A multiple-frame multiple-hypothesis method for tracking at low SNR

John H. Greenewald [a], Stanton H. Musick [b ∞]
[a] Veridian Engineering, 5200 Springfield Pike, Dayton, OH
[b] Sensors Directorate, Air Force Research Laboratory, Wright-Patterson AFB, OH

ABSTRACT

This paper develops a multiple-frame multiple-hypothesis tracking (MF-MHT) method and applies it to the problem of maintaining track on a single moving target from dim images of the target scene. From measurements collected over several frames, the MF-MHT method generates multiple hypotheses concerning the trajectory of the target. Taken together, these hypotheses provide a smoothed and reliable estimate of the target state.

This work supports TENET, an Air Force Research Laboratory project that is developing nonlinear estimation techniques for tracking. TENET software was used to simulate both target dynamics and sensor measurements over a series of Monte Carlo experiments conducted at various signal-to-noise ratios (SNRs). Results are presented that compare computational complexity and accuracy of MF-MHT to two previously-documented nonlinear approaches to predetection tracking, a finite difference scheme and a particle filter method. Results show that MF-MHT requires about 2-3 dB more SNR to compete with the nonlinear methods on an equal footing.

Keywords: tracking, multiple hypothesis tracking, predetection filtering, track-before-detect, nonlinear filtering

1. INTRODUCTION

Situations where sensors produce measurements of moving targets at low signal-to-noise ratios (SNR) challenge the ability of a tracker to maintain an effective estimate of the target state. For example, this situation can occur when a target is at the extreme range of the sensor. It is well known that predetection tracking can be used in such situations to avoid information losses caused by data thresholding[1]. Predetection tracking methods are those that use pixelized, unthresholded measurement data as the estimation filter input, in preference to the usual thresholded partial target state measurements such as range and range-rate.

Predetection methods, also known as track-before-detect (TBD) methods, are implemented with a nonlinear filter (NLF) that estimates a general density, i.e. one without a presumed mathematical form. Estimation algorithms in nonlinear filtering problems parallel Kalman algorithms for linear problems in that both contain time-propagation and measurement-update stages that are invoked separately. Whereas Kalman can implement fairly simple algorithms to propagate and update a joint Gaussian density, NLF must use a numerical technique to propagate a general density through time and a Bayesian scheme at measurement update time. Although the two methods achieve similar results for their respective densities, NLF algorithms are considerably more complex and generally require many more computer cycles, especially during the time-propagation stage.

The computational burdens associated with solving the propagation equations in nonlinear filtering represent a barrier to the usefulness of this otherwise promising estimation approach. In recognition of such NLF issues, the Air Force Research Laboratory (AFRL) is fostering research in this area through a challenge problem that it calls TENET, Techniques for Nonlinear Estimation of Tracks. TENET's aim is to develop NLF solution methods that produce robust solutions and execute in real or near-real time, and to demonstrate this NLF technology in appropriate applications. Tracking dim targets is one such application.

∞ Contact information: john.greenewald@veridian.com, (937) 255-7569; stanton.musick@wpafb.af.mil, (937) 255-1115 x4292. Presented at Signal Processing, Sensor Fusion, and Target Recognition XI, SPIE AeroSense '02, 1-5 April 2002.

A previous paper documents two NLF methods that use TENET scenarios and metrics [2]. These methods include a finite difference scheme named alternating direction implicit (ADI), and a particle filter (PF) method. The interested reader is referred to the TENET website where NLF software and related materials may be found [3].

This paper introduces a tracking method that has its roots in traditional implementations, a method that we call multiple-frame, multiple-hypothesis tracking, MF-MHT. Multiple hypothesis tracking methods like MF-MHT have typically been applied to multi-target tracking when SNR levels are high and Kalman techniques are appropriate[1,4]. MHT methods have worked well in such applications because of their ability to handle the large combinatorial problems that attend the data association step, and because at high SNR there are few false alarms to confuse the picture and exacerbate the data association problem. Although we recognize that MHT methods are not tailored to low SNR applications, they were nonetheless used here in order to examine the limits of this well-regarded method in a stressful setting. We reasoned that if MF-MHT could perform nearly as well as the more computationally burdensome NLF methods, then perhaps the NLF implementations we had made were on a wrong tack and were in need of major repair. In addition, a high-performing MHT-type method provides a believable baseline for judging problem difficulty and performance.

