allows temporal uncertainty to be taken in account, see Appendix A.

3) Prune: After new hypotheses are spawned the most likely according to equation 8 are saved, the rest are pruned. To avoid issues caused by the finite precision of floating point numbers \( \tilde{\zeta} \) is normalized for all hypotheses at this point.

4) Update: After all the hypotheses have been propagated to \( t_k \) a Kalman filter is used to update each hypothesis’ state and covariance. For each hypothesis \( i \) in \( \{1, \cdots, N\} \), the measurement residual (or innovation) is calculated and used to update the residual sequence likelihood using the equations below:

\[
\gamma^i(k) \triangleq \frac{\gamma^i(k-1)p[z(k)|H^i]}{\sum_{j} \gamma^j(k-1)p[z(k)|H^j]}
\]

(6)

\[
p[z(k)|H^i] = N[v^i(k); 0, S^i(k)]
\]

(7)

where \( v \) and \( S \) are the measurement residual and residual covariance for each hypothesis, see equations 17 and 18.
C. Output

The probability for each hypothesis \( i \in \{1, \ldots, N\} \) at time \( t \geq t_k \) given \( k \) measurements is calculated using:

\[
\mu^i(t|k) \triangleq Pr(H^i|Z^k, t) \approx \frac{\gamma^i(k)\zeta^i(t)}{\sum_j \gamma^j(k)\zeta^j(t)} \quad (8)
\]

\[
\zeta^i(t) \triangleq \sum_j \tilde{c}^i \Pr(e \in E|\tau^j > \tau^i_t) \quad (9)
\]

where \( \mu \) is the hypothesis probability, \( \zeta \) is the past and future cumulative event probability, \( \tau^i_t = t - t_e \) is the elapsed time since the previous event, and \( \tau^j \) is the sojourn time.

Using the law of total probability the output state is calculated as a mixture of Gaussians:

\[
x(t|k) = \sum_{i=1}^{N} \mu^i(t|k)x^i(t|k) \quad (10)
\]

\[
P(t|k) = \sum_{i=1}^{N} \mu^i(t|k)\{P^i(t|k) + [x(t|k) - x^i(t|k)][\cdots]T\} \quad (11)
\]

where \( x(t|k) \) and \( P(t|k) \) is the predicted state and covariance at time \( t \geq t_k \) given the measurements up to \( k \).

IV. EXPERIMENTAL SETUP

The problem of tracking boost phase missiles initially inspired the development of the SMME-n filter. However, boost phase tracking is excessively complex for demonstration purposes, therefore an alternative scenario is used. While not physically possible this simplified scenario is designed to stress the filters in the same way as the original problem, without complicated non-linear dynamics.

A system is considered that can operate in one of two modes at any given time, moving (mode 1) and stationary (mode 2).

\[
x(k + 1) = \begin{cases} 
\Phi x(k) + \Lambda + w_1(k) & M(k) = 1 \\
\Phi x(k) + w_2(k) & M(k) = 2 
\end{cases} \quad (12)
\]

While in the moving mode the state \( x(k) = [p_k, v_k]^T \) is a position and velocity column vector and for the stationary mode \( x(k) = [q_k] \) is position scalar. Both \( w_1(k) \sim N\{0, Q_1\} \) and \( w_2(k) \sim N\{0, Q_2\} \) are white Gaussian noise processes.

\[
\Phi = \begin{bmatrix} 1 & T_p \\ 0 & 1 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \frac{1}{2}T_p^2 \\ \frac{1}{2}T_p \end{bmatrix} a_c \\
Q_1 = \begin{bmatrix} T_p^3/3 \\ T_p^2/2 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} T_p \end{bmatrix} q_2
\]

where \( q_1 \) and \( q_2 \) are scalars, \( T_p \) is the sensor’s observation period, and \( a_c \) is a scalar constant control acceleration.

There are two possible maneuver events: stationary, and velocity jump. While in the moving mode it can perform both the stationary and velocity jump maneuvers. While in the stationary mode it can only perform the velocity jump maneuver. The probability of either event occurring while in the moving mode is 50% and while in the stationary mode the probability of the velocity jump occurring is 100%.

