Comparison of Particle Method and Finite Difference Nonlinear Filters for Low SNR Target Tracking

Stanton Musick
Sensors, AF Research Laboratory
Wright-Patterson AFB, OH, USA
Stanton.Musick@wpafb.af.mil

John Greenewald
Veridian Engineering
Dayton, OH, USA
John.Greenewald@Veridian.com

Chris Kreucher and Keith Kastella
Veridian Systems
Ann Arbor, MI, USA
First.Last@Veridan.com

***Abstract** - This paper presents the formulation of a low SNR target tracking challenge problem that we have designed to explore the numerical and estimation performance trades of nonlinear filtering methods. We have implemented a particle filter (PF) and alternating direction implicit-based (ADI) finite difference method filter. Our initial result from comparing these methods in terms of their RMS position error is that PF betters ADI by about 60% at 2.5dB SNR, the point of maximum error reduction. More significantly, PF also appears to be much more robust against track loss at these low SNR levels. This robustness derives from the inherent adaptivity of the particle method.*

Keywords - tracking, filtering, estimation, finite difference method, particle method.

1 Introduction

Development of effective numerical methods for nonlinear tracking applications is a significant hurdle for data fusion systems. One of the most stressing applications of nonlinear filtering (NLF) is pre-detection tracking: tracking algorithms that use pixelized, unthresholded data as the filter input. This avoids the information-losing step of thresholding the data to produce partial target state measurements such as range, range-rate and bearing. If these nonlinear problems can be solved with sufficient numerical accuracy, it will allow targets to be tracked at lower signal-to-clutter+noise ratios (SCNR), enabling improved performance for detecting and tracking targets under foliage, surface vessels in the littoral zone and cruise missiles.

While it is well known that the Kalman filter is the Bayes' optimal estimator for the standard linear Gaussian problem, much less is known theoretically about predetection tracking – even when the target motion itself is linear. However, there has been much recent progress in the area of nonlinear filtering that is applicable to problems of this sort. There are now a number of viable numerical methods to solve nonlinear filtering problems. Candidates include (but are not limited to):

1) particle methods [1, 2, 7],
2) finite difference methods [4,5],
3) spectral methods [6],
4) probabilistic data association methods, and
5) multiple hypothesis, multiple frame methods.

While not fully mature, these methods have been developed to the point where they can be considered for transition to deployable systems. Their relative performance for realistic problems needs to be better understood. The actual gain in estimation performance obtained by processing predetected data has not been characterized. How this performance gain varies with SCNR and the impact of target maneuvers remains to be investigated. The nature of the numerical error incurred by each method and its impact on various parts of the target envelope remains to be systematically studied.

To promote the development of engineering expertise in this area, the Air Force Research Lab (AFRL) is sponsoring a study project to evaluate the effectiveness and relative numerical performance of nonlinear filtering algorithms. To support this effort, which is being called a challenge problem, AFRL is developing a set of synthetic, ground-truthed data sets and associated metrics for testing in the research community. We propose two approaches to generating the target motion. As a baseline, we propose to generate random target motions using a simple 2nd order model (x, x, y, y). This has some advantages from an analysis point of view in that it is simple and enables focused study on numerical NLF issues. In a subsequent approach, we propose scenarios involving multiple targets maneuvering at unspecified times with variable normal accelerations.

The primary measures of performance (MOPs) are target RMS error as a function of SNR. As part of this effort, a separate MOPs module has been developed. This code is available in the form of commented MATLAB code at https://www.tenet.vdl.afrl.af.mil.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the basics of nonlinear filtering in a Bayesian formulation and defines a simple motion model for use here. Section 2.1 presents the sensor model, and Section 2.2 presents the definition of effective signal-to-noise ratio that we use. Section 2.3 briefly presents the Sampling, Importance, Resampling (SIR) particle method filter [1,2,5] developed for these tests and Section 2.4 presents the Alternating Direction Implicit (ADI) finite difference scheme [4,5,9,10] used here. Section 2.5 presents the definitions of the performance metrics used in this evaluation. Section 3 presents results and Section 4 presents conclusions and suggests a few directions for further work.

