International Telecommunications Symposium – ITS2002, Natal, Brazil

# An Accelerated Constant Modulus Algorithm

Magno T. M. da Silva, Max Gerken, and Maria D. Miranda*
Universidade de São Paulo, *Universidade Presbiteriana Mackenzie, São Paulo SP, Brazil

*Abstract*—***We present a novel adaptive algorithm for blind equalization. It is based on a tuner used in adaptive control that sets the second derivative of the parameter estimates and minimizes the cost function introduced by Godard. Based on simulation results we present a comparison with the Constant Modulus and the Shalvi-Weinstein algorithms. Both the convergence speed and the computational complexity of the proposed algorithm lie between those of the Constant Modulus and the Shalvi-Weinstein algorithms, thus presenting a more favorable compromise between convergence speed and computational complexity. Some preliminary results also suggest that the proposed algorithm presents a more robust behavior with respect to convergence to global minima.***

## I. Introduction

A fast, discrete-time adaptive filtering algorithm was introduced in a previous paper [1] and further analyzed in [2]. It was derived from a continuous-time tuner used in adaptive control, which sets the second derivative (”acceleration”) of the coefficient estimates [3]. For this reason the discrete-time algorithm was named the Accelerating Adaptive Filtering (AAF) algorithm. Results presented in [3] indicate that, at the cost of a moderate increase in complexity, this tuner is able to achieve a compromise between convergence speed and steady state coefficient error superior to that of the tuner that adjusts the first derivative (“velocity”) of the coefficient estimates.

In [1] it was shown that, for colored input signals, the AAF algorithm presents a more favorable compromise between convergence speed and steady-state estimation error than the LMS or NLMS algorithms, a property obtained at the cost of a moderate increase in computational complexity. A deterministic convergence proof using Lyapunov's method was also presented.

A simplified version of the AAF algorithm and the results of first and second order moment analyses of its filter coefficient errors were presented in [2]. Expressions that characterize the transient behavior of the mean squared error and the misadjustment allowed comparison with the LMS algorithm, showing the advantages of using the AAF algorithm. For similar convergence speeds, the AAF algorithm presents a lower misadjustment than the LMS algorithm. As a result of performance comparison with the LMS algorithm, some practical hints for choosing the parameters of the AAF algorithm were presented. Finally, it was also shown that the AAF algorithm can be interpreted as a quasi-Newton method.

The above results motivated the development of an adaptive algorithm for blind equalization based on the same principles. Since the the AAF algorithm compares favorably with the LMS algorithm, one might expect that an algorithm for blind equalization based on the accelerating tuner would achieve a better performance than the Constant Modulus (CM) algorithm.

Magno T. M. da Silva and Max Gerken are with the Department of Telecommunications and Control Engineering at Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brazil. Maria D. Miranda is with the Universidade Presbiteriana Mackenzie, São Paulo, SP, Brazil. E-mails: {magno,mgk}@lcs.poli.usp.br, *mdm@mackenzie.com.br. This work was supported by FAPESP - São Paulo State Research Council (00/12350-6) and CNPq - Brasilian Research Council (300521/92-8).

In the sequel we initially present a summary of the accelerating tuner. Then we introduce a non-linearity and use Euler's reverse method to discretize the non-linear tuner obtaining an Accelerated Constant Modulus (ACM) algorithm. Next we present simulation results comparing the convergence behavior of the ACM, CM and Shalvi-Weinstein (SW) algorithms. We close the paper making some concluding remarks.

## II. The Accelerating Tuner

In adaptive control and recursive coefficient estimation one often needs to adjust recursively an estimate $\mathbf{p}(t)$ of a coefficient vector $\mathbf{p}_\text{o}$ using a measured signal

$$d(t) = \mathbf{x}^T(t)\mathbf{p}_\text{o} + \eta(t), \tag{1}$$

where $\mathbf{x}(t)$ is the input signal vector (regressor) and $\eta(t)$ is the measurement noise. The goal is to maintain both the estimation error

$$e(t) = \mathbf{x}^T(t)\mathbf{p}(t) - d(t), \tag{2}$$

and the coefficient error

$$\boldsymbol{\Delta}\mathbf{p}(t) = \mathbf{p}(t) - \mathbf{p}_\text{o} \tag{3}$$

as small as possible.

The most straightforward tuning method used in adaptive control sets the first derivative of the coefficient estimates proportional to the estimation error:

$$\begin{array}{rcl} \dot{\mathbf{p}}(t) & = & -\mathbf{M}\mathbf{x}^*(t)e(t) \\ e(t) & = & \mathbf{x}^T(t)\mathbf{p}(t) - d(t), \end{array} \tag{4}$$

where $\mathbf{M}$ is a positive-definite matrix of appropriate dimensions and $^*$ stands for complex conjugate.

It is worth noting that the popular LMS algorithm can be obtained by applying Euler's method to discretize the above “velocity” tuner. Indeed, by setting $\mathbf{M} = \mu_\text{o}\mathbf{I}$ and $\mu = \mu_\text{o}T$, where $T$ is the integration constant, expressions

$$\mathbf{p}[n+1] = \mathbf{p}[n] - \mu\mathbf{x}^*[n]e[n]$$

and

$$e[n] = \mathbf{x}^T[n]\mathbf{p}[n] - d[n]$$

easily follow from (4).

A tuner that adjusts the second derivative (“acceleration”) of the coefficient estimates was introduced in [3]. With $\mathbf{q}(t) = \dot{\boldsymbol{\Delta}}\mathbf{p}(t) = \dot{\mathbf{p}}(t)$ (see (3)), the accelerating tuner can be described as follows [3]:

$$\begin{array}{rcl} \dot{\mathbf{p}}(t) & = & \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) & = & -\mathbf{M}_1\mathbf{x}^*(t)e(t) + \\ & & -2\mathbf{M}_1\left(\mathbf{M}_2 + \mathbf{x}^*(t)\mathbf{x}^T(t)\mathbf{M}_1\mathbf{M}_3\right)\mathbf{q}(t) \\ e(t) & = & \mathbf{x}^T(t)\mathbf{p}(t) - d(t). \end{array} \begin{array}{r} (5) \\ \\ (6) \\ (7) \end{array}$$

Column vectors $\mathbf{x}(t)$, $\mathbf{p}(t)$ and $\mathbf{q}(t)$ have dimension $N$, and the $N \times N$ symmetric matrices $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$ are positive-definite.

If no measurement noise is present we may write $d(t) = \mathbf{x}^T(t)\mathbf{p}_o$ and $e(t) = \mathbf{x}^T(t)\mathbf{\Delta p}(t)$. In this case the dynamics of the accelerating tuner can be described using the coefficient error vector $\mathbf{\Delta p}(t)$ as follows:

$$\underbrace{\begin{bmatrix} \dot{\mathbf{\Delta p}}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix}}_{\dot{\mathbf{s}}(t)} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}}_{\mathbf{A}(t)} \underbrace{\begin{bmatrix} \mathbf{\Delta p}(t) \\ \mathbf{q}(t) \end{bmatrix}}_{\mathbf{s}(t)}, \tag{8}$$

with

$$\begin{aligned} &\mathbf{A}_{21} = -\mathbf{M}_1\mathbf{x}^*(t)\mathbf{x}^T(t) \\ &\mathbf{A}_{22} = -2\mathbf{M}_1\left(\mathbf{M}_2 + \mathbf{x}^*(t)\mathbf{x}^T(t)\mathbf{M}_1\mathbf{M}_3\right). \end{aligned}$$

Sufficient conditions to guarantee stability of this system are

$$4\mathbf{M}_1\mathbf{M}_3\mathbf{M}_1\mathbf{M}_2 > \mathbf{I} \tag{9}$$

and

$$\mathbf{M}_2\mathbf{M}_1\mathbf{M}_3 + \mathbf{M}_1\mathbf{M}_3\mathbf{M}_2 > \mathbf{M}_1^{-1}/2. \tag{10}$$

The Lyapunov function used to establish this is defined by the matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{M}_2 & \mathbf{M}_1^{-1}/2 \\ \mathbf{M}_1^{-1}/2 & \mathbf{M}_3, \end{bmatrix} \tag{11}$$

which has also been used to prove stability of the AAF algorithm [1].

The AAF algorithm was obtained [1] from (5) to (7) using Euler's reverse method. The reasons for this choice are quite simple: the direct Euler method results in a low complexity algorithm that may be unstable; other numerical integration methods like the trapezoidal rule result in algorithms with higher computational complexity. For these reasons we use Euler's reverse method in the next section to derive the ACM algorithm.