The velocity jump maneuver changes the velocity to a value selected from a uniform distribution between \( 20\frac{m}{s} \) and \( 40\frac{m}{s} \) and sets the mode to moving. The stationary maneuver changes the mode to stationary, effectively setting the velocity to zero. The transformation functions associated with these events are given in equations 13 and 14 for the stationary and velocity jump events respectively.

\[
G[x(t)|1] = \begin{bmatrix} p \end{bmatrix} \quad (13)
\]

\[
G[x(t)|2] = \begin{bmatrix} p \end{bmatrix} \begin{bmatrix} U(20, 40) \end{bmatrix}^T \quad (14)
\]

Sensor position measurements are corrupted by additive noise:

\[
z(k) = Hx(k) + v(k) \quad (15)
\]

where \( v(k) \sim N\{0, \sigma_v^2\} \) and \( H = [1, 0] \) or \( [1] \) depending upon the mode.

All noise processes are independent and uncorrelated with each other and with time. Unless stated otherwise the following simulation parameters were used; \( \sigma_v^2 = 144, T_p = 0.5s, q_1 = 1, q_2 = 4, \) and \( a_c = 5 \frac{m}{s^2} \).

A. IMM Configuration

To provide a point of reference for relative performance the IMM filter [12] is used. While other MM filters, such as the MM \( n \)-Best [1], are closer to the computational complexity of the SMME-n, the IMM was selected for two reasons. It is one of the most popular MM filter, and in a study [15], none of the other filters were better in all situations.

The IMM filter is a suboptimal filter that is an approximation of the Generalized Pseudo-Bayesian 2 (GPB2) filter [2]. The filter is composed of \( N \) internal modes (or models). A Markov chain describes how the target maneuvers by specifying the probability of mode switches. During the interaction step, hypotheses are merged with each other using the conditional probability of switching modes. After merging, the model weights and state estimates are updated using the measurement’s residual.

While the IMM does not have the capability to model the system being considered perfectly, it can approximate it. Such approximations are often done. For example, an IMM was proposed in [16] for tracking a boosting missile. Boosting missiles are difficult to model accurately in an IMM, because the acceleration changes with time and can undergo abrupt changes during a staging event. The following IMM filter is similar in spirit to the approach taken in [16].

An IMM is proposed that is composed of two models that are specified in equation 12; moving, and stationary. The Markov chain is defined using a transition probability matrix, specified below:

\[
\pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}
\]

The moving mode’s plant noise was inflated from truth to \( q_1 = 100 \) to account for the velocity jump event while it is in the moving mode. The transition probability matrix and plant noise magnitudes were determined empirically by trading off consistency and accuracy across the whole trajectory.
During model interaction when a stationary mode’s state is mixed with a moving mode’s state the velocity of the stationary mode is set to \(30\frac{m}{s}\), the expected velocity following a velocity jump event.

B. SMME-n Configuration

The SMME-n filter can directly model the system described in section IV. Two different sojourn distributions are used; exponential, and pseudo-Gaussian. The decay constant in the exponential distribution was chosen such that it would have an identical sojourn distribution to the IMM, see Appendix C. This allows a more direct comparison to the IMM filter.

The procedure for generating the state and covariance of a hypothesized event is given in Appendix A. For a velocity event the added uncertainty in its initial velocity is also taken in account.

V. RESULTS AND DISCUSSION

A. Performance Metrics

One of the metrics used to test the quality of a filter is the mean Chi-Square (\(\chi^2\)) metric. This metric is a test of filter consistency for filters that use a Gaussian assumption. Filter consistency is important since it allows target tracks to be correctly associated with observations. According to [2] filter consistency is characterized by three criteria:

1) The state errors should be acceptable as zero mean and have magnitude commensurate with the state covariance as yielded by the filter.
2) The innovations should also have the same property.
3) The innovations should be acceptable as white.

Monte-Carlo (MC) simulation allows the filter to be tested for consistency by calculating the mean \(\chi^2\). Informally, a filter is consistent if the mean \(\chi^2\) is close to the number of degrees of freedom (DOF) of the target’s state vector. Using simulations a filter can be tested for consistency at specific moments in time or across a trajectory. A filter can not be consistent at any arbitrary moment in time unless it knows the maneuvers in advance. A filter can not be consistent across a whole trajectory unless it knows the process that controls the maneuvers.

Since the DOF of the target’s state vector changes with time the test is performed on the observations in this paper. The sensor observation has one DOF, therefore the expected mean \(\chi^2\) is one. The \(\chi^2\) is calculated according to the following equations:

\[
\chi^2 = v(k)^T S(k)^{-1} v(k) \quad (16)
\]

\[
v(k) = \tilde{z}(k|k-1) - z(k) \quad (17)
\]

\[
S(k) = HP(k|k-1)H^T + \sigma_r^2 \quad (18)
\]

where \(v\) and \(S\) are the residual and residual covariance.