The remainder of this paper is organized as follows. Section 2 describes the TENET Simulator, focusing on its target motion and scene generation capabilities. Section 3 describes the MF-MHT algorithm that was implemented for this study. Section 4 compares MF-MHT performance with the two previous benchmark trackers, namely the particle filter and the finite difference scheme methods. Section 5 presents conclusions.

2. TENET SIMULATOR

The TENET Simulator is a suite of MATLAB© functions that provide a test bed for evaluating nonlinear filtering methods. TENET materials, including MATLAB© software, software documentation and publications, are openly available at https://www.tenet.vdl.afrl.af.mil. AFRL hopes that researchers will download and use these tools to contribute to the science and technology of nonlinear filtering. We would be pleased to hear from you if you use these materials.

Currently, the TENET Simulator implements scenarios involving a single moving target and a single imaging sensor. The single target subtends one pixel and moves at nearly constant velocity (NCV) in a two-dimensional target region. The size of the target region, the initial position and velocity of the target, the target diffusion rate, and the run duration were all chosen judiciously so the target would not leave the region.

A sensor provides noisy imagery of the entire target region once per second at a specified SNR. Although the target is not visible to the naked eye in any image sequence where SNR is low, over many measurements the target-containing pixels tend to exhibit greater intensity than the non-target pixels, a fact that predetection methods are able to exploit. At measurement update, the ADI and PF methods process the unthresholded image data using Bayesian techniques, whereas MF-MHT uses the usual Kalman update algebra. To help focus these results on the track maintenance problem, 1) the mean of the estimate is initialized in every run at the actual target location, and 2) mis-modeling between truth and filter is non-existent.

The simulator implements a Monte Carlo experiment in which the target trajectory and the sensor measurements vary during each run. The ensemble of run outputs from such an experiment is a study, from which performance can be evaluated from ensemble statistics. To maintain fairness in all experiments, each method was initialized identically and each was presented the same image sequence during corresponding runs. Finally, SNR was the primary independent variable in all of the results presented here.

With the addition of the MF-MHT method, TENET now contains three tracker implementations, one based on particle filter methods, another on finite difference methods that use the alternating direction implicit algorithm, and finally the newest MF-MHT method. These simple baseline implementations provide three concrete illustrations of how to set up, solve and evaluate a track estimation problem using nonlinear filtering methods. The TENET Simulator will accommodate new tracker implementations in addition to the baseline set. The performance of any new tracking algorithm can be compared to the baseline methods using the provided metrics.

2.1 Target dynamics

Let the following stochastic differential equation describe the motion of a single target with state $\mathbf{x}_t$ between observations at times t_k.

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + \mathbf{G}(\mathbf{x}_t, t)d\beta_t, \quad t_k \le t < t_{k+1} \tag{1}$$

Here $\mathbf{x}_t$ and $\mathbf{f}(.)$ are n-vectors, $\mathbf{G}(.)$ is an $n \times r$ matrix function, and $d\beta_t$ is an r-vector Brownian motion process with $E\{d\beta_t d\beta_t^{\mathrm{T}}\} = \mathbf{Q}(t)dt$. Because Eq. (1) is general, it can describe all manner of complex stochastic dynamics.

In this study, movement was restricted to a 2D target region where the target maintains nearly constant velocity with state vector $\mathbf{x}_t = \begin{pmatrix} x & v_x & y & v_y \end{pmatrix}^{\mathrm{T}}$, and dynamics $\mathbf{f}(\mathbf{x}_t) = \begin{pmatrix} v_x & 0 & v_y & 0 \end{pmatrix}^{\mathrm{T}}$. In this very simple physics-based model, the TENET target diffuses at an equal rate q in both directions. Thus

$$\mathbf{G} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \mathbf{Q} = \begin{pmatrix} q & 0 \\ 0 & q \end{pmatrix} \tag{2}$$

The TENET Simulator creates stochastic target trajectories per Eq. (1). These trajectories have randomized initial velocities, and have assigned values for diffusion strength Q, the time step between measurement updates, and the total time-step count. When Q values are low and initial speed is high, the trajectories have a shape resembling a smooth arc, whereas the trajectories are jagged and random in their appearance when Q values are high and initial speed is low.