2 Bayesian nonlinear filtering

To track a target from a sequence of measurements $\mathbf{y}_k$ made at discrete times t_k, let the target dynamics be described by an Ito stochastic differential equation [3] for the time-dependent target state $\mathbf{x}_t$

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + \mathbf{G}(\mathbf{x}_t, t)d\beta_t, \quad t \ge t \tag{1}$$

where $\mathbf{x}_t$ and $\mathbf{f}$ are n-vectors, $\mathbf{G}$ is an $n \times r$ matrix function, and $\{\beta_t, t \ge t_0\}$ is an r-vector Brownian motion process with $E\{d\beta_t d\beta_t^{\mathrm{T}}\} = \mathbf{Q}(t)dt$.

The observations up to time τ are denoted

$$Y_\tau = \{\mathbf{y}_l : t_l \le \tau\} \tag{2}$$

Between observations the evolution of the conditional density is determined by the target dynamics as characterized by the Ito equation. The time evolution of the joint density between measurements is the solution to the Fokker-Planck equation (FPE)

$$\frac{\partial p}{\partial t}(\mathbf{x}_t \mid Y_{t_k}) = L(p), \quad t_k \le t < t_{k+1}, \tag{3}$$

where

$$L(p) \equiv -\sum_{i=1}^{n} \frac{\partial(\mathbf{f}_i p)}{\partial \mathbf{x}_i} + \frac{1}{2} \sum_{i,j=1}^{n} \frac{\partial^2\left(p\left(\mathbf{GQG}^{\mathrm{T}}\right)_{ij}\right)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \tag{4}$$

with initial condition given by $p(\mathbf{x}_{t_k} \mid Y_{t_k})$.

Then given a new observation $\mathbf{y}_k$, the updated conditional density $p(\mathbf{x}_{t_k} \mid Y_{t_k})$ is obtained from the predicted density $p(\mathbf{x}_{t_k} \mid Y_{t_{k-1}})$ using Bayes' formula:

$$p(\mathbf{x}_{t_k} \mid Y_{t_k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_{t_k}) p(\mathbf{x}_{t_k} \mid Y_{t_{k-1}})}{\int p(\mathbf{y}_k \mid \mathbf{x}'_{t_k}) p(\mathbf{x}'_{t_k} \mid Y_{t_{k-1}}) d\mathbf{x}'_{t_k}} \tag{5}$$

The minimum mean square error target state estimate $\hat{\mathbf{x}}_t$ is

$$\hat{\mathbf{x}}_t = \int \mathbf{x}_t \; p(\mathbf{x}_t \mid Y_t) d\mathbf{x}_t \tag{6}$$

In this version of the challenge problem, we focus on the effect of measurement nonlinearity due to low SNR, and use a *linear* motion model, the so-called "nearly constant velocity" model with $\mathbf{x} = (x, x, y, y)^{\mathrm{T}}$, $\mathbf{f}(\mathbf{x}) = (x, 0, y, 0)^{\mathrm{T}}$, and $\mathbf{G}(t)$ and $\mathbf{Q}(t)$ given by

$$\mathbf{G} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \mathbf{Q} = \begin{pmatrix} q & 0 \\ 0 & q \end{pmatrix}. \tag{7}$$

The resulting FPE is

$$\frac{\partial p}{\partial t} = -x\frac{\partial p}{\partial x} - y\frac{\partial p}{\partial y} + \frac{q}{2}\frac{\partial^2 p}{\partial x^2} + \frac{q}{2}\frac{\partial^2 p}{\partial y^2} \tag{8}$$

2.1 Sensor model

An example of how to incorporate the likelihood function into NLF is presented by pixelized data such as point-target image-tracking applications. Envelope-detected radar data takes a similar form. In this case an image contains M pixels labeled $i = 1, \quad , M$. The measurement consists of the pixel output vector $\mathbf{y}_k = [y_{k,1}, \quad , y_{k,M}]^{\mathrm{T}}$ where the $y_{k,i}$ can be positive, real or complex, depending on the details of the sensor and signal processing. For point targets, pixel outputs are conditionally independent and

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \prod_{i=1}^{M} p(y_{k,i} \mid \mathbf{x}_k) \tag{9}$$

When there is no target in a pixel, its output statistics are determined by the background rate $p_0(y_{k,i})$. Depending on the nature of the imager, this may be modeled by Rayleigh, Poisson, or more complicated distributions. Further, it may vary with pixel index and time, depending on the clutter statistics. When the target projects into a pixel i, its output statistics will be given by $p_1(y_{k,i})$, which, again, may be pixel and time dependent and will depend on the detailed nature of the sensor and target.