When a fixed event sequence is used, the filters are tested for consistency as a function of time across the trajectory. A mean \(\chi^2\) test is performed at the time of each observation, which is possible since the observation times are fixed across MC trials. Performing this test on a fixed trajectory can provide information on the filter’s response to a maneuver. However, even an optimal filter will not be consistent across the whole trajectory.

One reason an optimal filter will not be consistent across the whole trajectory is that the covariance needs to be adjusted to take in account the possibility of a maneuver. The covariance is adjusted by “inflating” it, which causes the first condition to not be met.

Mean position error (PE) and mean velocity error (VE) are used as a metrics to measure the accuracy of the filters. Position/velocity error is the absolute value of the difference between the filter’s estimated position/velocity and the true target position. The error is calculated immediately after a measurement is processed.

B. Results

Simulation results were generated from 5000 MC trials. The following shorthand is used through out this section; “Exp” for SMME-n with an exponential sojourn distribution, and “P-Gauss” for SMME-n with a pseudo-Gaussian distribution.

![Example Trajectory for a Single MC Trial](image)

Fig. 4. A plot showing the position and velocity across a single MC trial for the first scenario.

In the first scenario a fixed event sequence is used. The target starts out in the moving mode, then proceeds to have a stationary event following by two velocity jump events. The velocity events have magnitudes \(20\frac{m}{s}\) and \(30\frac{m}{s}\), in that order. The sojourn time for all events is twelve seconds. The observation noise, plant noise, and initial state are varied across MC trials. A single MC test has been plotted in Figure 4.

Mean \(\chi^2\), mean PE, and mean VE are plotted as a function of time in Figures 5, 6, and 7 respectively. The IMM Filter does not correctly take in account the possibility of an event while in the stationary mode. This can be seen since it has a \(\chi^2\) of one from 12s to 24s. As a result when the second event occurs there is a large spike in \(\chi^2\). On the other hand both of the SMME-n filters have a \(\chi^2\) less than one before an event occurs, significantly reducing the size of their spikes.

In the second scenario the target’s trajectory is varied and lasts for 48 seconds. The sojourn distribution for maneuvers is a pseudo-Gaussian distribution with a mean of 12 seconds and a standard deviation of 3 seconds. Initially the target always starts out in the moving mode. The timing of events, type
of events, observation noise, plant noise, and initial state are varied across MC trials.

![Fig. 5. Comparison of mean χ² statistic for a fixed trajectory. Note the log-scale in the y-axis. N = 6 and Tₑ = 0.5s](image)

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Mean Chi-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50</td>
<td>10 0 10 1</td>
</tr>
</tbody>
</table>

![Fig. 6. Comparison of position error for a fixed trajectory. N = 6 and Tₑ = 0.5s](image)

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Mean Position Error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50</td>
<td>0 5 10 15 20 25</td>
</tr>
</tbody>
</table>

![Fig. 7. Comparison of velocity error for a fixed trajectory. N = 6 and Tₑ = 0.5s](image)

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Mean Velocity Error (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50</td>
<td>0 10 20 30 40 50</td>
</tr>
</tbody>
</table>

Table I summarizes the performance of the three filters in the second scenario. As expected the P-Gauss filter performed the best in almost all the metrics. The IMM slightly outperformed it in the max position error metric. By comparing the Exp filter to the P-Gauss filter the performance gained by using a correct sojourn distribution can be inferred. By comparing the IMM to Exp filter the performance gained by correct modeling of maneuvers can be inferred. The largest performance gains came from correct modeling of maneuvers and a more modest improvement from correct modeling of the sojourn distribution.

Increasing the hypotheses frequency in SMME-n can improve performance; however the maximum number of hypotheses also needs to be increased. Table II shows how increasing the hypotheses frequency increases filter accuracy, given enough hypotheses. If there are not enough hypotheses then the performance is degraded, Table III.

![Fig. 8. Comparison of filter performance for a fixed trajectory. N = 6 and Tₑ = 0.5s](image)

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Mean Chi-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 10 20 30 40 50</td>
<td>10 0 10 1</td>
</tr>
</tbody>
</table>