## III. The Accelerated Constant Modulus (ACM) algorithm

We begin substituting Equation (7) by a non-linear version

$$e(t) = \varphi(\mathbf{x}^T(t)\mathbf{p}(t)) \tag{12}$$

with $\varphi(.)$ satisfying

$$\frac{\partial \varphi(\mathbf{x}^T\mathbf{p})}{\partial \mathbf{p}} = g(\mathbf{x}^T\mathbf{p})\mathbf{x}, \tag{13}$$

where $\frac{\partial}{\partial \mathbf{p}}$ stands for derivation with respect to the complex vector $\mathbf{p}$, and $g(.)$ is a scalar function. Thus, a linear approximation of $\varphi(.)$ may be written as

$$\begin{aligned} \varphi(\mathbf{x}^T\mathbf{p}) &\approx \varphi(\mathbf{x}^T\mathbf{p}_v) + \left[ \left. \frac{\partial \varphi(\mathbf{x}^T\mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_v} \right]^T (\mathbf{p} - \mathbf{p}_v) \\ &\approx \varphi(\mathbf{x}^T\mathbf{p}_v) + g(\mathbf{x}^T\mathbf{p}_v)\mathbf{x}^T(\mathbf{p} - \mathbf{p}_v). \end{aligned} \tag{14}$$

Applying Euler's reverse rule to expressions (5), (6) and (12), considering an integration step $\alpha$, we obtain

$$\begin{aligned} \mathbf{p}[n] &= \mathbf{p}[n-1] + \alpha\mathbf{q}[n] && (15) \\ \mathbf{q}[n] &= \mathbf{q}[n-1] - \alpha\mathbf{M_1}\{\mathbf{x}^*[n]e[n] + \\ &\quad +2\left(\mathbf{M_2} + \mathbf{x}^*[n]\mathbf{x}^T[n]\mathbf{M_1}\mathbf{M_3}\right)\mathbf{q}[n]\} && (16) \\ e[n] &= \varphi(\mathbf{x}^T[n]\mathbf{p}[n]). && (17) \end{aligned}$$

These equations do not make the update of $\mathbf{p}[n]$ and $\mathbf{q}[n]$ possible. To overcome this obstacle we introduce an *a priori* error

$$e_a[n] = \varphi(\mathbf{x}^T[n]\mathbf{p}[n-1]) \tag{18}$$

and note from Equation (15) that

$$\mathbf{x}^T[n]\mathbf{p}[n] = \mathbf{x}^T[n]\mathbf{p}[n-1] + \alpha\mathbf{x}^T[n]\mathbf{q}[n]. \tag{19}$$

By means of (19) and (14) the *a posteriori* error $e[n]$ can be computed from the *a priori* error $e_a[n]$ as follows:

$$\begin{aligned} e[n] &= \varphi\Big(\underbrace{\mathbf{x}^T[n]\mathbf{p}[n-1]}_{y[n]} + \alpha\mathbf{x}^T[n]\mathbf{q}[n]\Big) && (20) \\ &\approx \underbrace{\varphi(y[n])}_{e_a[n]} + \alpha g(y[n])\mathbf{x}^T[n]\mathbf{q}[n]. && (21) \end{aligned}$$

Using (21) and (16) we obtain an update expression:

$$\begin{aligned} \mathbf{q}[n] &= \mathbf{G}^{-1}\left(\mathbf{q}[n-1] - \alpha\mathbf{M_1}\mathbf{x}^*[n]e_a[n]\right) \\ \mathbf{G} &= \mathbf{I} + \alpha^2 g(y[n])\mathbf{M_1}\mathbf{x}^*[n]\mathbf{x}^T[n] + \\ &\quad +2\alpha\mathbf{M_1}\left(\mathbf{M_2} + \mathbf{x}^*[n]\mathbf{x}^T[n]\mathbf{M_1}\mathbf{M_3}\right). \end{aligned}$$

It can be shown that the inverse of matrix $\mathbf{G}$ is given by

$$\mathbf{G}^{-1} = \mathbf{A}\left\{\mathbf{I} - \frac{\mathbf{M_1}\mathbf{x}^*[n]\mathbf{x}^T[n]\mathbf{B}[n]\mathbf{M_1^{-1}}}{1 + \mathbf{x}^T[n]\mathbf{B}[n]\mathbf{x}^*[n]}\right\} \tag{22}$$

with

$$\mathbf{A} = \left(\mathbf{I} + 2\alpha\mathbf{M_1}\mathbf{M_2}\right)^{-1} \tag{23}$$

and

$$\mathbf{B}[n] = \alpha\left\{\alpha g(y[n])\mathbf{I} + 2\mathbf{M_1}\mathbf{M_3}\right\}\mathbf{A}\mathbf{M_1}. \tag{24}$$

Substituting this result into (22) we obtain a first version of the ACM algorithm:

$$\begin{aligned} \mathbf{A} &= \left(\mathbf{I} + 2\alpha\mathbf{M_1}\mathbf{M_2}\right)^{-1} \\ y[n] &= \mathbf{x}^T[n]\mathbf{p}[n-1] \\ e_a[n] &= \varphi(y[n]) \\ \mathbf{B}[n] &= \alpha\left(\alpha g(y[n])\mathbf{I} + 2\mathbf{M_1}\mathbf{M_3}\right)\mathbf{A}\mathbf{M_1} \\ \mathbf{C}[n] &= \frac{\alpha e_a[n] + \mathbf{x}^T[n]\mathbf{B}[n]\mathbf{M_1^{-1}}\mathbf{q}[n-1]}{1 + \mathbf{x}^T[n]\mathbf{B}[n]\mathbf{x}^*[n]}\mathbf{M_1} \\ \mathbf{q}[n] &= \mathbf{A}\left(\mathbf{q}[n-1] - \mathbf{C}[n]\mathbf{x}^*[n]\right) \\ \mathbf{p}[n] &= \mathbf{p}[n-1] + \alpha\mathbf{q}[n]. \end{aligned}$$

The computational complexity of this version is proportional to $N^2$. However, if matrices $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ are diagonal the computational complexity becomes proportional to $N$. In particular if we take $\mathbf{M}_i = m_i\mathbf{I}$, $i = 1, 2, 3$, with $m_i$ being positive constants, the following low complexity version of the ACM algorithm results:

$$a = 1 + 2\alpha m_1 m_2$$

$$y[n] = \mathbf{x}^T[n]\mathbf{p}[n-1]$$

$$e_a[n] = \varphi(y[n])$$

$$b[n] = 2\alpha m_1^2 m_3 + \alpha^2 m_1 g(y[n])$$

$$c[n] = \frac{b[n]\mathbf{x}^T[n]\mathbf{q}[n-1] + \alpha m_1 a e_a[n]}{a + b[n]\left\|\mathbf{x}[n]\right\|^2}$$

$$\mathbf{q}[n] = \frac{1}{a}\left(\mathbf{q}[n-1] - c[n]\mathbf{x}^*[n]\right)$$

$$\mathbf{p}[n] = \mathbf{p}[n-1] + \alpha\mathbf{q}[n],$$

where $a$, $1/a$, $\alpha m_1 a$, $2\alpha m_1^2 m_3$ and $\alpha^2 m_1$ need to be computed only once. For real signals this version requires $6N + 4$ multiplications, 1 division, $5N$ additions and two non-linearity computations, a computational complexity that lies between those of the CM and SW algorithms, the latter being proportional to $N^2$.

To complete the derivation of the ACM algorithm functions $\varphi(.)$ and $g(.)$ must be chosen. To this end we consider the instantaneous versions of the cost functions introduced by Godard [5], [4]:

$$\Psi_q(y) = \frac{1}{2q}\left(|y|^q - R_q\right)^2, \qquad (25)$$

where $y = \mathbf{p}^T\mathbf{x}$ and $q$ is a positive integer.

Introducing

$$f_q(y) = \left(|y|^q - R_q\right)|y|^{q-2} \qquad (26)$$

it results [5], [4] that

$$\nabla_{\mathbf{p}}\Psi_q(y) = \underbrace{f_q(y)y}_{\varphi_q(y)}\mathbf{x}^* \qquad (27)$$

and

$$\frac{\partial\varphi_q(y)}{\partial\mathbf{p}} = \underbrace{q\left(f_q(y) + |y|^{2(q-1)}\right)}_{g_q(y)}\mathbf{x}, \qquad (28)$$

where $\nabla_{\mathbf{p}}$ stands for the gradient with respect to $\mathbf{p}$. For $q = 2$ we obtain the non-linearities that are used in the ACM algorithm:

$$\varphi(y) \triangleq \varphi_2(y) = \left(|y|^2 - R_2\right)y \qquad (29)$$

$$g(y) \triangleq g_2(y) = 2\left(2|y|^2 - R_2\right). \qquad (30)$$

We observe that the more general nonlinearities $\varphi_q(.)$ and $g_q(.)$ can also be used.

## IV. Simulations Results

In this Section we compare the convergence behaviour of the ACM algorithm with the well-known CM and SW algorithms. The first problem we are faced with when using the ACM algorithm is how to choose the parameters $\alpha$, $m_1$, $m_2$ and $m_3$ to obtain an adequate performance. Motivated by the fact that the AAF algorithm is stable and reaches its fastest convergence at the upper bound of (9) [2] we introduce a parameter $\gamma$ to set $m_1m_2 = 1/(2\gamma)$ and $m_1m_3 = \gamma/2$. This guarantees $4m_1^2m_2m_3 = 1$. Consequently, we need to set three positive parameters, $\alpha$, $\gamma$ and $m_1$, to adjust the performance of the ACM algorithm. An investigation of the stability domain of these parameters still must be performed. However, our experience indicates that this is not a critical matter. As long as they are positive, simulation results suggest that $\alpha$ should be chosen lower than one and $\gamma$ and $m_1$ may be chosen almost without constraints.

Table I shows the channel models used in the simulations. For channel $\mathsf{H}_7$ coefficients $h_0(n)$, $h_1(n)$ and $h_2(n)$ are generated by passing a Gaussian white noise through a second order Butterworth filter designed to simulate a fade rate of $0.1$ Hz [6].