2.2 Sensor image model

Just as sensor images are uncorrelated in time, so their pixel-to-pixel intensities are uncorrelated in space. The model for producing such images must generate randomized background noise in target-free pixels, and a randomized intensity characteristic of the sensor-target interaction in the target-containing pixel. As it happens in TENET, all images needed to conduct a run are formed at run start from knowledge of noise and target densities, and from the target's time-varying position.

We adopt two Rayleigh distributions to describe the intensity in target-free and target-containing pixels. If there are M pixels labeled $i = 1, \ldots, M$, and y_i denotes intensity in the i-th pixel, $p_o(y_i)$ denotes the density of background noise in target-free pixels, and $p_1(y_g)$ denotes the density of target intensity in the target pixel y_g, we can write

$$p_0(y_i) = y_i \exp\left(-y_i^2 / 2\right) \tag{3}$$

$$p_1(y_g) = \frac{y_g}{1+\lambda} \exp\left(-y_g^2 / 2(1+\lambda)\right) \tag{4}$$

Equation (4) is dependent only on the SNR parameter λ. It is straightforward and computationally inexpensive to generate images by random sampling from these two Rayleigh densities.

Note that the information-based quantity known as the divergence $L(p, q)$ (a measure of the separation between densities p and q) can be used as an "effective" signal-to-noise ratio[2]. For the Rayleigh densities in (3) and (4), the divergence can be calculated in closed form as $L(p_o, p_1) = \lambda^2 \big/ (\lambda + 1)$, a form that was used consistently in this study to designate SNR when discussing or presenting results. Generally, we will convert effective SNR to "units" of decibels using $\text{SNR} = 10\ \log[\lambda^2 \big/ (\lambda + 1)]$.

See Figures 1 and 2 for examples of an unthresholded and thresholded sensor image .

3. MULTIPLE-FRAME MULTIPLE-HYPOTHESIS TRACKER

MHT techniques have been used successfully in a wide variety of real tracking applications and today are considered the most robust and accurate techniques available. These techniques are typically implemented to account for multiple targets, variable maneuver models, and a host of other special attributes that accompany real tracking problems. With TENET's assumptions of one target and one steady maneuver, the challenge problem does not seriously challenge MHT capabilities except in one area -- the difficulty of identifying the real target among the huge numbers of false alarms that occur at low SNR. In this section we discuss the MHT method used here, which we call MF-MHT. We focus on implementation issues and in particular on our strategy for dealing with the large numbers of false alarms that arise as SNR drops off.

As with most MHT-type techniques, MF-MHT derives its benefits from maintaining multiple hypotheses from which to choose a confirmed state estimate in a delayed declaration. However, since MF-MHT requires neither management of a large hypothesis tree nor hypothesis pruning, it avoids two major complexities usually associated with MHT methods.

MF-MHT is implemented using the most-recent N_{Frames} images and N_{ST} soft tracks, where both values are constants. A *soft track* is an estimated target trajectory with measurements assigned and processed up through the current time. Soft tracks are grouped in sets called *track hypotheses*, each set being a full explanation of the current target picture. In general, different track hypotheses propose different numbers of targets because they use different sequences of data associations between the image-derived measurements and the postulated tracks.

In this implementation of MF-MHT, the number of tracks remains fixed (new tracks are not initialized after run start, and old tracks are not pruned). This "unity ground rule" leads to the conclusion that there is one and only one track at all times. The implications of the unity ground rule will be pointed out as the discussion unfolds.

Each track hypothesis in MF-MHT carries a score representing its overall likelihood to explain the current target picture. This likelihood score is a composite of heuristic measures of likelihood in both the propagation and update stages, and is formed over the interval from track initiation to the present. Shortly, we shall discuss the method for computing these scores. The track hypothesis with the highest score is the *best current hypothesis*.