**TABLE I**

<table>
<thead>
<tr>
<th>N</th>
<th>Tₑ</th>
<th>Mean χ²</th>
<th>Mean PE (m)</th>
<th>Max PE (m)</th>
<th>Mean VE (m/s)</th>
<th>Max VE (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>3s</td>
<td>1.30</td>
<td>5.82</td>
<td>60.21</td>
<td>5.48</td>
<td>123.40</td>
</tr>
<tr>
<td>6</td>
<td>2s</td>
<td>1.05</td>
<td>5.55</td>
<td>60.14</td>
<td>5.48</td>
<td>123.39</td>
</tr>
<tr>
<td>6</td>
<td>1s</td>
<td>0.94</td>
<td>5.46</td>
<td>60.21</td>
<td>5.48</td>
<td>123.39</td>
</tr>
<tr>
<td>6</td>
<td>0.5s</td>
<td>0.99</td>
<td>5.46</td>
<td>60.21</td>
<td>5.48</td>
<td>123.39</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>N</th>
<th>Tₑ</th>
<th>Mean χ²</th>
<th>Mean PE (m)</th>
<th>Max PE (m)</th>
<th>Mean VE (m/s)</th>
<th>Max VE (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2s</td>
<td>3.13</td>
<td>6.09</td>
<td>60.12</td>
<td>5.57</td>
<td>121.86</td>
</tr>
<tr>
<td>6</td>
<td>1s</td>
<td>1.11</td>
<td>5.59</td>
<td>60.21</td>
<td>5.57</td>
<td>121.86</td>
</tr>
<tr>
<td>6</td>
<td>0.5s</td>
<td>1.05</td>
<td>5.51</td>
<td>60.12</td>
<td>5.57</td>
<td>121.86</td>
</tr>
</tbody>
</table>

**TABLE III**

When applying the filter to real world problems selecting an accurate SM process across all target types is difficult. For example the shape of the distribution might be known, but the exact parameters can vary widely across targets. Having an individual distribution for each type of possible target is not practical for computational reasons and undesirable because it would make the filter target specific. Using an extremely broad distribution requires increasing the maximum number of hypotheses and reduces the performance gained by using a SM process.

The runtime of SMME-n was typically significantly greater than the IMM algorithm. SMME-n needs to consider a greater number of hypotheses since its maneuver model is more flexible. While propagating the filter SMME-n can perform
multiple spawning/pruning calculations, while the IMM algorithm would only perform a single interaction.

VI. CONCLUSION

The proposed sequential filter, SMME-n, uses several strategies to improve its target state estimate. It models maneuvers as instantaneous events that can cause a discontinuous change to the target’s state and mode. This is more flexible than the MM approach, which only changes the target’s mode. Events are hypothesized at a frequency independent of the sensor’s observation frequency, allowing maneuvers to be modeled better. The maneuver sojourn time is modeled using the more realistic and flexible SM process.

In the test scenarios, the largest performance boost came from the maneuver model, and a relatively modest improvement came from the SM sojourn distribution and observation independent hypothesis frequency. The SMME-n outperformed the IMM filter in terms of track accuracy and consistency. However, the performance gains come at a cost of increased algorithmic complexity and increased runtime.

VII. ACKNOWLEDGMENTS

The authors would like to thank the following people for providing feedback and support for this work: Gary Montgomery, Moses Chan, Eamon Barrett, Paul Reifler, Paul Yang, and Joann Luu.

APPENDIX

A. Accounting for Event Temporal Uncertainty

When an event is hypothesized, the event time is hypothesized to occur inside a time interval of length $T_e$ centered around time $t$. The new hypothesis will have its state set to the the expected state and covariance set to the parent hypotheses’ covariance, plus the maneuver sojourn below:

$$\bar{x(t)} = E[x(t|\tau)] = \frac{1}{T_e} \int_{t_a}^{t_b} x(t|\tau)d\tau$$

$$P(t) = E[(x(t|\tau) - \bar{x(t)})(x(t|\tau) - \bar{x(t)})^T]$$

$$= \frac{1}{T_e} \int_{t_a}^{t_b} [x(t|\tau) - \bar{x(\tau)})]...[\tau]d\tau$$

where $x(t|\tau)$ is the state at time $t$ given that an event occurred at time $\tau$, $t_a = t + T_e/2$ and $t_b = t - T_e/2.$

B. Pseudo-Gaussian Density Function

A pseudo-Gaussian distribution is used for the simulated target’s sojourn distribution. A pseudo-Gaussian distribution is a renormalized Gaussian distribution that has the probability of a negative sojourn time set to zero. The PDF for this distribution is given below:

$$p(\tau) = \begin{cases} \alpha \frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{(\tau - \mu)^2}{2\sigma^2}} & \tau > 0 \\ 0 & \tau \leq 0 \end{cases}$$

where $\alpha$ is a normalization constant, $\mu$ is the mean and $\sigma$ is the standard deviation of the distributions. See Figure 1 for a plot of this distribution.