TABLE I
COMMUNICATION CHANNEL MODELS USED IN THE SIMULATIONS.

| |
|---|
| $\mathsf{H}_1(z) = (1 + 2z^{-1} + z^{-2})/\sqrt{6}$<br>Zeros: $\{-1; -1\}$ |
| $\mathsf{H}_2(z) = (1 + 2z^{-1} + 0.96z^{-2})/\sqrt{5.9216}$<br>Zeros: $\{-1.2; -0.8\}$ |
| $\mathsf{H}_3(z) = (1 + 1.6z^{-1} + z^{-2})/\sqrt{4.56}$<br>Zeros: $\{e^{\pm j\theta}, \theta = \pi/4.882\}$ |
| $\mathsf{H}_4(z) = (1 + 1.9z^{-1} + 1.2z^{-2})/\sqrt{6.05}$<br>Zeros: $\{-0.9545 \pm j0.5454\}$ |
| $\mathsf{H}_5(z) = 1/(1 + 0.6z^{-1})$<br>Pole: $\{-0.6\}$ |
| $\mathsf{H}_6(z) = (1.11 + z^{-1} + 0.9z^{-2})/\sqrt{3.0421}$<br>Zeros: $\{-0.4505 \pm j0.7797\}$ |
| $\mathsf{H}_7$ (Time-variant [6])<br>$x(n) = h_0(n)a(n) + h_1(n)a(n-1) +$<br>$+h_2(n)a(n-2)$,<br>$a(n)$: channel input symbols<br>$x(n)$: noiseless channel output. |

Fig. 1 shows the convergence behavior by means of the Intersymbol Interference (ISI) curves of the ACM, CM and SW algorithms for 4-QAM modulation and the noiseless channels $\mathsf{H}_1$, $\mathsf{H}_2$, $\mathsf{H}_3$ and $\mathsf{H}_4$. For the channels $\mathsf{H}_1$ and $\mathsf{H}_2$, which have real zeros on the unit circle and close to it respectively, the performance of the ACM algorithm lies between the performances of the CM and SW algorithms. For the channels $\mathsf{H}_3$ and $\mathsf{H}_4$ the ACM algorithm has a behavior very close to the SW algorithm.

The contour plot of $\mathrm{E}\{\Psi_2(y)\}$ (Godard cost function) for channel $\mathsf{H}_5$ is shown in Fig. 2. This figure shows two local and two global minima of the cost function and different trajectories of the CM, SW and ACM algorithms. If initialized at $\mathbf{p}[0] = [0\ 1]^T$ the algorithms present similar behaviors.

For $\mathbf{p}[0] = [-0.4\ \ 0.05]^T$, which is close to one of the local minima, the ACM algorithm crosses the local minimum and reaches one of the global minima while the CM and SW algorithms stagnate at the local minimum. If the pole of $\text{H}_5(z)$ is changed from $-0.6$ to $-0.2$ the local minima become more pronounced. In this case, the ACM algorithm shows the same behavior of the CM and SW algorithms, being attracted to the local minima. This behavior suggests that, to some extent, the ACM algorithm has the ability to escape from soft local minima.

For $\mathbf{p}[0] = [0.2\ \ -0.4]^T$ the ACM algorithm reaches one of the global minima crossing the attraction domain of a local minimum while the CM and SW algorithms converge to the other global minimum. The corresponding ISI curves are shown in Fig. 3. In this case, the ACM algorithm follows the "rocky trail" while the others converge straight to a valley. While successfully avoiding a local minimum, the ACM algorithm shows a slower convergence behavior. If the pole of $\text{H}_5(z)$ is again changed from $-0.6$ to $-0.2$ the three algorithms present the same behavior, converging to the same global minimum.

Fig. 4 shows ISI curves of the ACM algorithm for different initializations. For each initialization, the ISI of the ACM algorithm almost reaches the ISI value of a Wiener solution with the same delay of combined channel equalizer response. Besides being a very regular behavior, it confirms that the ACM algorithm doesn't avoid deep local minima.

Fig. 5 shows the equalizer's output for the CM, SW and ACM algorithms. In this case a time-variant channel, $\text{H}_7$, is used and the signal-to-noise ratio is set to 20 dB. The absolute values of the roots of $h_0(n)x^2 + h_1(n)x + h_2(n)$ are shown in Fig. 5-d so that burst of errors can be associated with rapid changes of these roots. Particularly, the bursts near iterations $5000$ and $27000$ are due to strong spectral nulls (absolute value equal one is indicated by a straight line). The ACM algorithm shows a faster recuperation than the CM algorithm, presenting a behavior very close to the SW algorithm. The corresponding ISI curves are presented in Fig. 6 confirming the similar behaviors of the ACM and SW algorithms.

## V. Conclusions

We have proposed an algorithm for blind equalization that presents a more favorable compromise between convergence speed and computational complexity than the CM and SW algorithms. Through simulations we showed that the ACM algorithm outperforms the CM algorithm and has a behavior close to the SW algorithm in many situations. For example, to some extent it was able to avoid local minima and for a time-variant channel it showed faster recovery times.

## Acknowledgments

The authors thank Dr. Vitor H. Nascimento for his suggestions.

## References

[1] M. Gerken, F.M. Pait, P.E. Jojoa Gomez: An Adaptive Algorithm with Parameter Acceleration, *Proceedings of ICASSP'2000*, Vol I, April 2000, pp. 17-20.

[2] P.E. Jojoa Gomez, M. Gerken, F.M. Pait: An Adaptive Algorithm with Parameter Acceleration, *Proceedings of IFAC ALCOSP'2001*, August 2001, Cernobio-Como, Italy, pp. 331-335.

[3] F.M. Pait: A Tuner that Accelerates Parameters, *Systems & Control Letters* 35 (1998) 65-68.

[4] S. Haykin: *Adaptive Filter Theory*, 3. Ed., Prentice Hall, 1996.

[5] Z. Ding, Ye Li: *Blind Equalization and Identification*, Marcel Dekker, 2001.

[6] T. Shimamura, and C. F. N. Cowan, "Equalisation of time variant multipath channels using amplitude banded techniques," in *Proc. IEEE ICASSP'1997*, pp. 2497-2500.

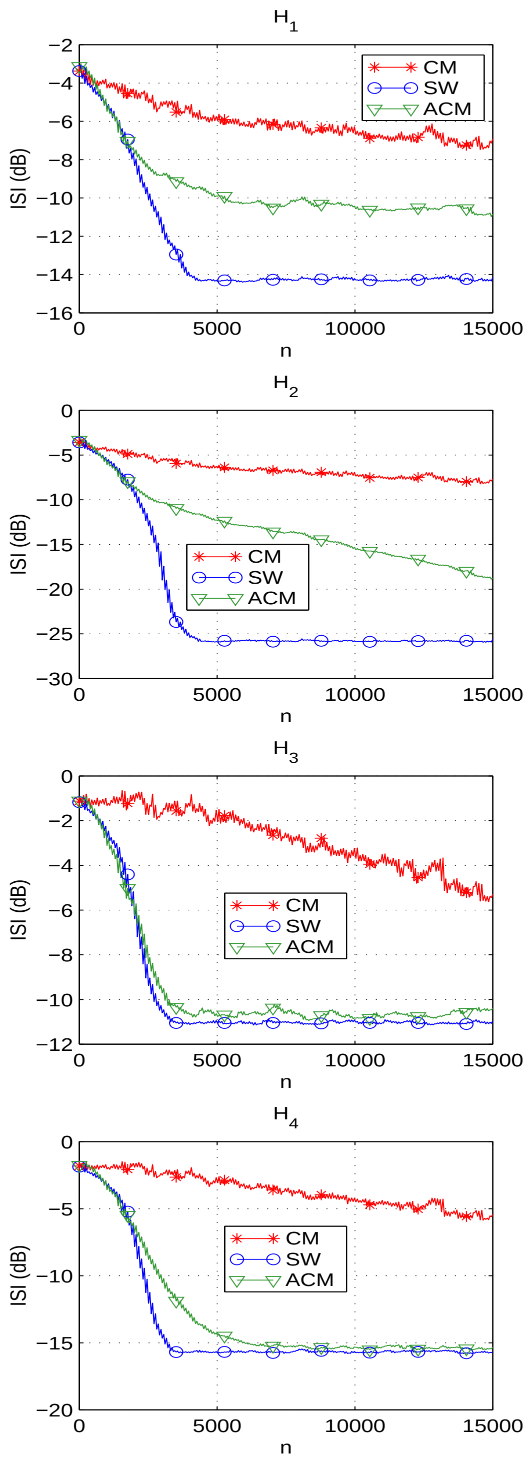


Fig. 1. ISI for the CM ($\mu = 0.0012$), SW ($\lambda = 0.9975$) and ACM ($\alpha = 0.0975$, $m_1 = 0.9975$, $\gamma = 75.19$) algorithms. For 4-QAM, $N = 27$ and channels $\text{H}_1$, $\text{H}_2$, $\text{H}_3$ and $\text{H}_4$.

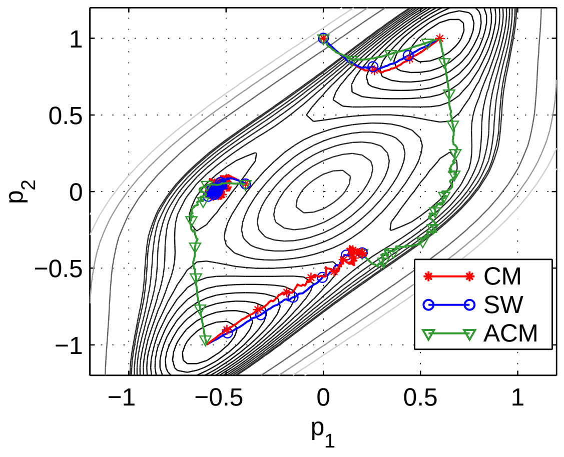


Fig. 2. Contour plot of the Godard cost function (E$\{\Psi_2(y)\}$) for channel H$_5$ with trajectories of the CM ($\mu = 0.0025$), SW ($\lambda = 0.995$) and ACM ($\alpha = 0.25$, $m_1 = 0.1592$, $\gamma = 106$) algorithms. Initialization at points $\mathbf{p}[0] = [0\ 1]^T$, $\mathbf{p}[0] = [-0.4\ 0.05]^T$ and $\mathbf{p}[0] = [0.2\ \ -0.4]^T$. For 2-PAM.