Using Bayes' formula directly to update the density is cumbersome because it involves M factors in the product for each discretized value of the target state vector $\mathbf{x}_{t_k}$. This can be simplified by defining the target-space to pixel-space projection:

$$h_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ projects to pixel } i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

This projection is a binary mapping from the target state to the pixel space. Note that this is a highly nonlinear function of the target state. For Doppler-sensitive sensors such as radar or ladar, this projection will depend on the target velocity as well as its location. Using the target-space to pixel-space projection,

$$\begin{aligned} p(\mathbf{y}_k \mid \mathbf{x}_k) &= \prod_{i=1}^{M}\left(p_0(y_{k,i}) + h_i(\mathbf{x}_k)\left(p_1(y_{k,i}) - p_0(y_{k,i})\right)\right) \\ &= \kappa p_1(y_{k,i_{\mathbf{x}_k}}) / p_0(y_{k,i_{\mathbf{x}_k}}) \end{aligned} \tag{11}$$

where $i_{\mathbf{x}}$ is the target-containing pixel (i.e. $h_{i_{\mathbf{x}_k}}(\mathbf{x}_k) = 1$) and

$$\kappa = \prod_{i=1}^{M} p_0(y_{k,i}) \tag{12}$$

is a constant that can be dropped in the Bayes' formula update. With this expression only the measurement likelihood ratio

$$l(y_{k,i_{\mathbf{x}_k}}) = p_1(y_{k,i_{\mathbf{x}_k}}) / p_0(y_{k,i_{\mathbf{x}_k}}) \tag{13}$$

needs to be evaluated for each discretized value of the target state vector $\mathbf{x}_k$, a significant saving in computation.

To illustrate, for a Rayleigh target with SNR parameter λ, the probability distribution for intensity $y_{i_{\mathbf{x}_k}}$ of the target-containing pixel is

$$p_1(y_{i_{\mathbf{x}_k}}) = \frac{y_{i_{\mathbf{x}_k}}}{1+\lambda} \exp\left(- y_{i_{\mathbf{x}_k}}{}^2 / 2(1+\lambda) \right) \tag{14}$$

The distribution for the background pixels is given by the same expression with $\lambda = 0$. A little algebra shows that up to an irrelevant constant that can be dropped, the likelihood is

$$l(y_{i_{\mathbf{x}_k}}) = \exp\left(\frac{\lambda y_{i_{\mathbf{x}_k}}{}^2}{2(1+\lambda)} \right) \tag{15}$$

2.2 Effective SNR

While signal-to-noise ratio (SNR) is often used to gauge likely performance for filter processes, it is only rigorously related to performance in the case of Gaussian signals. The Kullback-Leibler discrimination, a more general quantity that is related to detection and estimation performance, is defined by

$$L(q_0; q_1) \equiv \int q_0(y) \ln\left(q_0(y) / q_1(y)\right) dy \tag{16}$$

Its symmetrized relative, the divergence, is defined by

$$L(q_0; q_1) \equiv L(q_0; q_1) + L(q_1; q_0) \tag{17}$$

The divergence is a convenient measure of effective SNR. For a Gaussian signal given by

$$p_1(y) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-(y-\lambda)^2 / 2\lambda\right) \tag{18}$$

and

$$p_0(y) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-y^2 / 2\lambda\right) \tag{19}$$

it is straightforward to find that

$$L^{Gauss}(p_0; p_1) = \lambda \tag{20}$$

For Rayleigh distributed measurements with

$$p_1(y) = \frac{y}{1+\lambda} \exp\left(-y^2 / 2(1+\lambda)\right) \tag{21}$$

$$p_0(y) = y \exp\left(-y^2 / 2\right) \tag{22}$$

we have this divergence

$$L^{Ray}(p_0; p_1) = \frac{\lambda^2}{1+\lambda} . \tag{23}$$

Note that in the large λ limit, the Rayleigh and Gaussian divergences are the same, while they differ significantly for small values of λ.

2.3 Particle method

Particle methods are a collection of Monte Carlo techniques in which the probability density is represented by a collection of N independent and identically distributed random samples, $\left\{\mathbf{x}_k^{(i)}; i = 1, \quad , N\right\}$. The samples $\mathbf{x}_k^{(i)}$, referred to as particles, are distributed according to $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k \mid Y_k)$. Then the conditional likelihood is approximated by

$$p(\mathbf{x}_k \mid Y_k) \approx \frac{1}{N} \sum_{i=1}^{N} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right) \tag{24}$$

where $\delta(\mathbf{x})$ is the Dirac δ -function in target state space.