C. Exponential Density Function

A Markov process has an exponentially decaying sojourn distribution. The PDF for this distribution is given below:

$$p(\tau) = \begin{cases} \lambda e^{-\lambda \tau} & \tau > 0 \\ 0 & \tau \leq 0 \end{cases}$$

where $\lambda \geq 0$ is the decay constant. The probability of state transitioning back into the same state in a discrete time Markov chain can be calculated from a continuous time distribution:

$$\lambda = -\frac{\log(\pi_{ii})}{T}$$

where $\pi_{ii}$ is the probability of the system staying in mode $i$ for a time period of length $T$. 

D. Hypothesis Likelihood Derivation

Using Bayes’ formula the posterior probability of hypothesis $H$ being correct given measurement sequence $Z^k = \{z(1), \ldots, z(k)\}$ at time $t \geq t_k$ is:

$$\mu(k, t) \triangleq Pr(H|Z^k, t) = \frac{p(Z^k|H)Pr(H|t)}{p(Z^k|t)}$$

Note that $p(Z^k|H) = p(Z^k|H, t)$. The denominator $p(Z^k|t)$ is independent of the hypothesis and can therefore be treated as a normalization constant.

Looking at the likelihood function in the numerator on the right hand side, and noting that $Z^k = \{z(k), Z^{k-1}\}$:

$$p(Z^k|H) = p(z(k)|Z^{k-1}, H)p(Z^{k-1}|H)$$

The first term on the right hand side, under a linear Gaussian assumption, is a likelihood function as defined below:

$$p(z(k)|Z^{k-1}, H) \triangleq N[z(k); v(k), S(k)]$$

where $v$ and $S$ is the residual and residual covariance.

To reduce the effects of floating point precision issues it’s advantageous to normalize the measurement likelihood sequence across all the hypotheses. After each new measurement is received a new normalization constant is calculated.

$$\gamma(k) \triangleq \alpha p(Z^k|H) = \alpha p(z(k)|Z^{k-1}, H)p(Z^{k-1}|H)$$

where $\alpha$ is a normalization constant. Replacing $p(Z^k|H)$ with $\gamma(k)$ into equation 25 will yield the same answer as if $p(Z^k|H)$ was used since the normalization constant is canceled out:

$$Pr(H|Z^k, t) = \frac{\gamma(k)Pr(H|t)}{\sum_i \gamma_i(k)Pr(H_i|t)}$$

The probability of the hypotheses’ event sequence $e^N \triangleq \{e_1, \ldots, e_N\}$ at current time $t$ is $Pr(H|t)$.

$$Pr(H|t) = Pr(H|e^N)Pr(e \in E | \tau > t - t_e)$$

where $\tau$ is the true sojourn time, and $t_e \leq t$ is the time of the previous event. The first term on the right side represents the probability of $e^N$ occurring and the right most term
Therefore a normalization constant is needed to ensure all hypotheses will be available. However, the probability of event \( e_i \) hypothesized to occur between \( \tau_i \) is the initial probability of the hypothesis, \( m_i \) is the mode the target is in after the \( i \)-th event, \( P_r(e_i|m_{i-1}) \) is the probability of event \( e_i \) occurring while in mode \( m_{i-1} \), \( t_{ab} = \{ \tau | t_a < \tau < t_b \} \) is a set of sojourn times that event \( e_i \) is hypothesized to occur between. Rewriting the equation in recursive form yields:

\[
Pr(H|e^N) = \frac{\mu_0 \prod_{j=1}^{N} [Pr(e_i|m_{i-1}) \times Pr(e \in E|m_{i-1}, t_{ab})]}{Pr(e \in E|m_{N-1})} (30)
\]

where \( \mu_0 \) is the initial probability of the hypothesis. Decomposing the first term to get:

\[
Pr(H|e^N) = P_r(H|e^{N-1})P_r(e_N|m_{N-1}) (31)
\]

The total probability theorem states that the sum of the probabilities for each hypothesis adds up to one. However because of pruning not all hypotheses will be available. Therefore a normalization constant is needed to ensure all hypotheses add up to one.

\[
\zeta(t) \triangleq \frac{Pr(H|t)}{\sum_{i} Pr(H^i|t)} (32)
\]

If an optimal filter (infinite number of hypotheses) is used then \( \zeta^*(t) \) will equal \( Pr(H^i|t) \). Using equations 28 and 32 the probability of a hypothesis being correct is approximated by:

\[
u^*(t|k) \approx \frac{\gamma^*(k)\zeta^*(t)}{\sum_{i} \gamma^*(k)\zeta^*(t)} (33)
\]

REFERENCES