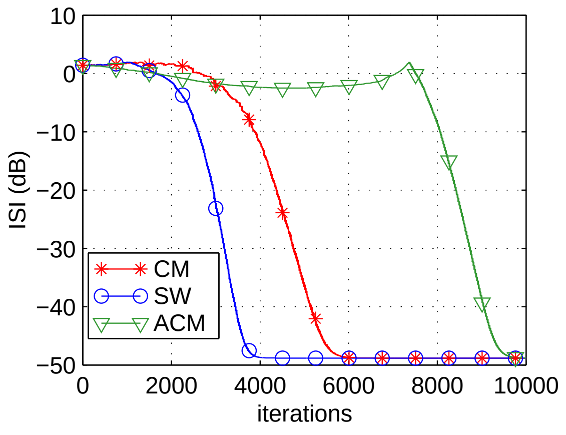


Fig. 3. ISI of the CM ($\mu = 0.0025$), SW ($\lambda = 0.995$) and ACM ($\alpha = 0.25$, $m_1 = 0.1592$, $\gamma = 106$) algorithms initialized at $\mathbf{p}[0] = [0.2\ \ -0.4]^T$. For channel H$_5$, $N = 2$ and 2-PAM.

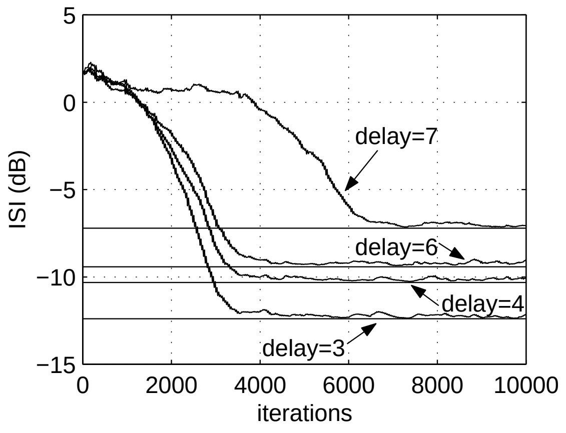


Fig. 4. ISI of the ACM ($\alpha = 0.25$, $m_1 = 0.1592$, $\gamma = 106$) algorithm with different initializations. Straight lines show ISI values of Wiener solutions for different delays. For $N = 11$, 2-PAM and channel H$_6$.

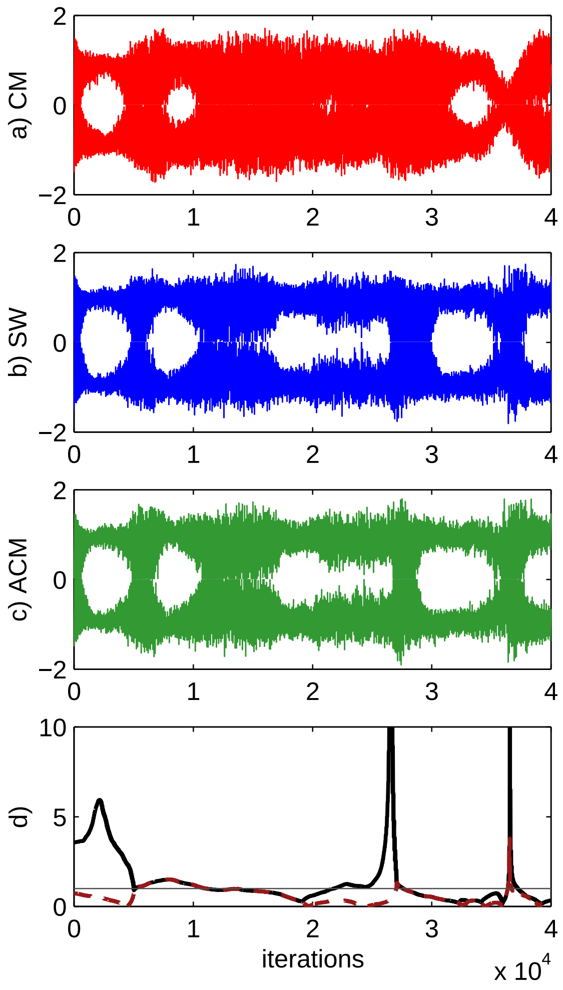


Fig. 5. Equalizer output of a) CM ($\mu = 0.0025$), b) SW ($\lambda = 0.995$), c) ACM ($\alpha = 0.25$, $m_1 = 0.1592$, $\gamma = 106$) algorithms. For channel H$_7$, $N = 11$, SNR=20 dB and 2-PAM. d) Absolute root values of the polynomial $h_0(n)x^2 + h_1(n)x + h_2(n)$.

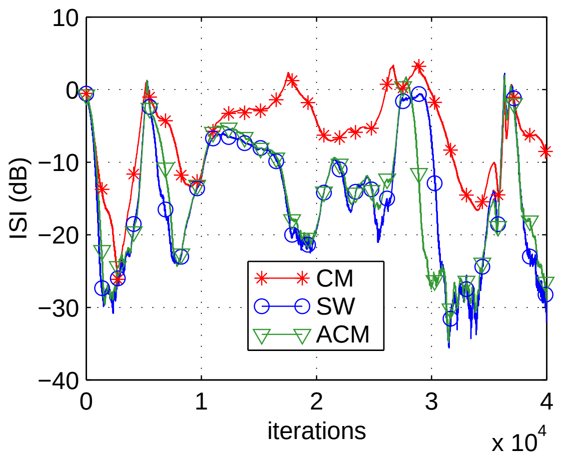


Fig. 6. ISI curves for the CM ($\mu = 0.0025$), SW ($\lambda = 0.995$) and ACM ($\alpha = 0.25$, $m_1 = 0.1592$, $\gamma = 106$) algorithms. For 2-PAM, $N = 11$, SNR=20 dB and channel H$_7$.

summation limits accordingly in (12) for centroid locations near the borders, see [1] for details.
Clutter Model We capture the 2D spatial correlation of the background clutter using a noncausal, spatially homogeneous Gauss-Markov random field (GMrf) model [10]. The clutter returns at frame $n$, $V_n(i,\,j)$, $1 \leq i \leq L$, $1 \leq j \leq M$, are described by the 2D finite difference equation

$$\begin{aligned} V_n(i,\,j) &= \beta_v^c\left[V_n(i-1,\,j)+V_n(i+1,\,j)\right] \\ &+ \quad \beta_h^c\left[V_n(i,\,j-1)+V_n(i,\,j+1)\right]+U_n(i,\,j) \end{aligned} \quad (13)$$

where $E\left[V_n(i,\,j)\,U_n(p,\,r)\right]=\sigma_c^2\,\delta_{i-p,\,j-r}$. The assumption of zero-mean clutter implies a pre-processing of the data that subtracts the mean of the background.

### 3.1. Likelihood Function

Let $\mathbf{y}_n$ be a 1D long-vector representation of the frame $\mathbf{Y}_n$ obtained by either row or columnwise scanning. Assuming a 2D GMrf background as in (13) and deterministic signature parameters $\{a_{k,l}\}$, we use the results in [1] to write the likelihood function of the observed $n$th frame as

$$p(\mathbf{y}_n \mid \mathbf{x}_{n,1},\,\mathbf{x}_{n,2}) \propto \exp\left[\frac{2\lambda(\mathbf{x}_{n,1},\,\mathbf{x}_{n,2})-\rho}{2\sigma_c^2}\right]\ . \quad (14)$$

where $\rho$ is a target energy term that is constant away from the image borders, see [1] for details. The function $\lambda$ in (14) is in turn given by [1]

$$\lambda(\mathbf{x}_{n,1},\,\mathbf{x}_{n,2}) = \sum_{k=-r_i}^{r_s}\sum_{l=-l_i}^{l_s} a_{k,l}\,\mu(x_n^*(1)+k,\,x_n^*(2)+l) \quad (15)$$

where $x_n^*(i)$, $i=1,2$, are obtained respectively from (10) and (11), and $\mu(p,r)$ is the output of the differential operator

$$\begin{aligned} \mu(p,r) &= Y_n(p,r)-\beta_h^c\left[Y_n(p,r-1)+Y_n(p,r+1)\right] \\ &- \quad \beta_v^c\left[Y_n(p-1,r)+Y_n(p+1,r)\right] \end{aligned} \quad (16)$$

with Dirichlet (identically zero) boundary conditions. Equation (15) is valid for $r_i+1 \leq x_n^*(1) \leq L-r_s$ and $l_i+1 \leq x_n^*(2) \leq M-l_s$. For centroid positions close to the image borders, the summation limits in (15) must be varied accordingly as explained in [1]. Intuitively, we can interpret the function $\lambda$ in (15) as the concatenation of two linear filtering operations: first, we pass the image through a noncausal differential filter to generate a residual error image and, then, we apply to this error image a 2D correlation filter that is matched to the (known) 2D target template.

## 4. PARTICLE FILTER TRACKER

The on-line Bayesian estimation problem can be summarized as follows: given a sequence of observed frames $\mathbf{Y}_1^n = \{\mathbf{y}_1,\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n\}$ and the probability density function (pdf) $p(\mathbf{x}_0)$ of the initial (continuous-valued) target state, $\mathbf{x}_0$, compute recursively the *posterior* pdf $p(\mathbf{x}_n \mid \mathbf{Y}_1^n)$, for $n \geq 1$, using the Markovian transition kernel, $p(\mathbf{x}_{n+1} \mid \mathbf{x}_n)$, and the likelihood function, $p(\mathbf{y}_n \mid \mathbf{x}_n)$. From the posterior pdf, we can then infer the value of the hidden (unobserved) state at instant $n$, $\mathbf{x}_n$, using some optimality criteria, e.g. minimum mean-square error (MMSE) or maximum a posteriori (MAP).

