Estimation of Dynamically Varying Support of Sparse Signals via Sequential Monte-Carlo Method

Jin Hyeok Yoo, Sun Hong Lim, Byonghyo Shim∗, and Jun Won Choi
Hanyang University
Email: {jhyoo, shlim}@spa.hanyang.ac.kr, junwchoi@hanyang.ac.kr
Seoul National University∗
Email: bshim@snu.ac.kr

Abstract—In this paper, we address the problem of tracking time-varying support of a sparse signal given a sequence of observation vectors. We model the dynamic variation of the support set using the discrete-state Markov process and employ the Rao-Blackwellized sequential Monte Carlo method, which allows for separate tracking of the support set and the amplitude of the unknown signals. Specifically, the samples for the support variables are drawn from their posteriori joint distributions using a Gibbs sampler while the continuous amplitude variables are separately estimated using the Kalman filter. Our numerical evaluation shows that the proposed method achieves significant performance gain over the existing sparse estimation methods.

Index Terms—Sparse recovery algorithm, Compressed sensing, Particle filter, Support recovery, Rao-Blackwellization, Sequential Monte-Carlo method

I. INTRODUCTION

Let us consider a scenario where the observation vectors \( y_1, \ldots, y_n \) are sequentially acquired by a noisy linear system

\[
y_n = H_n g_n + w_n, \tag{1}
\]

where \( n \) is the time index, \( y_n \) is the \( N \times 1 \) observation vector, \( H_n \) is the \( N \times M \) system matrix, \( g_n \) is the \( M \times 1 \) unknown signal vector, and \( w_n \) is the Gaussian noise vector \( \mathcal{CN}(0, W_n) \). In particular, we are interested in the problem of estimating the unknown vector \( g_n \) based on the measurement vectors \( y_1, \ldots, y_n \) given the knowledge of the system matrices \( H_1, \ldots, H_n \). Such a problem arises frequently in various signal processing applications including channel estimation for wireless communications, image reconstruction using dynamic MRI, and indoor positioning in a wireless network [1]–[3].

Given the statistical model on \( g_n \) and \( w_n \), the optimal solution to such problem can be found under the Bayesian framework. The maximum a posteriori (MAP) estimate of \( g_n \) is given by

\[
g_n = \arg \max_{g_n} P(g_n | y_1, \ldots, y_n). \tag{2}
\]

Depending on the assumed model and the underlying structure of the systems, different estimation algorithms can be derived to find the solution of (2).

For example, consider a scenario wherein the signal vectors \( g_1, \ldots, g_n \) are statistically uncorrelated with each other.

Because \( g_n \) is independent of the previously acquired observations \( y_1, \ldots, y_{n-1} \), the MAP estimate of \( g_n \) in (2) can be simplified to

\[
g_n = \arg \max_{g_n} P(g_n | y_n). \tag{3}
\]

Assuming that both \( g_n \) and \( w_n \) are zero-mean Gaussian, the optimal MAP estimate is given by

\[
g_n = Q_n H_n^H (H_n Q_n H_n^H + W_n)^{-1} y_n, \tag{4}
\]

where \( Q_n \) and \( W_n \) are the covariance matrices of \( g_n \) and \( w_n \), respectively. In contrast, if \( g_1, \ldots, g_n \) are assumed to be temporally correlated, it would be better to perform joint processing of the multiple measurement vectors \( y_1, \ldots, y_n \). In particular, when the temporal evolution of \( g_n \) is modeled by the Gauss-Markov process, then the optimal linear estimate of \( g_n \) can be obtained using the Kalman filter. This filter recursively updates the first and second-order statistics of the posteriori distribution of \( g_n \) whenever a measurement vector \( y_n \) is available.