To implement a particle filter using the technique called Sampling, Importance, Resampling (SIR), we mechanize prediction and measurement update using the approximating density above. For prediction at time k, we propagate each particle $\mathbf{x}_{k-1}^{(i)}$ by producing a single draw from $p\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{(i)}\right)$. The predicted particles are denoted $\mathbf{x}_k^{(i)}$ and the predicted density is approximated by

$$p(\mathbf{x}_k \mid Y_{k-1}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right) \tag{25}$$

To perform measurement update, we can approximate (up to a normalization constant κ)

$$p(\mathbf{x}_k \mid Y_k) \approx \kappa p(y_k \mid \mathbf{x}_k)\sum_{i=1}^{N} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right)$$
$$= \kappa \sum_{i=1}^{N} p\left(y_k \mid \mathbf{x}_k^{(i)}\right)\delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right) \quad (26)$$

where each particle is weighted by its measurement likelihood $p\left(y_k \mid \mathbf{x}_k^{(i)}\right)$. In the so-called Sequential Importance Sampling (SIS), this process is simply iterated. The weight for particle i after K steps is $w_K^i \propto \prod_{k=1}^{K} p(y_k \mid \mathbf{x}_k^{-(i)})$. SIS is not a very effective algorithm since the particle trajectories are not influenced by the measurements at all. As a result, they quickly diffuse away from the region of high likelihood and very poor estimation performance results.

This problem with SIS is corrected in the Sampling Importance Resampling (SIR) algorithm. Here a new set of particles is generated from the predicted set by resampling with likelihood $p\left(y_k \mid \mathbf{x}_k^{(i)}\right)$. Thus particles with high likelihood are sampled many times while particles with low likelihood are unlikely to appear in the resampled set. With this, we have the SIR algorithm:

Particle Filter Algorithm (SIR)

1. *Initialization,* $k = 0$:
   - ♦For $i = 1, \quad , N$ sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$ and set $k = 1$.
2. *Prediction step:*
   - ♦For $i = 1, \quad , N$ sample $\tilde{\mathbf{x}}_k^{(i)} \sim p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}\right)$.
3. *Resampling step:*
   - ♦Compute importance weights, $w_k^{(i)} = p(y_k \mid \tilde{\mathbf{x}}_k^{(i)})$.
   - ♦Normalize the weights according to $w_k^{\max} = \max_i w_k^{(i)}$; $w_k^{(i)} \leftarrow w_k^{(i)} / w_k^{\max}$.
   - ♦Generate N resampled particles $\left\{\mathbf{x}_t^{(i)}; i = 1, \quad , N\right\}$ with $\Pr\left(\mathbf{x}_k^{(i)} = \tilde{\mathbf{x}}_k^{(j)}\right) = \frac{1}{N} w_k^{(j)}$.
   - ♦Set $k \leftarrow k + 1$ and go to Step 2.

2.4 ADI finite difference method

For the finite difference filter, we used the Alternating Direction Implicit (ADI) scheme to solve the Fokker-Planck Equation. Space and time are discretized on a uniform grid with time resolution Δt and spatial resolution $\Delta\mathbf{x} = (\Delta x, \Delta x, \Delta y, \Delta y)^{\mathrm{T}}$. In this subsection p denotes a continuum solution to the FPE while g denotes a function defined on the grid that approximates p. Defining the sub-operators

$$A_1 = -x\frac{\partial}{\partial x} \quad (27)$$

$$A_2 = \frac{q}{2}\frac{\partial^2}{\partial x^2} \quad (28)$$

$$A_3 = -y\frac{\partial}{\partial y} \quad (29)$$

$$A_4 = \frac{q}{2}\frac{\partial^2}{\partial y^2} \quad (30)$$

the FPE can be written as

$$\frac{\partial p}{\partial t} = \sum_i A_i p \quad (31)$$

Discretizing in time but not in space, for now, we abbreviate $p^k = p(\mathbf{x}, t_k)$. The implicit Euler scheme [9] for the FPE is obtained by using a Taylor series in time for $p(\mathbf{x}, t_k + \Delta t)$, leading to

$$\frac{p^{k+1} - p^k}{\Delta t} = \sum_i A_i p^{k+1} + O(\Delta t) \quad (32)$$

Rearranging terms leads to

$$\left(1 - \Delta t \sum_i A_i\right)p^{k+1} = p^k + O\left((\Delta t)^2\right) \quad (33)$$