Particle filters [3] are a simulation approach to Bayesian estimation in which the posterior pdf is represented at each instant $n$ by a set of particles $\left\{\mathbf{x}_n^{(j)}\right\}$, $1 \leq j \leq N_p$, with associated weights $w_n^{(j)}$. The MMSE estimate of the hidden state $\mathbf{x}_n$ is then obtained simply as [3] a weighted average of the particles. Alternatively, the MAP estimate can be obtained [4] from the histogram of the particles.

Ideally, we would like the particles to be samples from the true posterior in which case all weights $w_n^{(j)}$ would be identical and equal to $1/Np$. In practice, however, either the posterior pdf is unavailable or difficult to sample from. An alternative approach known as *sequential importance sampling* (SIS) [4] is to sample $\mathbf{x}_n^{(j)}$ sequentially from an importance function $\pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)$ and update the weights using the recursion

$$w_n^{(j)} \propto w_{n-1}^{(j)}\frac{p(\mathbf{y}_n \mid \mathbf{x}_n^{(j)})\,p(\mathbf{x}_n^{(j)} \mid \mathbf{x}_{n-1}^{(j)})}{\pi(\mathbf{x}_n^{(j)} \mid (\mathbf{X}^{(j)})_0^{n-1},\,\mathbf{Y}_1^n)}, \qquad \sum_{j=1}^{N_p} w_n^{(j)} = 1\ . \quad (17)$$

In principle, the importance function $\pi$ can be arbitrary provided that it satisfies two restrictions [4]

1. $\pi(\mathbf{X}_0^n \mid \mathbf{Y}_1^n)$ must have the same support as $p(\mathbf{X}_0^n \mid \mathbf{Y}_1^n)$ and be strictly positive and integrable to 1 in that support.

2. $\pi(\mathbf{X}_0^n \mid \mathbf{Y}_1^n) = \pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)\pi(\mathbf{X}_0^{n-1} \mid \mathbf{Y}_1^{n-1})$.

### 4.1. Bootstrap Tracker

The first practical SIS filter was the bootstrap filter [5], where the Markovian transition kernel $p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$ is used as the importance function $\pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)$. The weight update equation reduces then to

$$w_n^{(j)} \propto w_{n-1}^{(j)}\,p(\mathbf{y}_n \mid \mathbf{x}_n^{(j)}) \qquad \sum_{j=1}^{N_p} w_n^{(j)} = 1\ . \quad (18)$$

The key innovation of the bootstrap filter [5] was to introduce an additional particle selection step to avoid the so-called *degeneracy phenomenon* [3], i.e., the tendency of the distribution of the particle weights to become skewed as the number of SIS iterations increases. The selection step proposed in [5] consisted simply of resampling $N_p$ times from the set $\left\{\mathbf{x}_n^{(j)}\right\}$ with replacement according to the particle weights so that low-weight particles are discarded whereas high-weight particles are multiplied. The new weights after the resampling step are all reset then to $1/N_p$.

Using the motion, target and clutter models from sections 2 and 3 and recalling the expressions for the Markovian transition kernel in subsection 2.1 and for the likelihood function in subsection 3.1, we present in Table 1 a bootstrap filter algorithm for target tracking in image sequences.

```
1. Initialization For $j = 1, \ldots, N_p$
   • Draw $\mathbf{x}_{0,1}^{(j)} \sim p(\mathbf{x}_{0,1})$, $\mathbf{x}_{0,2}^{(j)} \sim p(\mathbf{x}_{0,2})$,
     make $w_0^{(j)} = 1/N_p$ and set $n = 1$.
2. Importance Sampling Step For $j = 1, \ldots, N_p$
   • Draw $\tilde{\mathbf{x}}_{n,1}^{(j)} \sim N(\mathbf{F}\mathbf{x}_{n-1,1}^{(j)}, \mathbf{Q})$
     and $\tilde{\mathbf{x}}_{n,2}^{(j)} \sim N(\mathbf{F}\mathbf{x}_{n-1,2}^{(j)}, \mathbf{Q})$.
   • Compute the importance weights
     $\tilde{w}_n^{(j)} \propto w_{n-1}^{(j)}\ p(\mathbf{y}_n \mid \tilde{\mathbf{x}}_{n,1}^{(j)}, \tilde{\mathbf{x}}_{n,2}^{(j)}) \qquad \sum_{j=1}^{N_p} \tilde{w}_n^{(j)} = 1$
     using equations (14), (15), and (16).
3. Selection Step
   • Generate a new set of samples
     $\left\{\mathbf{x}_n^{(j)} = \left[(\mathbf{x}_{n,1}^{(j)})^T\ (\mathbf{x}_{n,2}^{(j)})^T\right]^T\right\} \qquad 1 \leq j \leq N_p$
     such that $P(\mathbf{x}_n^{(j)} = \tilde{\mathbf{x}}_n^{(k)}) = \tilde{w}_n^{(k)}$.
   • Make $w_n^{(j)} = 1/N_p$, $1 \leq j \leq N_p$.
   • While $n \leq N_{\text{max}}$, set $n = n + 1$
     and go back to step 2.
```

Table 1: Algorithm I: Bootstrap filter for target tracking in 2D cluttered image sequences.

Clutter Adaptation When the clutter model parameters $\beta_h^c$, $\beta_v^c$ and $\sigma_c^2$ are unknown, they must be estimated from the observed data. For computational simplicity, we use in this paper a suboptimal approach to clutter adaptation where we estimate the GMrf clutter parameters directly from each available sensor frame $\mathbf{Y}_n$ using a variation of *approximate maximum likelihood* (AML) parameter estimation algorithm introduced in [10].

## 5. LINEARIZED KALMAN-BUCY TRACKER

We compare the proposed nonlinear bootstrap tracker to the association of a single frame maximum likelihood (ML) position estimator and a linear Kalman-Bucy filter. The preliminary ML estimates of the 2D pixel location of the target centroid are given by

$$\widehat{\mathbf{x}_n^*} = \arg\max_{\mathbf{x}_n^*} p(\mathbf{y}_n \mid \mathbf{x}_n) \ . \tag{19}$$

The pixel estimates of the centroid location are converted to continuous space and treated as decoupled noisy measurements of the true centroid positions in each dimension. A conventional Kalman-Bucy filter with knowledge of the state dynamics is then used to refine the preliminary ML estimates. Table 2 summarizes the algorithm. In Table 2, $\sigma_{v,i}^2$ denotes the variance of the ML centroid position estimation error in the dimension $i$ and $\mathbf{C}$ is the row vector $\mathbf{C} = [1\ 0]$. We denote the mean and covariance matrix of $\mathbf{x}_{0,i}$, $i = 1, 2$, respectively by $\overline{\mathbf{x}}_{0,i}$ and $\Sigma_{0,i}$. The symbols $\widehat{\mathbf{x}}_{n|n,i}$ and $\Sigma_{n|n,i}$ denote respectively the Kalman filter estimate of the target state $\mathbf{x}_{n,i}$ at instant $n$ in dimension $i$ and its associated error covariance matrix.

```
Initialization For $i = 1, 2$
• $\widehat{\mathbf{x}}_{0|0,i} = \overline{\mathbf{x}}_{0,i} \qquad \Sigma_{0|0,i} = \Sigma_{0,i}$
For $n = 1$ to $N_{\text{max}}$
  ML Step
    Compute $p(\mathbf{y}_n \mid x_{n,1}(1), x_{n,2}(1))$
    using equations (14), (15) and (16), make
    $(\widehat{z}_1, \widehat{z}_2) = \arg\max_{\mathcal{L}} p(\mathbf{y}_n \mid x_{n,1}(1), x_{n,2}(1))$
    and $\tilde{x}_{n,i} = \text{round}(\widehat{z}_i\, \Delta_i), \qquad i = 1, 2$
  Kalman Filter Step For i=1,2
    • $\widehat{\mathbf{x}}_{n|n-1,i} = \mathbf{F}\widehat{\mathbf{x}}_{n-1|n-1,i}$
    • $\Sigma_{n|n-1,i} = \mathbf{F}\Sigma_{n-1|n-1,i}\mathbf{F}^T + \mathbf{Q}$
    • $\mathbf{K}_{n,i} = \Sigma_{n|n-1,i}\mathbf{C}^T \underbrace{(\mathbf{C}\Sigma_{n|n-1,i}\mathbf{C}^T + \sigma_{v,i}^2)}_{\mathbf{S}_{n|n-1}}{}^{-1}$
    • $\widehat{\mathbf{x}}_{n|n,i} = \widehat{\mathbf{x}}_{n|n-1,i} + \mathbf{K}_{n,i}(\tilde{x}_{n,i} - \mathbf{C}\widehat{\mathbf{x}}_{n|n-1,i})$
    • $\Sigma_{n|n,i} = \Sigma_{n|n-1,i} - \mathbf{K}_{n,i}\mathbf{S}_{n|n-1,i}\mathbf{K}_{n,i}^T$
End of for loop
```

Table 2: Algorithm II: Association ML Estimator + Linear Kalman-Bucy Filter (for comparison purposes only).

## 6. PERFORMANCE RESULTS

We compare next the tracking performances of the bootstrap tracker and the KBf tracker using a simulated image sequence that was generated from real infrared

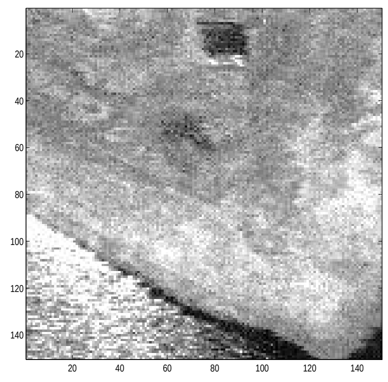


Figure 1: Real IRAR intensity image from Portage, USA (Lincoln Laboratory IRAR collection).