We often encounter the scenarios where the signal \( g_n \) possesses additional structures and properties. One of the typical structures frequently observed in natural signals is sparsity, meaning that the signal vector \( g_n \) is represented by only a few coefficients on an appropriately chosen basis. That is, if \( g_n \) is represented with the basis matrix \( \Phi \), i.e., \( g_n = \Phi s_n \), most elements of \( s_n \) are close to zero. If we use such a sparsity model for \( g_n \), then the observation vector \( y_n \) can be expressed as

\[
y_n = H_n \Phi s_n + w_n, \tag{5}
\]

\[
= B_n s_n + w_n, \tag{6}
\]

where \( g_n = \Phi s_n \) and \( B_n = H_n \Phi \). It should be noted that the estimation of \( g_n \) is reformulated into the problem of estimating the sparse vector \( s_n \). It is well known that when the size of \( y_n \) is smaller than that of \( s_n \), i.e., \( N < M \), the conventional linear estimator fails to determine a good estimate of the sparse vector \( s_n \). In previous decades, the signal processing community has tried to uncover a condition in which the unknown signal \( s_n \) can be recovered from the observation \( y_n \) in an underdetermined setup. According to the theoretical framework called compressed sensing (CS) [1], [4], [5], a \( K \)-sparse vector (i.e., the number of nonzero elements is \( K \)) can be recovered only from a vector of \( K \log(\frac{N}{K}) \)
randomly projected measurements. Motivated by this, various efficient recovery algorithms (referred to as sparse estimation algorithms in the sequel) have been proposed. The well-known sparse estimation algorithms include the orthogonal matching pursuit (OMP) [6], generalized OMP [7], CoSamp [8], basis pursuit (BP) [9], and multipath matching pursuit [10].

When the sparse vector $s_n$ is temporally correlated, we exploit this feature to find a better solution. In general, the sparse vector $s_n$ can be represented by the support set (i.e., the indices of nonzero elements) and the amplitude of nonzero elements. Depending on how the support set and the amplitude vary in time, we can further categorize the scenarios into four distinct scenarios.

- Scenario I) The amplitude of $s_n$ is time-varying. The support set and the sensing matrix $B_n$ are locally fixed.
- Scenario II) The amplitude of $s_n$ and the sensing matrix $B_n$ are time-varying. The support set is locally fixed.
- Scenario III) The amplitude and the support set of $s_n$ are time-varying. The sensing matrix $B_n$ is fixed.
- Scenario IV) The amplitude, the support set of $s_n$ and the sensing matrix $B_n$ are all time-varying.

All four scenarios are illustrated in Fig. I. In scenario I, we can construct the measurement matrix $Y$ by stacking the sequence of the measurement vectors as

$$Y = [y_1 \ldots y_n] = B [s_1 \ldots s_n] + [w_1 \ldots w_n] = BS + W$$

where $B = B_1 = \ldots = B_n$, $S = [s_1 \ldots s_n]$, and $W = [w_1 \ldots w_n]$. The sparse estimation algorithm estimates the matrix $S$ from the measurement matrix $Y$ under the constraint of the common sparsity for $s_1, \ldots, s_n$. The well-known algorithms for this scenario include the simultaneous OMP (SOMP) [11], multiple sparse Bayesian learning [12], convex relaxation [13], TSBL [14], MUSIC-CS [15], and Kalman-CS [16]. In scenario II, some major modifications to the algorithm are required to deal with the time-varying sensing matrix; such algorithms include sKTS [17], KSTBL [18] and AMP-MMV [19]. In scenarios III and IV, both the support set and the amplitude of the sparse signal are time-varying. We are particularly interested in the scenario wherein the support set of $s_n$ varies continuously with some temporal correlation. Because the support set changes across the consecutive measurement vectors, the recovery algorithms designed for scenarios I and II would not work in these scenarios and the algorithms should be able to track the time-varying support set. The algorithms developed for scenario III include the modified CS [20], GF-Lasso [21], PaFiMoCS [22], KF-CS [23] and IHT-PKS [24]. In [25], some algorithms developed for scenario III were modified to handle Scenario IV, including dynamic regularized modified BPDN (reg-mod-BPDN) [26], streaming modified weighted-$\ell_1$ (streaming mod-wl1) [27], Bayesian sequential CS [28], BCSKF [29], and Kalman filtered modified-CS (KF-ModCS) [16].