If we look back over the last N_{Frames} images, the best predecessor hypothesis is called the *confirmed hypothesis.* Clearly, the confirmed hypothesis must contain the track or tracks that led to the best current hypothesis. The tracks in the confirmed hypothesis are called *confirmed tracks* or *hard tracks*. The declaration that yields hard tracks can be made at any time, and is made in this study after each image update. Experience shows that the hard tracks in the confirmed hypothesis are a better representation of the truth precisely because a delay is introduced. In this implementation the delay is $N_{Frames} = 5$ images (equivalent to 5 sec). Furthermore, there will be just one hard track in the confirmed hypothesis because of the unity ground rule introduced above.

To understand how tracks evolve in MF-MHT, first consider how a single track at frame k associates with gated measurements (*reports*) at frame k+1. (Gating is discussed below.) Every distinct acceptable report-to-track association creates another new track (each report can be used in only one track), potentially resulting in a large number of new tracks from each new image. As this fan out is repeated on successive frames, the target count grows rapidly and can easily overwhelm available computational resources. Therefore, an important artifice for limiting the number of track hypotheses is limiting the number of *candidate reports* that can be associated with each existing track. Denoting this number as

$N_{Candidates}$ and assuming that exactly $N_{Candidates}$ report-to-track associations are made in each image for each track, the number of soft tracks in the problem will grow to the following value in steady state.

$$N_{ST} = (N_{Candidates})^{N_{Frames}-1} \tag{5}$$

In this implementation of MF-MHT, $N_{Candidates} = 7$ reports so $N_{ST} = 7^4 = 2401$ tracks in steady state, a count that is present from the 5-th image to the last.

Candidate reports are the pixel locations of the $N_{Candidates}$ measurements with the largest intensities in and around an update gate. The *update gate* defines the part of the target region that is searched to identify candidate reports for a particular track. In this study, the update gate at time t_k^- was a circle with center at the expected track location and radius equal to the geometric mean of the standard deviations of position. Thus the center is $(\hat{x}, \hat{y})$ and the radius is $(\sigma_x\ \sigma_y)^{1/2}$, where all values are taken from track estimates at time t_k^-.

It can happen that there are reports that lie outside the update gate that would be reasonable to include. When such reports are within two gate radii, the philosophy adopted here is that they can be candidates too, but at a price. The price is to reduce their strength per the following weighting formula

$$l_{Weighted} = \begin{cases} l & if\ r \le gate \\ l \times r^{-1} & if\ \ gate < r < 3 \times gate \end{cases} \tag{6}$$

where each l is an observed intensity in a pixel, and r is the distance in pixels from the gate center to the pixel center. The chosen reports are the best $N_{Candidates}$ from the weighted set of Eq. (6). Note that weightings are only used for selecting candidates. Actual measurements are used in updating track estimates and in computing the measurement update component of the likelihood score.

Before discussing track likelihood scoring, we need to introduce the standard Kalman implementation that was used to form estimates. We modeled target dynamics using the standard discrete-time Markov form of the Kalman filter.

$$\hat{\mathbf{x}}_{k+1} = \Phi\ \ \hat{\mathbf{x}}_k \tag{7}$$

$$\hat{P}_{k+1} = \Phi\ \ \hat{P}_k\ \ \Phi^T + Q \tag{8}$$

where the transition matrix for a time step of Δt seconds is

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

and Q is defined in (2).

Measurement update is also patterned after standard Kalman procedures, beginning with the measurement equation

$$\tilde{z} = H\ \hat{x}_k + v_k \tag{10}$$

where the measurement distribution and noise strength matrices are

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

$$R = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \tag{12}$$

Now defining S as the variance of the innovation

$$S = H \;\; \hat{P}_{k+1} \;\; H^T + R \tag{13}$$

the kinematic likelihood score for each track is updated using the position innovation.

$$L_{Kin}^{k+1} = \frac{\exp(-(z-\hat{z})^2 / 2P_{Gate}^2)}{\sqrt{|S|}} \times L_{Kin}^{k} \tag{14}$$

Similarly, the measurement likelihood score is updated as

$$L_{Sen}^{k+1} = \mu \times L_{Sen}^{k} \tag{15}$$

where μ is the observed intensity normalized on a scale of zero to one.

After all track estimates and likelihoods have been updated, the kinematic and sensor likelihoods are then multiplied with one another to form the combined likelihood score as

$$L = L_{Sen} \times L_{Kin} \tag{16}$$

The maximum likelihood soft track determines which estimated state and covariance values at time $k - N_{Frames} + 1$ update the target estimates at time $k - N_{Frames}$. The position estimate declaration delay is therefore the number of frames minus one.