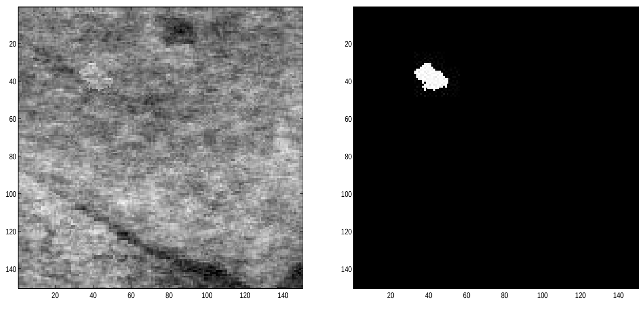


Figure 2: (a) Simulated cluttered target image, PTCR= 7.3 dB, (b) Clutter-free target template shown as a binary image.

airborne radar (IRAR) intensity imagery. The base image, shown in Figure 1, is an aerial scene from Portage USA, extracted from the IRAR image collection at Johns Hopkins University's Center for Imaging Sciences. For details on the simulation of the clutter background sequence, see [11]. We add to the background sequence a simulated target template that moves according to a white noise acceleration model, see section 2, with parameters $q = 10$ and $\Delta = 4ms$. The spatial resolution (pixel size) is $\Delta_1 = \Delta_2 = 20cm$. The image frame extends from 0 to 30 meters (150 pixels) in both the horizontal and vertical dimensions. The target's (continuous) initial vertical and horizontal positions are uniformly distributed respectively between 4 and 12 meters, and between 4 and 8 meters. The initial target velocity is 10 $m/s$ in both dimensions. Figures 2(a) and (b) show respectively the simulated cluttered and clutter-free image of a target centered at pixel location $(40, 40)$. The peak target-to-clutter ratio (PTCR) in Figure 2(a) is 7.3 dB. Figures 3 and 4 show the root mean-square error (RMSE) in meters of the MAP target centroid position estimates for a 3400-particle bootstrap tracker, respectively in the vertical and horizontal directions. The error curves were obtained from 45 Monte Carlo runs with PTCR lowered to -5.7 dB and 14 frames per run. The plots in Figures 3 and 4 show good steady-state tracking performance and low target acquisition time for the proposed bootstrap tracker.

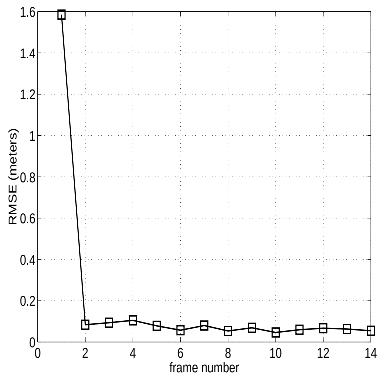


Figure 3: Bootstrap tracking performance, PTRC= -5.7 dB, vertical dimension.

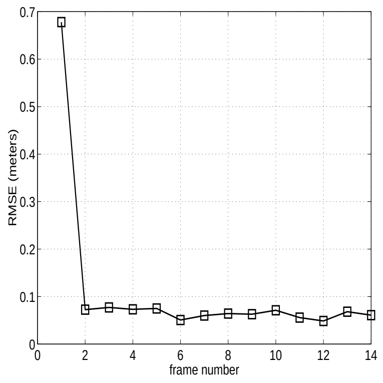


Figure 4: Bootstrap tracking performance, PTRC= -5.7 dB, horizontal dimension.

In the sequel, we compare the proposed bootstrap tracker to the linearized KBf tracker described in Table 2. Figure 5 shows the vertical RMS centroid position estimation error in meters, respectively for the 3400-particle bootstrap tracker with MAP estimation criterion (dashed line) and for the KBf tracker (solid line). The corresponding results for the horizontal dimension are shown in Figure 6. The performance curves were also estimated from 45 Monte Carlo runs with PTCR = -5.7 dB. We see from the plots in Figures 5 and 6 that the association ML/KBf has a very poor tracking performance, basically failing to acquire and track the target. Conversely, despite the low PTCR and poor visibility of the target, the proposed nonlinear bootstrap filter tracks the simulated vehicle with a final tracking error after 14 frames of less than 1 pixel within the image resolution.

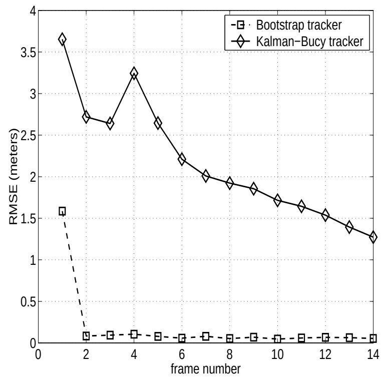


Figure 5: RMSE in meters for the bootstrap tracker (dashed) and the KBf tracker (solid), PTCR = -5.7 dB, vertical dimension.

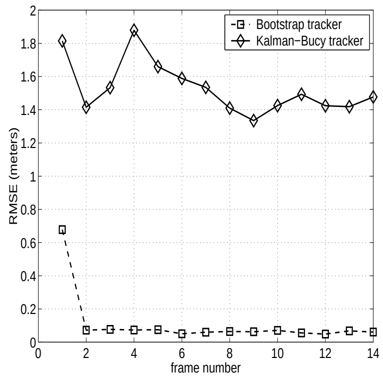


Figure 6: RMSE in meters for the bootstrap tracker (dashed) and the KBf tracker (solid), PTCR = -5.7 dB, horizontal dimension.

## 7. CONCLUSIONS

We introduced in this paper a new particle filter (bootstrap) algorithm for direct target tracking from image sequences and tested its performance using a Monte Carlo simulation. The simulation results show good tracking performance for the proposed tracker using 3400 particles and 14 image frames in a scenario of low target-to-clutter ratio. In contrast to the nonlinear boostrap tracker, a linearized Kalman-Bucy tracker fails to acquire and track the target under the same simulation conditions.

## 8. REFERENCES

[1] M. G. S. Bruno and J. M. F. Moura, “Multiframe detection/tracking in clutter: optimal performance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol.37, n.3, pp 925-946, July 2001.

[2] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS, Storrs, CT, 1995.

[3] M. S. Arulampalam, S. Maskell, N. J. Gordon and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol.50, n.2, pp 174-188, February 2002.

[4] A. Doucet, J. F. G. Freitas, and N. J. Gordon, “An introduction to sequential Monte Carlo methods,” in *Sequential Monte Carlo Methods in Practice*, A. Doucet, J. F. G. Freitas, and N. J. Gordon, Editors. New York: Springer-Verlag, 2001.

[5] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, ”Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *IEE Proceedings-F*, vol.140, n.2, pp 107-113, 1993.

[6] J. S. Liu and R. Chen, “Blind deconvolution via sequential imputations,” *Journal of the American Statistical Association*, 90(430), pp 567-576, 1995.

[7] S. Thurn, W. Bugard, and D. Fox, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Machine Learning*, vol.31, pp 29-53, 1998.

[8] A. Blake, B. Bascle, M. Isard, and J. MacCormick, “Statistical models of visual shape and motion,” *Philosophical Transactions of the Royal Society A.*, vol.356, pp 1283-1302, 1998.

[9] N. Bergman, L. Ljung, and F. Gustafsson, ”Terrain navigation using Bayesian statistics,” *IEEE Control Systems Magazine*, vol.19, n.3, pp 33-40, 1999.

[10] J. M. F. Moura and N. Balram, “Noncausal Gauss-Markov random fields: parameter structure and estimation,” *IEEE Transactions on Information Theory*, IT-39(4), pp 1333-1355, July 1993.

[11] M. G. S. Bruno and J. M. F. Moura, “Clutter adaptive tracking of multiaspect targets in IRAR imagery,” *Proceedings ICASSP 2002*, Orlando FL, USA, May 13-17, 2002.

summation limits accordingly in (12) for centroid locations near the borders, see [1] for details.
Clutter Model We capture the 2D spatial correlation of the background clutter using a noncausal, spatially homogeneous Gauss-Markov random field (GMrf) model [10]. The clutter returns at frame $n$, $V_n(i,\, j)$, $1 \le i \le L$, $1 \le j \le M$, are described by the 2D finite difference equation

$$\begin{aligned} V_n(i,\, j) &= \beta_v^c \left[ V_n(i-1,\, j) + V_n(i+1,\, j) \right] \\ &+ \quad \beta_h^c \left[ V_n(i,\, j-1) + V_n(i,\, j+1) \right] + U_n(i,\, j) \end{aligned} \quad (13)$$

where $E\left[ V_n(i,\, j)\, U_n(p,\, r) \right] = \sigma_c^2\, \delta_{i-p,\, j-r}$. The assumption of zero-mean clutter implies a pre-processing of the data that subtracts the mean of the background.

## 3.1. Likelihood Function

Let $\mathbf{y}_n$ be a 1D long-vector representation of the frame $\mathbf{Y}_n$ obtained by either row or columnwise scanning. Assuming a 2D GMrf background as in (13) and deterministic signature parameters $\{a_{k,l}\}$, we use the results in [1] to write the likelihood function of the observed $n$th frame as

$$p(\mathbf{y}_n \mid \mathbf{x}_{n,1},\, \mathbf{x}_{n,2}) \propto \exp\left[ \frac{2\lambda(\mathbf{x}_{n,1},\, \mathbf{x}_{n,2}) - \rho}{2\sigma_c^2} \right] . \quad (14)$$

where $\rho$ is a target energy term that is constant away from the image borders, see [1] for details. The function $\lambda$ in (14) is in turn given by [1]

$$\lambda(\mathbf{x}_{n,1},\, \mathbf{x}_{n,2}) = \sum_{k=-r_i}^{r_s} \sum_{l=-l_i}^{l_s} a_{k,l}\, \mu(x_n^*(1)+k,\, x_n^*(2)+l) \quad (15)$$

where $x_n^*(i)$, $i = 1, 2$, are obtained respectively from (10) and (11), and $\mu(p, r)$ is the output of the differential operator

$$\begin{aligned} \mu(p,r) &= Y_n(p,r) - \beta_h^c \left[ Y_n(p, r-1) + Y_n(p, r+1) \right] \\ &- \beta_v^c \left[ Y_n(p-1, r) + Y_n(p+1, r) \right] \end{aligned} \quad (16)$$

with Dirichlet (identically zero) boundary conditions. Equation (15) is valid for $r_i + 1 \le x_n^*(1) \le L - r_s$ and $l_i + 1 \le x_n^*(2) \le M - l_s$. For centroid positions close to the image borders, the summation limits in (15) must be varied accordingly as explained in [1]. Intuitively, we can interpret the function $\lambda$ in (15) as the concatenation of two linear filtering operations: first, we pass the image through a noncausal differential filter to generate a residual error image and, then, we apply to this error image a 2D correlation filter that is matched to the (known) 2D target template.