In this paper, we propose a new recovery algorithm that tracks both the time-varying support and amplitude of the sparse signals for scenarios III and IV. The proposed method is based on the Bayesian filtering framework, where both the time-varying support set and the amplitude of the signal are sequentially estimated given the previously acquired observation vectors. Assuming that the support set of the sparse signal is slowly varying, we model it using the discrete-state
Markov process. This yields the dynamic state space model, which comprises two types of state variables: discrete support and continuous amplitude variables. However, it is difficult to determine the joint optimal solution to such a problem with feasible computational complexity. To make the solution tractable, we propose a sub-optimal solution based on the sequential Monte-Carlo method [30]. The proposed method, referred to as the Monte-Carlo support tracking (MST) algorithm, applies Rao-Blackwellization (RB) [31] to formulate the joint estimation of the mixed type of variables into the estimation of each one. Specifically, given that the observation vector is linearly dependent on the amplitude variable for the given support set, we marginalize the amplitude variable from the joint posteriori distribution and draw only the samples for the support set. We then employ the Gibbs sampler to sample the K elements of the support set simultaneously from their joint density. For each sample, the amplitude variable is separately estimated using the Kalman filter. We also develop a low-complexity implementation of our MST algorithm exploiting the particular structure of the problem. We evaluate our algorithm in the context of estimating 1) the synthetically generated sparse data and 2) the sparse channels in millimeter-wave communications. Our extensive simulations show that our algorithm significantly outperforms the existing algorithms in both scenarios. The contributions of our work are summarized as follows:

- We employ the sequential Monte-Carlo method for joint estimation of amplitude and support. In fact, the sequential Monte-Carlo methods have been applied to CS problems previously in [22], [28]. Our method is different from these methods in that dynamic variations of the support set are explicitly modeled using the discrete-state Markov process. In addition, owing to the RB, the dual interaction between Gibbs sampling and Kalman filter is enabled to solve the sequential CS problem for tracking the support set and amplitude separately.
- While most sequential Monte-Carlo methods use the prior state distribution to generate the samples, our method generates the samples from the posteriori distribution with an aid of the Gibbs sampler. The Gibbs sampler offers significant improvement in sample efficiency and the use of the Gibbs sampler for this purpose has not been tried in the previous literature.
- Our numerical evaluation demonstrates that the proposed algorithm offers state-of-the-art performance regarding a sequential estimation of the sparse signals with time-varying support.

The remainder of this paper is organized as follows. In Section II, we describe the system and signal model used for our method. In Section III, we derive the proposed sparse signal estimation algorithm and present its low-complexity implementation. The simulation results are presented in Section IV and the paper is concluded in Section VI.

II. SYSTEM MODEL

In this section, we introduce the system model for the proposed MST algorithm. Recall that the measurement vector $y_n$ is expressed as $y_n = B_n s_n + w_n$, where $y_n$ is the $N \times 1$ vector, $s_n$ is the $M \times 1$ vector, and $w_n$ is distributed as $CN(0, W_n)$. We set $N \leq M$ and the number of nonzero entries in $s_n$ is $K(< M)$. Let $\mathcal{D}_n = \{d_{n,1}, \ldots, d_{n,K}\}$ be the support set, i.e., the indices of nonzero entries in $s_n$. Accordingly, we define the $M$-dimensional vector $d_n$, which has a value of one for the $d_{n,k}$th entry and zero for the rest. Subsequently, the sparse signal $s_n$ can be expressed as

$$s_n = \Lambda(\mathcal{D}_n) g_n,$$

where $\Lambda(\mathcal{D}_n) = \text{diag}(d_n)$ and $g_n$ is the $M \times 1$ amplitude vector. For example, if $M = 4$, $K = 2$ and $\mathcal{D}_n = \{1,3\}$, we have $d_n = \{1,0,1,0\}^T$ and

$$\Lambda(\mathcal{D}_n) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$ (11)