In principle, this expression can be solved for p^{k+1} by inverting the operator $1 - \Delta t \sum_i A_i$, but direct inversion is computationally expensive. An expression that is equivalent to $O\left((\Delta t)^3\right)$ but much simpler to invert is obtained by using the operator identity

$$\prod_i (1 - \Delta t A_i) \cong 1 - \Delta t \sum_i A_i + \Delta t^2 \sum_{i<j} A_i A_j \quad (34)$$

It can be shown that, if $A_{i\Delta\mathbf{x}}$ is an $O(\Delta\mathbf{x})$ discretization of A_i, then

$$\prod_i (1 - \Delta t A_{i\Delta\mathbf{x}})p^{k+1} \cong p^k \quad (35)$$

and the grid function g defined by

$$\prod_i (1 - \Delta t A_{i\Delta\mathbf{x}})g^{k+1} = g^k \quad (36)$$

approximates p up to the error terms proportional to $O(\Delta\mathbf{x}) + O(\Delta t)$ (higher order approximations can be constructed, at some additional computational cost).

To propagate the density, we solve Eq. (36) for g^{k+1} using

$$g^{k+i/4} = \left(1 - \Delta t A_{i\Delta\mathbf{x}}\right)^{-1} g^{k+(i-1)/4} A, i = 1, \quad ,4 \qquad (37)$$

The essential point to note is that each factor $\left(1 - \Delta t A_{i\Delta\mathbf{x}}\right)$ in Eq. (36) is inverted separately, simplifying the calculation. To discretize A_i, abbreviate $q(k\Delta t, x \pm \Delta x, x, y, y, \omega) = q_{x\pm1}$ with similar definitions for $q_{x\pm1}, \quad , q_{\omega\pm1}$. Using upwind differencing for the first order spatial derivatives, we have

$$A_{1\Delta\mathbf{x}} g = -\frac{x}{\Delta x}\begin{cases} q_x - q_{x-1}, & x > 0 \\ q_{x+1} - q_x, & x < 0 \end{cases} \qquad (38)$$

$$A_{2\Delta\mathbf{x}} q = \frac{q}{2\Delta x^2}\left(q_{x+1} - 2q + q_{x-1}\right) \qquad (39)$$

with similar expressions for $A_{3\Delta\mathbf{x}}$ and $A_{4\Delta\mathbf{x}}$. With this discretization, each operator $\left(1 - \Delta t A_{i\Delta\mathbf{x}}\right)$ is tridiagonal and can be inverted using Thomas's algorithm [9].

To completely specify the FPE solution, it must be restricted to a finite domain leading to an initial-boundary value problem. The finite grid domain consists of the points $((i + i_0)\Delta x, (j + j_0)\Delta x, (k + k_0)\Delta y, (l + l_0)\Delta y)^{\mathrm{T}}$, $i = 0, \quad , N_x$, $j = 0, \quad , N_x$, $k = 0, \quad , N_y$, $l = 0, \quad , N_y$, where $i_0, \quad , m_0$ are offsets used to translate the origin. The total number of grid nodes is $(N_x + 1)(N_x + 1)(N_y + 1)(N_y + 1)$ while the number of unknowns is $N = (N_x - 1)(N_x - 1)(N_y - 1)(N_y - 1)$. Boundary conditions must be specified on this hyper-cube to determine the solution to the FPE uniquely. We assume that the target signal-to-noise ratio is sufficiently high that the target has been localized. Then the density will be concentrated in some small region and decay exponentially far from this region. We assumed that the grid was large enough that the density was small on its boundary. With this motivation we used a homogenous Dirichlet condition with the solution held at 0 on the boundary.

To reduce the size of the grid required to represent the target joint density and thereby save computations, the grid was translated after each measurement to approximately maintain the target's location near its center. After each measurement update the target position estimate $(\hat{x}_t, \hat{y}_t)$ was evaluated and the grid was shifted to center of the grid near this position. This was achieved by placing the lower left corner of the spatial grid at (i_0, j_0) where

$$i_0 = \left[\hat{x} / \Delta x - N_x / 2\right] \qquad (40)$$
$$j_0 = \left[\hat{y} / \Delta y - N_y / 2\right] \qquad (41)$$

and $[x]$ denotes rounding to the nearest integer. This always translated the grid by an integral multiple of $(\Delta x, \Delta y)$. Grid nodes outside the intersection of the original and translated grids were set to 0.