## 4. PARTICLE FILTER TRACKER

The on-line Bayesian estimation problem can be summarized as follows: given a sequence of observed frames $\mathbf{Y}_1^n = \{\mathbf{y}_1,\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n\}$ and the probability density function (pdf) $p(\mathbf{x}_0)$ of the initial (continuous-valued) target state, $\mathbf{x}_0$, compute recursively the *posterior* pdf $p(\mathbf{x}_n \mid \mathbf{Y}_1^n)$, for $n \ge 1$, using the Markovian transition kernel, $p(\mathbf{x}_{n+1} \mid \mathbf{x}_n)$, and the likelihood function, $p(\mathbf{y}_n \mid \mathbf{x}_n)$. From the posterior pdf, we can then infer the value of the hidden (unobserved) state at instant $n$, $\mathbf{x}_n$, using some optimality criteria, e.g. minimum mean-square error (MMSE) or maximum a posteriori (MAP).

Particle filters [3] are a simulation approach to Bayesian estimation in which the posterior pdf is represented at each instant $n$ by a set of particles $\left\{ \mathbf{x}_n^{(j)} \right\}$, $1 \le j \le N_p$, with associated weights $w_n^{(j)}$. The MMSE estimate of the hidden state $\mathbf{x}_n$ is then obtained simply as [3] a weighted average of the particles. Alternatively, the MAP estimate can be obtained [4] from the histogram of the particles.

Ideally, we would like the particles to be samples from the true posterior in which case all weights $w_n^{(j)}$ would be identical and equal to $1/Np$. In practice, however, either the posterior pdf is unavailable or difficult to sample from. An alternative approach known as *sequential importance sampling* (SIS) [4] is to sample $\mathbf{x}_n^{(j)}$ sequentially from an importance function $\pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)$ and update the weights using the recursion

$$w_n^{(j)} \propto w_{n-1}^{(j)} \frac{p(\mathbf{y}_n \mid \mathbf{x}_n^{(j)})\, p(\mathbf{x}_n^{(j)} \mid \mathbf{x}_{n-1}^{(j)})}{\pi(\mathbf{x}_n^{(j)} \mid (\mathbf{X}^{(j)})_0^{n-1},\, \mathbf{Y}_1^n)}, \qquad \sum_{j=1}^{N_p} w_n^{(j)} = 1 \,. \quad (17)$$

In principle, the importance function $\pi$ can be arbitrary provided that it satisfies two restrictions [4]

1. $\pi(\mathbf{X}_0^n \mid \mathbf{Y}_1^n)$ must have the same support as $p(\mathbf{X}_0^n \mid \mathbf{Y}_1^n)$ and be strictly positive and integrable to 1 in that support.

2. $\pi(\mathbf{X}_0^n \mid \mathbf{Y}_1^n) = \pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)\pi(\mathbf{X}_0^{n-1} \mid \mathbf{Y}_1^{n-1})$.

## 4.1. Bootstrap Tracker

The first practical SIS filter was the bootstrap filter [5], where the Markovian transition kernel $p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$ is used as the importance function $\pi(\mathbf{x}_n \mid \mathbf{X}_0^{n-1}, \mathbf{Y}_1^n)$. The weight update equation reduces then to

$$w_n^{(j)} \propto w_{n-1}^{(j)} p(\mathbf{y}_n \mid \mathbf{x}_n^{(j)}) \qquad \sum_{j=1}^{N_p} w_n^{(j)} = 1 \,. \quad (18)$$

The key innovation of the bootstrap filter [5] was to introduce an additional particle selection step to avoid the so-called *degeneracy phenomenon* [3], i.e., the tendency of the distribution of the particle weights to become skewed as the number of SIS iterations increases. The selection step proposed in [5] consisted simply of resampling $N_p$ times from the set $\left\{\mathbf{x}_n^{(j)}\right\}$ with replacement according to the particle weights so that low-weight particles are discarded whereas high-weight particles are multiplied. The new weights after the resampling step are all reset then to $1/N_p$.

Using the motion, target and clutter models from sections 2 and 3 and recalling the expressions for the Markovian transition kernel in subsection 2.1 and for the likelihood function in subsection 3.1, we present in Table 1 a bootstrap filter algorithm for target tracking in image sequences.

```
1. Initialization For $j = 1, \ldots, N_p$
• Draw $\mathbf{x}_{0,1}^{(j)} \sim p(\mathbf{x}_{0,1})$, $\mathbf{x}_{0,2}^{(j)} \sim p(\mathbf{x}_{0,2})$,
  make $w_0^{(j)} = 1/N_p$ and set $n = 1$.
2. Importance Sampling Step For $j = 1, \ldots, N_p$
• Draw $\tilde{\mathbf{x}}_{n,1}^{(j)} \sim N(\mathbf{F}\mathbf{x}_{n-1,1}^{(j)}, \mathbf{Q})$
  and $\tilde{\mathbf{x}}_{n,2}^{(j)} \sim N(\mathbf{F}\mathbf{x}_{n-1,2}^{(j)}, \mathbf{Q})$.
• Compute the importance weights
  $\tilde{w}_n^{(j)} \propto w_{n-1}^{(j)}\ p(\mathbf{y}_n \mid \tilde{\mathbf{x}}_{n,1}^{(j)}, \tilde{\mathbf{x}}_{n,2}^{(j)}) \qquad \sum_{j=1}^{N_p} \tilde{w}_n^{(j)} = 1$
  using equations (14), (15), and (16).
3. Selection Step
• Generate a new set of samples
  $\left\{\mathbf{x}_n^{(j)} = \left[(\mathbf{x}_{n,1}^{(j)})^T\ (\mathbf{x}_{n,2}^{(j)})^T\right]^T\right\} \qquad 1 \le j \le N_p$
  such that $P(\mathbf{x}_n^{(j)} = \tilde{\mathbf{x}}_n^{(k)}) = \tilde{w}_n^{(k)}$.
• Make $w_n^{(j)} = 1/N_p$, $1 \le j \le N_p$.
• While $n \le N_{\max}$, set $n = n + 1$
  and go back to step 2.
```

Table 1: Algorithm I: Bootstrap filter for target tracking in 2D cluttered image sequences.

Clutter Adaptation When the clutter model parameters $\beta_h^c$, $\beta_v^c$ and $\sigma_c^2$ are unknown, they must be estimated from the observed data. For computational simplicity, we use in this paper a suboptimal approach to clutter adaptation where we estimate the GMrf clutter parameters directly from each available sensor frame $\mathbf{Y}_n$ using a variation of *approximate maximum likelihood* (AML) parameter estimation algorithm introduced in [10].

## 5. LINEARIZED KALMAN-BUCY TRACKER

We compare the proposed nonlinear bootstrap tracker to the association of a single frame maximum likelihood (ML) position estimator and a linear Kalman-Bucy filter. The preliminary ML estimates of the 2D pixel location of the target centroid are given by

$$\widehat{\mathbf{x}_n^*} = \arg\max_{\mathbf{x}_n^*} p(\mathbf{y}_n \mid \mathbf{x}_n)\ . \qquad (19)$$

The pixel estimates of the centroid location are converted to continuous space and treated as decoupled noisy measurements of the true centroid positions in each dimension. A conventional Kalman-Bucy filter with knowledge of the state dynamics is then used to refine the preliminary ML estimates. Table 2 summarizes the algorithm. In Table 2, $\sigma_{v,i}^2$ denotes the variance of the ML centroid position estimation error in the dimension $i$ and $\mathbf{C}$ is the row vector $\mathbf{C} = [1\ 0]$. We denote the mean and covariance matrix of $\mathbf{x}_{0,i}$, $i = 1, 2$, respectively by $\overline{\mathbf{x}}_{0,i}$ and $\Sigma_{0,i}$. The symbols $\widehat{\mathbf{x}}_{n|n,i}$ and $\Sigma_{n|n,i}$ denote respectively the Kalman filter estimate of the target state $\mathbf{x}_{n,i}$ at instant $n$ in dimension $i$ and its associated error covariance matrix.

```
Initialization For $i = 1, 2$
• $\widehat{\mathbf{x}}_{0|0,i} = \overline{\mathbf{x}}_{0,i} \qquad \Sigma_{0|0,i} = \Sigma_{0,i}$
For $n = 1$ to $N_{\max}$
  ML Step
  Compute $p(\mathbf{y}_n \mid x_{n,1}(1), x_{n,2}(1))$
  using equations (14), (15) and (16), make
  $(\widehat{z}_1, \widehat{z}_2) = \arg\max_{\mathcal{L}} p(\mathbf{y}_n \mid x_{n,1}(1), x_{n,2}(1))$
  and $\tilde{x}_{n,i} = \text{round}(\widehat{z}_i\, \Delta_i), \qquad i = 1, 2$
  Kalman Filter Step For i=1,2
  • $\widehat{\mathbf{x}}_{n|n-1,i} = \mathbf{F}\widehat{\mathbf{x}}_{n-1|n-1,i}$
  • $\Sigma_{n|n-1,i} = \mathbf{F}\Sigma_{n-1|n-1,i}\mathbf{F}^T + \mathbf{Q}$
  • $\mathbf{K}_{n,i} = \Sigma_{n|n-1,i}\mathbf{C}^T \underbrace{(\mathbf{C}\Sigma_{n|n-1,i}\mathbf{C}^T + \sigma_{v,i}^2)}_{\mathbf{S}_{n|n-1}}{}^{-1}$
  • $\widehat{\mathbf{x}}_{n|n,i} = \widehat{\mathbf{x}}_{n|n-1,i} + \mathbf{K}_{n,i}(\tilde{x}_{n,i} - \mathbf{C}\widehat{\mathbf{x}}_{n|n-1,i})$
  • $\Sigma_{n|n,i} = \Sigma_{n|n-1,i} - \mathbf{K}_{n,i}\mathbf{S}_{n|n-1,i}\mathbf{K}_{n,i}^T$
End of for loop
```

Table 2: Algorithm II: Association ML Estimator + Linear Kalman-Bucy Filter (for comparison purposes only).