With this representation of $s_n$, the observation vector $y_n$ is expressed as

$$y_n = B_n \Lambda(\mathcal{D}_n) g_n + w_n.$$ (12)

Assume that the support set $\mathcal{D}_n$ changes with the time index $n$. The temporal variation in the $k$th element $d_{n,k}$ in $\mathcal{D}_n$ can be modeled by the discrete-state Markov random process. The probability that the state $m_i$ for $d_{n-1,k}$ changes into the state $m_j$ for $d_{n,k}$ is given by the transition probability

$$T_{i,j} = \text{Pr}(d_{n,k} = m_j | d_{n-1,k} = m_i) \text{ for } 1 \leq m_i, m_j \leq M.$$ (13)

Assuming that each element of the support changes slowly, one can, for example, use the exponentially decaying function for the transition probability, i.e.,

$$T_{i,j} = \frac{1}{C} e^{\alpha |m_i - m_j|},$$ (14)

where $C$ is the normalization factor. Note that the smaller the parameter $\alpha \in (0,1)$ is, the slower the support index changes in time. We also assume that the elements of the support set are independent of each other, i.e.,

$$\text{Pr}(\mathcal{D}_n | \mathcal{D}_{n-1}) = \prod_{k=1}^{K} \text{Pr}(d_{n,k} | d_{n-1,k}).$$ (15)

The amplitude vector $g_n$ in (10) is also modeled by the Gauss-Markov process, where the temporal evolution of the amplitude vector $g_n$ is described by

$$g_n = F_n g_{n-1} + v_n,$$ (16)

where $v_n$ is the Gaussian process noise $CN(0, V_n)$ and the matrix $F_n$ is the auto-regressive parameter. If we assume independence between the nonzero elements of $s_n$, both matrices $F_n$ and $V_n$ should be diagonal matrix. Combining (14), (15), and (12), we can construct the state-space model.

- Prior dynamic model

$$\text{Pr}(\mathcal{D}_n | \mathcal{D}_{n-1}) = \prod_{k=1}^{K} \text{Pr}(d_{n,k} | d_{n-1,k})$$ (16)

$g_n = F_n g_{n-1} + v_n$ (17)
• Measurement model

\[ y_n = B_n x_n + w_n \]  

It should be noted that this state space model has mixed types of states: the discrete support variable \( D_n \) and the continuous amplitude variable \( g_n \). In the next section, we present the method by which to estimate these two state variables sequentially in a computationally efficient manner.

### III. PROPOSED SPARSE ESTIMATION TECHNIQUE

We aim to determine the estimate of the sparse signal \( s_n \) based on the measurement vectors acquired up to the current time step, \( y_1, ..., y_n \). To this end, we pursue the joint MAP estimation of the support set \( D_n \) and the amplitude vector \( g_n \), i.e.,

\[ \hat{D}_n, \hat{g}_n = \arg \max_{D_n, g_n} P(D_n, g_n | y_{1:n}) \]  

where \( y_{1:n} = \{ y_1, ..., y_n \} \). To find an optimal solution to (19), we need to account for all possible transitions from \( D_{n-1} \) to \( D_n \), which makes it difficult to obtain the joint estimate of \( D_n \) and \( g_n \) with feasible complexity. In this case, the sequential Monte-Carlo method (a.k.a. particle filter [32]) has been popularly used to find a sub-optimal solution with reduced complexity.

The sequential Monte-Carlo method needs to generate the samples for \( D_n \) and \( g_n \). However, sampling the mixed type of variables \( D_n \) and \( g_n \) from their joint posterior distribution is difficult owing to the high dimension of \( g_n \) and a large number of configurations of \( D_n \). To reduce the complexity for dealing with these variables, we employ the RB, which decouples the joint estimation of \( D_n \) and \( g_n \) into the estimation of the individual variables. Owing to the RB, we can generate the samples only for \( D_n \) and the variable \( g_n \) can be linearly estimated using the Kalman filter.