As noted above, each track hypothesis carries a *likelihood score* representing its ability to explain the current target picture. A track's likelihood score is comprised of components that evaluate both kinematics and sensor image measurements. These two components must be multiplied together to achieve best performance.

The best soft track of the 2401 hypotheses is that track with the highest likelihood score. This evaluation was made based on the product of the product of the kinematic and measurement component scores. To prevent underflow, the composite likelihood score was renormalized to unity after each measurement update.

4. RESULTS

The purpose of the experiments presented in this section is to compare the three tracking methods (PF, ADI and MF-MHT) to each other. We discuss metrics, experimental setup, and numerical results.

4.1 Metrics

RMS position and velocity error were chosen as metrics to illustrate performance in this paper. Error is defined as the difference between estimate and truth, $\mathbf{e}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t$, where the estimate $\hat{\mathbf{x}}_t$ is computed as the mean of the a posteriori density estimate $p(\mathbf{x}_t \mid \mathbf{Y}_t)$. Although RMS accuracy alone does not provide a thorough characterization of performance in a general tracking problem, it would seem to be appropriate in this special case where there is just one target (the usual multi-track metrics degenerate or disappear) and track state initialization is exceptionally good (which greatly diminishes the normal convergence transients).

As noted previously, it is expensive and unnecessary to compute the joint density for the two nonlinear methods (PF and ADI) over the entire range of motion. Instead, a *computational gate* was established on which to produce a solution, this gate being a small fixed-size subset of the full target region. As the target moves, the computational gate is translated to keep the position of the propagated target approximately centered in the gate. Because estimated position is used in the gate control algorithm, the actual target may drift and exit the gate when the filter estimate is poor. When the target exits the computational gate, we declare a *lost lock* event, the run output is discarded, and the run is restarted from time zero. These actions were necessary because, without target observations, the filter has no mechanism to recover and usually diverges rapidly. As expected, lost locks occur more frequently at the lower SNR levels where estimates are poorest, and at run startup before the covariance cross terms are developed. Lost lock events indicate problems in the estimation process, and were therefore logged.

4.2 Experimental setup

At each observation, the simulated sensor imaged the entire target region and produced a scene of 256 x 256 pixels. The intensity in each non-target pixel is governed by the Rayleigh distribution of Eq. (3) with noise power one. In the pixel with the target, the intensity is adjusted for the SNR of the study, Eq. (4). Identical simulated sensor images are input to each filter as measurements, but only the portion of the scene in the computational gate contributes to the joint density estimate. ADI and MF-MHT methods perform this gating similarly by selecting only pixels within the a predetermined range whereas PF method gates by the assigning a likelihood of zero to pixels that are outside the range of the diffusion of particles. The initial values of each track filter are chosen to match the truth. The initial density of each filter is uniform in each of the four dimensions of state $\mathbf{x}_t$ and extends over the region in the initial gate. For the results that follow, the computational gate was fixed at 10x10 pixels in (x, y).

In the case of a lost lock event, accuracy degrades precipitously and the run is effectively ruined. When this occurs, data from that run is removed from the study ensemble, and a new run is made to replace the spoiled one.

Experimental results are based on studies of 50 Monte Carlo runs of 100 seconds duration with measurements taken at the rate of one frame per second. Studies were conducted for MF-MHT, ADI and PF separately, at 2 dB intervals in the range 4-20 dB effective SNR.

4.3 Numerical results

For this study we allotted a fixed budget of floating point operations (flops) to each method since some methods are better vectorized for faster operation in MATLAB$^{©}$ than others. The flop budget for this study was 6Mflops. The results shown in Figure 3 show the RMS error over the ensemble of 50 Monte Carlo runs for each of the three filters. The particle filter method performs best and MF-MHT is slightly worse than ADI down to 6dB. The velocity error shown in Figure 4 shows that MF-MHT is twice that of PF but five times better than ADI. The position variances are comparable for all filters above 8dB and MF-MHT is comparable to ADI between 4 and 6dB as shown in Figure 5. Figure 6 shows that the velocity variance is nearly as good as PF. The processing time is comparable to PF given the number of hypotheses and particles that were selected.