## 6. PERFORMANCE RESULTS

We compare next the tracking performances of the bootstrap tracker and the KBf tracker using a simulated image sequence that was generated from real infrared

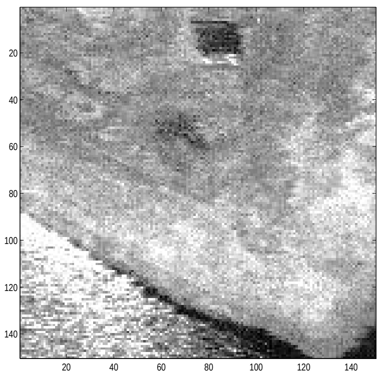
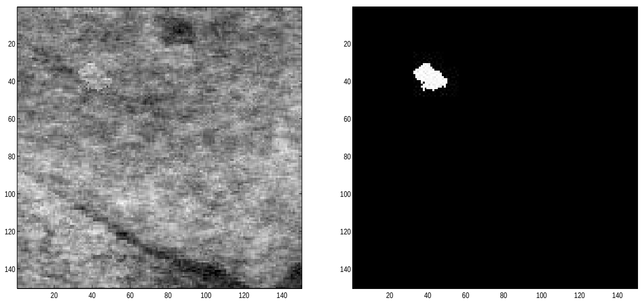


Figure 1: Real IRAR intensity image from Portage, USA (Lincoln Laboratory IRAR collection).



(a) (b)

Figure 2: (a) Simulated cluttered target image, PTCR= 7.3 dB, (b) Clutter-free target template shown as a binary image.

airborne radar (IRAR) intensity imagery. The base image, shown in Figure 1, is an aerial scene from Portage USA, extracted from the IRAR image collection at Johns Hopkins University's Center for Imaging Sciences. For details on the simulation of the clutter background sequence, see [11]. We add to the background sequence a simulated target template that moves according to a white noise acceleration model, see section 2, with parameters $q = 10$ and $\Delta = 4ms$. The spatial resolution (pixel size) is $\Delta_1 = \Delta_2 = 20cm$. The image frame extends from 0 to 30 meters (150 pixels) in both the horizontal and vertical dimensions. The target's (continuous) initial vertical and horizontal positions are uniformly distributed respectively between 4 and 12 meters, and between 4 and 8 meters. The initial target velocity is 10 $m/s$ in both dimensions. Figures 2(a) and (b) show respectively the simulated cluttered and clutter-free image of a target centered at pixel location $(40, 40)$. The peak target-to-clutter ratio (PTCR) in Figure 2(a) is 7.3 dB. Figures 3 and 4 show the root mean-square error (RMSE) in meters of the MAP target centroid position estimates for a 3400-particle bootstrap tracker, respectively in the vertical and horizontal directions. The error curves were obtained from 45 Monte Carlo runs with PTCR lowered to -5.7 dB and 14 frames per run. The plots in Figures 3 and 4 show good steady-state tracking performance and low target acquisition time for the proposed bootstrap tracker.

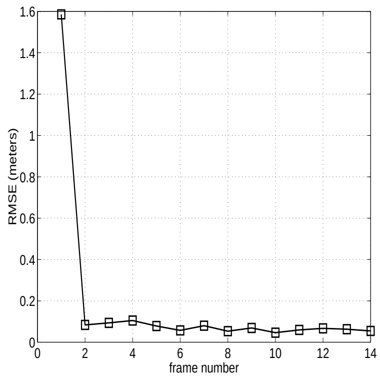


Figure 3: Bootstrap tracking performance, PTRC= -5.7 dB, vertical dimension.

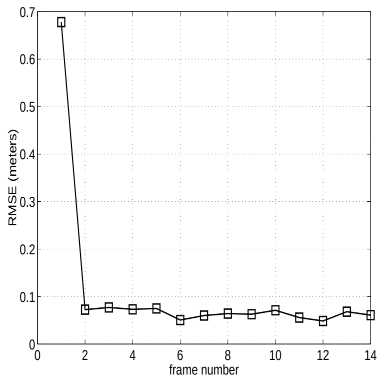


Figure 4: Bootstrap tracking performance, PTRC= -5.7 dB, horizontal dimension.

In the sequel, we compare the proposed bootstrap tracker to the linearized KBf tracker described in Table 2. Figure 5 shows the vertical RMS centroid position estimation error in meters, respectively for the 3400-particle bootstrap tracker with MAP estimation criterion (dashed line) and for the KBf tracker (solid line). The corresponding results for the horizontal dimension are shown in Figure 6. The performance curves were also estimated from 45 Monte Carlo runs with PTCR = -5.7 dB. We see from the plots in Figures 5 and 6 that the association ML/KBf has a very poor tracking performance, basically failing to acquire and track the target. Conversely, despite the low PTCR and poor visibility of the target, the proposed nonlinear bootstrap filter tracks the simulated vehicle with a final tracking error after 14 frames of less than 1 pixel within the image resolution.

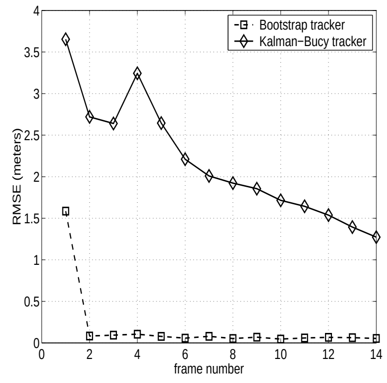


Figure 5: RMSE in meters for the bootstrap tracker (dashed) and the KBf tracker (solid), PTCR = -5.7 dB, vertical dimension.

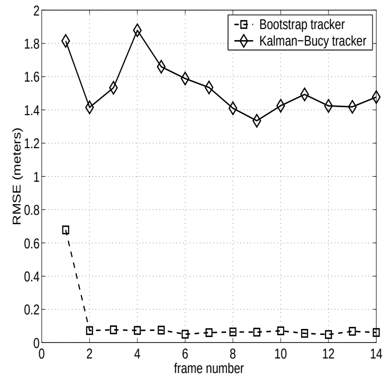


Figure 6: RMSE in meters for the bootstrap tracker (dashed) and the KBf tracker (solid), PTCR = -5.7 dB, horizontal dimension.

## 7. CONCLUSIONS

We introduced in this paper a new particle filter (bootstrap) algorithm for direct target tracking from image sequences and tested its performance using a Monte Carlo simulation. The simulation results show good tracking performance for the proposed tracker using 3400 particles and 14 image frames in a scenario of low target-to-clutter ratio. In contrast to the nonlinear boostrap tracker, a linearized Kalman-Bucy tracker fails to acquire and track the target under the same simulation conditions.

## 8. REFERENCES

[1] M. G. S. Bruno and J. M. F. Moura, “Multiframe detection/tracking in clutter: optimal performance,” *IEEE Transactions on Aerospace and Electronic Systems*, vol.37, n.3, pp 925-946, July 2001.

[2] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS, Storrs, CT, 1995.

[3] M. S. Arulampalam, S. Maskell, N. J. Gordon and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol.50, n.2, pp 174-188, February 2002.

[4] A. Doucet, J. F. G. Freitas, and N. J. Gordon, “An introduction to sequential Monte Carlo methods,” in *Sequential Monte Carlo Methods in Practice*, A. Doucet, J. F. G. Freitas, and N. J. Gordon, Editors. New York: Springer-Verlag, 2001.

[5] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, ”Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *IEE Proceedings-F*, vol.140, n.2, pp 107-113, 1993.

[6] J. S. Liu and R. Chen, “Blind deconvolution via sequential imputations,” *Journal of the American Statistical Association*, 90(430), pp 567-576, 1995.

[7] S. Thurn, W. Bugard, and D. Fox, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Machine Learning*, vol.31, pp 29-53, 1998.

[8] A. Blake, B. Bascle, M. Isard, and J. MacCormick, “Statistical models of visual shape and motion,” *Philosophical Transactions of the Royal Society A.*, vol.356, pp 1283-1302, 1998.

[9] N. Bergman, L. Ljung, and F. Gustafsson, ”Terrain navigation using Bayesian statistics,” *IEEE Control Systems Magazine*, vol.19, n.3, pp 33-40, 1999.

[10] J. M. F. Moura and N. Balram, “Noncausal Gauss-Markov random fields: parameter structure and estimation,” *IEEE Transactions on Information Theory*, IT-39(4), pp 1333-1355, July 1993.

[11] M. G. S. Bruno and J. M. F. Moura, “Clutter adaptive tracking of multiaspect targets in IRAR imagery,” *Proceedings ICASSP 2002*, Orlando FL, USA, May 13-17, 2002.