2.5 Tracking metrics

The RMS values of 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 estimated density $p(\mathbf{x}_t \mid \mathbf{Y}_t)$. 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 nearly perfect (e.g. convergence times would be artificially small).

As noted previously, it is expensive and unnecessary to compute the joint density over the entire range of motion. Instead, a computational gate is established on which to produce a solution, this gate being a small fixed-size subset of the motion region. The gate must be translated using estimates of target motion as inputs to a translation control algorithm. If the filter estimate drifts so far from the truth that the target exits the gate, *lost lock* occurs. With target observations lost, the filter diverges quickly and seldom recovers. The gate translation control problem was challenging anytime filter estimates were inaccurate, e.g. at startup or at the lowest SNRs. Lost lock events were logged and their frequency computed.

3 Results

Accuracy as measured by RMS error improves for the ADI method with finer grid resolution, and improves for the PF method with increasing numbers of particles. However, as grid resolution becomes finer, the computational burden for ADI grows polynomially, but as particle count increases, the computational burden for PF grows only linearly. Given such complexities at this fundamental level, it is easier and fairer to compare the two filtering methods on the basis of equal computational burden, which translates to "equal flops" in MATLAB. The results that follow examine performance at a single constant value of 630 Kflops per study. This flop level translates to 5121 particles for PF, and a 10x10x10x10 grid of 10,000 cells for ADI.

3.1 Experiment methodology

The estimation problem is to track a single dim target moving in a 2D space using intensity images of the track area for measurements. True target motion is generated using a nearly constant velocity (NCV) model of target dynamics, a model based on the assumption that acceleration is a white noise process, $a(t) = w(t)$. The white noise in the NCV model imposes randomness in the motion so that a different true trajectory is produced on every Monte Carlo run. Target motion is represented in each filter by the same NCV model. This decision to match filter to truth avoids most mismodeling issues. For the results that follow, even the noise strength of the filter was matched to the truth.

The simulated sensor images the entire target motion region to produce a scene of 256 x 256 pixels. The intensity in each pixel is governed by a Rayleigh distribution with noise power one, Eq. (22). In the pixel holding the target, the intensity is adjusted for the SNR of the study, Eq. (21). Identical simulated sensor images are input to each filter as measurements, but only the portion of the scene in the instantaneous gate contributes to the joint density estimate.

The initial state of each filter is chosen to approximately match the truth. The initial density of each filter is uniform in each of the four dimensions (x, x, y, y), and extends over the space in the initial gate. For the results that follow, the gate was fixed at 10x10 pixels in (x, y) space.

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 each. Studies were conducted for ADI and PF separately, at 2 dB intervals in the range 4-20 dB effective SNR, Eq. (23). Altogether, 18 studies (9 each for PF and ADI) contributed to the results reported next.

3.2 Discussion

Figures 1-4 demonstrate how well each method is estimating the target state from the image data that it receives. Figure 1 shows the PF estimate of the joint probability density function (pdf) for the position pair (x, y) after that estimate has converged. Figure 2 shows the analogous estimate from the ADI method. Figures 3 and 4 are the corresponding joint pdf estimates for the velocity pair (x, y). Note that PF densities were obtained by binning each particle into its corresponding pixel square and then computing a histogram over all the squares in the particle cloud range.

With the true target shown near the centroid of the pdf, Figures 1 and 3 demonstrate convergent behavior for the PF method. However, Figures 2 and 4 indicate that ADI is experiencing significant estimation difficulties because the target is on the periphery of the pdf. Additional experiments with ADI at other tuning settings (e.g. filter process noise strength 10 and 100 times truth) show that position is usually estimated fairly well but velocity is not. RMS results will confirm these trends for both methods.

Figures 5 and 6 show RMS performance in position and velocity for the two methods. These figures were built from the 18 Monte Carlo studies, each study consuming 630 Kflops in its 50 runs. Figure 5 shows reasonably good estimation of position for both methods. Position RMS error falls as SNR increases for both methods, as expected, with PF being roughly twice as accurate as ADI.