5. CONCLUSION

The MF-MHT approach successfully tracks the target to about 6dB SNR. As SNR dropped below 6dB, the frequency of both break lock and lost lock events increased. The results show that RMS accuracy performance is comparable to ADI down to 6dB SNR, but that MF-MHT has faster processing that is comparable to PF. For problems where a single target must be tracked at SNR levels down to 6 dB, the MF-MHT method may prove to be the best candidate.

ACKNOWLEDGEMENT

This work was funded in part by a grant to AFRL/SN from the Air Force Office of Scientific Research, Dr. Jon Sjogren Program Manager. The authors gratefully acknowledge that support.

REFERENCES

1. Blackman and Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, 1999.

2. Musick, Greenewald, Kreucher, and Kastella, "Comparison of Particle Method and Finite Difference Nonlinear Filters for Low SNR Target Tracking", *Conference Proceedings of Fusion2001*, August 6-10, 2001.

3. Musick and Greenewald, TENET Products, https://www.tenet.vdl.afrl.af.mil, July 2001.

4. Bar-Shalom and Blair, Editors, *Multitarget-Multisensor Tracking: Applications and Advances Volume III*, Artech House, 2000.

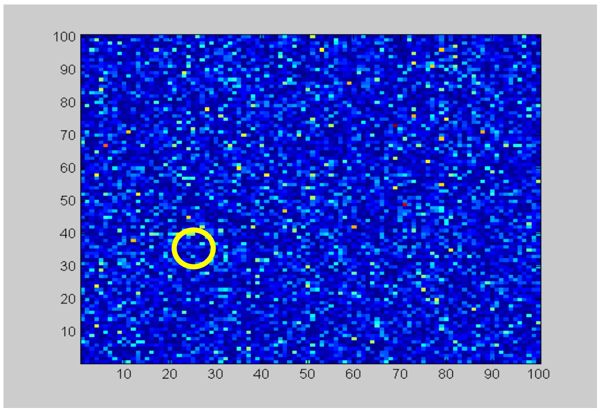

Figure 1. Unthresholded image at 6 dB effective SNR

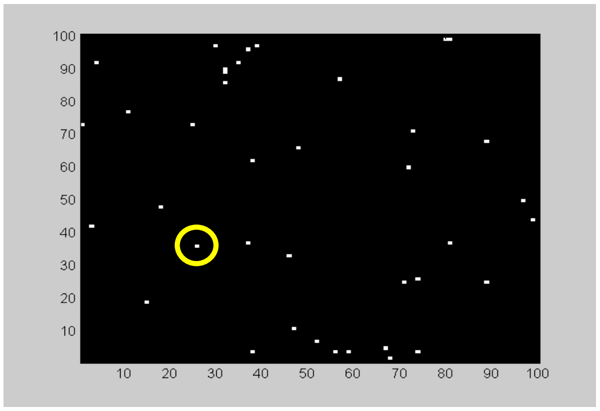

Figure 2. Thresholded image at 6 dB effective SNR

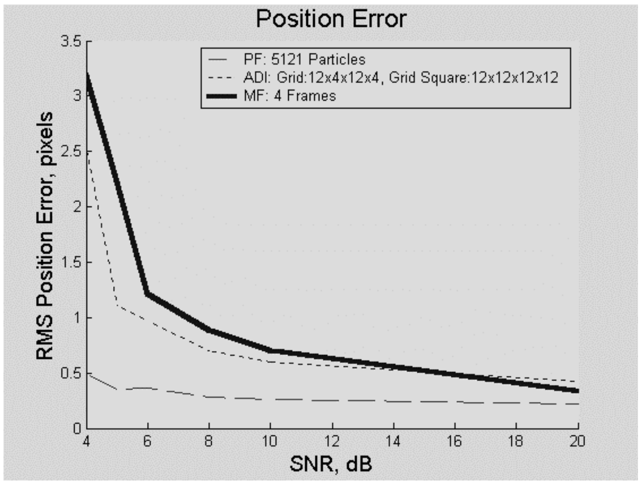

Figure 3. RMS Position Error

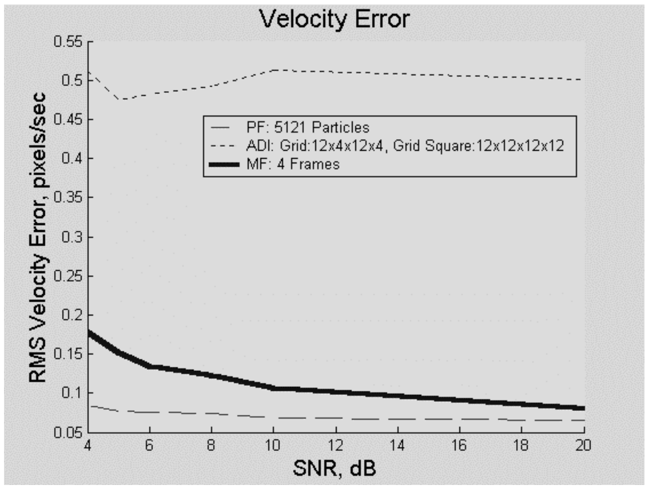

Figure 4. RMS Velocity Error

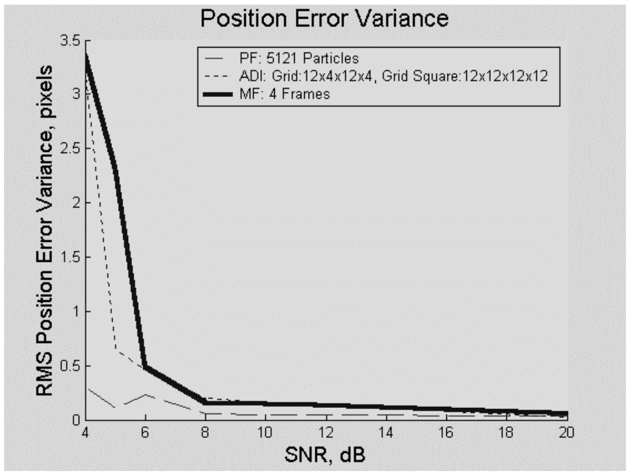

Figure 5. Position Error Variance

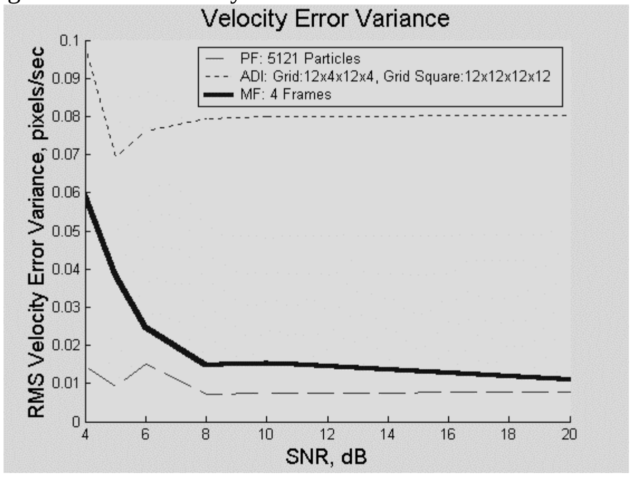

Figure 6. Velocity Error Variance

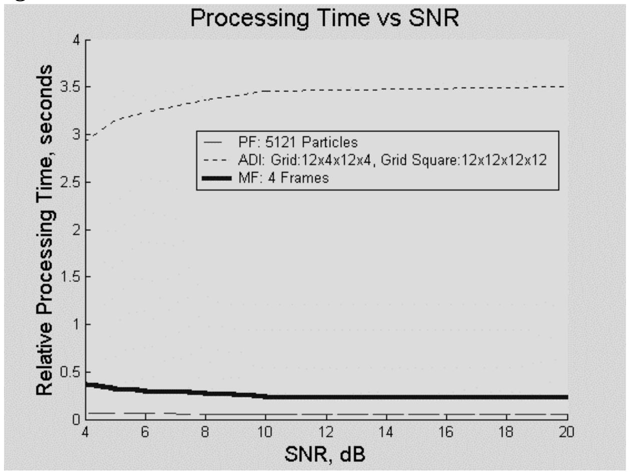

Figure 7. Processing Time