Figure 6 shows improving velocity estimation for PF as SNR rises. For ADI, however, the RMS curve indicates poor estimation performance. Other tests were run to assess this situation, and they all confirmed that ADI was not estimating velocity. The resolution-induced limit on estimation accuracy for ADI is approximately 0.06 m/sec (2 m/sec spanned by 10 square grid cells) so the “flat” performance at roughly 0.45 m/sec is not resolution induced. We are not certain why ADI does not estimate velocity but we suspect a problem with its code. Unfortunately, this issue was discovered rather recently and we failed to find a correction before this paper was due.

Over the 50 simulation runs in each of the nine studies, PF was robust to both noise and target maneuvers, and on average lost lock just 4 times (4/54 ≈ 7%) per study when SNR was 12dB or less, and under once (<1%) per study when SNR was above 12dB. ADI however lost lock on average 32 times (39%) per study at 4dB, 17 times (25%) at 8dB, and less than 4 times (0%-7%) for 10dB and above. For special studies with ADI at -3, 0 and 2 dB, the average number of lost lock events increased to well over 50%. Although 50-run studies of ADI were possible at these low SNRs, the computational costs for discarding so many runs to get 50 “good” ones were prohibitive. That is the main reason that lower SNRs were not examined.

4 Summary

This paper introduces a challenge problem in nonlinear filtering that is made available to the research community in the form of MATLAB code posted at a public web site. This problem consists of modules for scenario generation and performance evaluation, as well as modules offering baseline solutions for two methods, a particle filter and an alternating direction implicit version of a finite difference scheme. These baseline solutions demonstrate in concrete terms how nonlinear methods can be applied to image intensity data. We are hopeful that their availability will encourage researchers with innovative ideas to apply their methods to the same problem data that was used here.

The authors are extending this work to add a solution method based on multi-scan multiple-hypothesis methods. Early experiments with this method suggest that it may produce results competitive with those obtained above for simple problem scenarios.

This paper also provides technical descriptions of the two solution techniques, and illustrates results from each. This effort has shown that both particle and finite difference methods can solve the dim target tracking problem, and that particle methods produce greater accuracy for equal computational resources.

References

[1] A. Doucet, “On sequential simulation-based methods for Bayesian filtering”, Signal Processing Group, Department of Engineering, 1998, No. CUED/F-INFENG/TR.310, University of Cambridge.

[2] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, “A novel approach to nonlinear/non-Gaussian Bayesian state estimation”, *IEEE Proceedings on Radar and Signal Processing*, vol. 140, 1993, pp. 107-113.

[3] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[4] K. Kastella and A. Zatazelo, “Nonlinear filtering for low elevation targets in multipath,” *Proceedings of SPIE, Signal and Data Processing of Small Targets 1998,* 14-16 April, 1998, Orlando, Florida, Vol. 3373, pp. 452-459.

[5] K. Kastella, “Finite Difference Methods for Tracking and Automatic Target Recognition”, *Multitarget Multisensor Tracking: Advance Applications* Vol. 3, Y. Bar-Sholom, D. Blair (Eds.), Artech, 2000.

[6] S. Lototsky, R. Mikuleviclus, B. L Rozovskii, "Nonlinear filtering revisited: a spectral approach", *SIAM J. Contr. Optim.*, Vol. 35, No.2, March 1997.

[7] D. B. Rubin, *Using the SIR Algorithm to Simulate Posterior Distributions*, Bayesian Statistics 3, Oxford University Press, J. M. Bernardo, M. H. DeGroot, D. V. Lindley and A. F. M. Smith, eds., 1988, pp. 395-402.

[8] H.W. Sorenson, “On the development of practical nonlinear filters,” *Information Sciences*, Vol. 7, 1974, pp. 253-270.

[9] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Chapman & Hall, New York, 1989.

[10] N. N. Yanenko, *The Method of Fractional Steps*, Springer-Verlag, 1971.

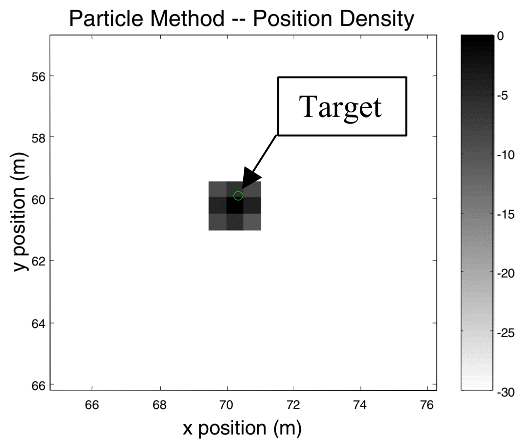

Figure 1 –Position density snapshot, PF

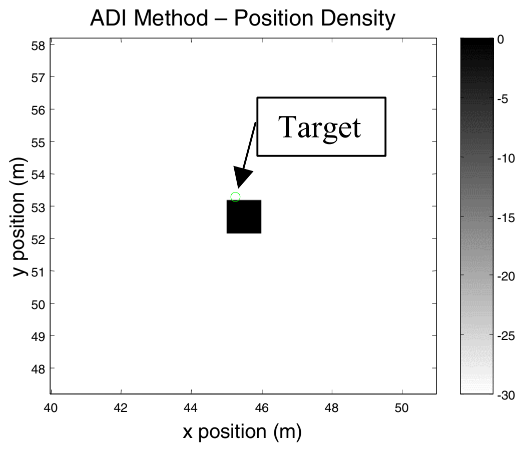

Figure 2 – Position density snapshot, ADI

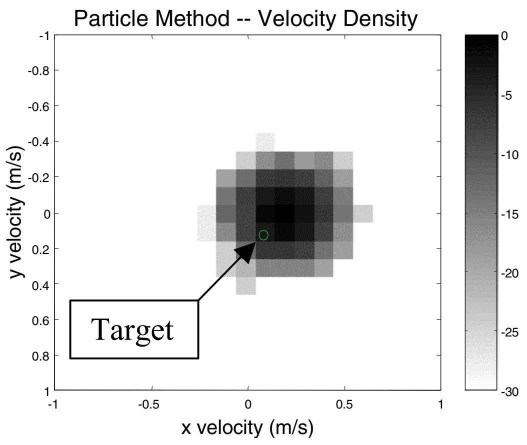

Figure 3 – Velocity density snapshot, PF

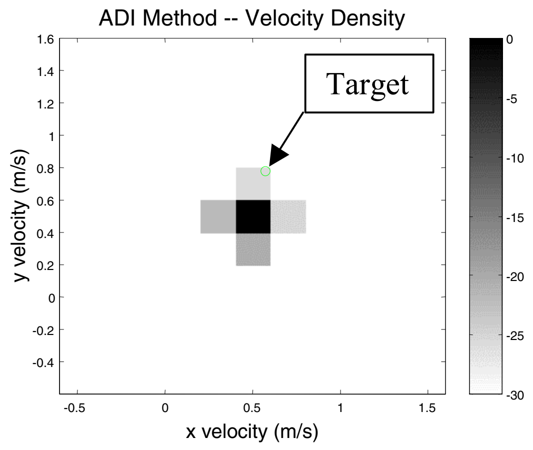

Figure 4 – Velocity density snapshot, ADI

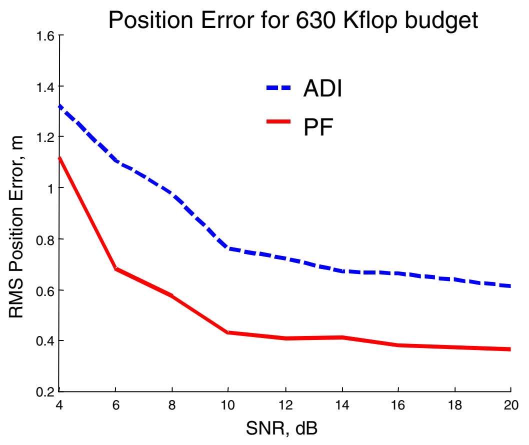

Figure 5 – RMS position error

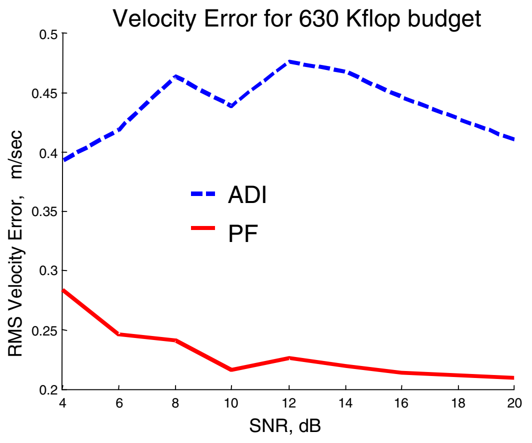

Figure 6 – RMS velocity error