#### A. Sampling step

To apply the RB, we divide the full posterior distribution of \( D_{0:n} \) and \( g_n \) into two terms [31]

\[ P(D_{0:n}, g_n | y_{1:n}) = P(g_n | D_{0:n}, y_{1:n}) P(D_{0:n} | y_{1:n}) \]  

where

\[ D_{j:k} = \{ D_j, ..., D_k \} \]  

Note that the first term \( P(g_n | D_{0:n}, y_{1:n}) \) follows a Gaussian distribution \( \mathcal{N}(\hat{g}_{n|n}, P_{n|n}) \), where \( \hat{g}_{n|n} = E[g_n | D_{0:n}, y_{1:n}] \) and \( P_{n|n} = \text{Cov}[g_n | D_{0:n}, y_{1:n}] \).

The non-Gaussian distribution \( P(D_{0:n} | y_{1:n}) \) in (20) is approximated using \( N_s \) samples, \( D_{0:n}^{(1)}, ..., D_{0:n}^{(N_s)} \), drawn according to the importance density \( Q(D_{0:n} | y_{1:n}) \). The importance weight \( w_n^{(i)} \) is used to reflect the disparity between \( Q(D_{0:n} | y_{1:n}) \) and \( P(D_{0:n} | y_{1:n}) \) for the \( i \)-th sample \( D_{0:n}^{(i)} \). The importance weight is given by

\[ w_n^{(i)} = \frac{P(D_{0:n}^{(i)} | y_{1:n})}{Q(D_{0:n}^{(i)} | y_{1:n})} \]  

Then, the distribution \( P(D_{0:n} | y_{1:n}) \) can be approximated using [32]

\[ P(D_{0:n} | y_{1:n}) \approx \sum_{i=1}^{N_s} w_n^{(i)} \delta(D_{0:n}, D_{0:n}^{(i)}) \]  

where the importance weights are normalized such that \( \sum_{i=1}^{N_s} w_n^{(i)} = 1 \), and

\[ \delta(D_{0:n}, D_{0:n}^{(i)}) = \begin{cases} 1 & \text{if } D_j = D_j^{(i)} \text{ for } j \in [0, n] \\ 0 & \text{otherwise} \end{cases} \]

where \( D_j = D_j^{(i)} \) implies that two sets have the same elements. As the number of samples increases, the approximation error decreases.

To draw the samples with the sequential sampling procedure, we restrict the importance density \( Q(D_{0:n} | y_{1:n}) \) to factorize such that [32]

\[ Q(D_{0:n} | y_{1:n}) = Q(D_{0:n-1} | y_{1:n-1}) Q(D_n | D_{0:n-1}, y_{1:n}) \]  

This choice allows us to generate the \( i \)-th sample \( D_{0:n}^{(i)} \) by appending the new sample \( D_{0:n}^{(i)} \) drawn from \( Q(D_n | D_{0:n-1}, y_{1:n}) \) to the existing one \( D_{0:n-1}^{(i)} \). There exist an infinite number of choices for determining the conditional importance density \( Q(D_n | D_{0:n-1}, y_{1:n}) \).

The best importance density in terms of sample efficiency is the posteriori density, i.e., \( Q(D_n | D_{0:n-1}, y_{1:n}) = P(D_n | D_{0:n-1}, y_{1:n}) \) [32]. The posteriori density is not frequently used as an importance density owing to the high complexity of the sampling process. We will show that our method can efficiently generate these samples from the posteriori density using Gibbs sampling and some judicious approximation, and such a choice enables good estimation accuracy with significantly fewer samples as compared to the case where the priori density is used. Furthermore, owing to the high sample efficiency, we do not need the resampling step which is commonly used to alleviate the degeneracy problem [32].

Next, we present the manner by which to draw the samples for the \( K \) elements \( \{ d_{n,1}, ..., d_{n,K} \} \) \( \in D_n \) from their joint distribution \( Q(d_n | D_{0:n-1}, y_{1:n}) \). Since sampling multiple variables from their joint density is not an entirely straightforward process, we employ Gibbs sampling, which samples each element one by one based on its distribution conditioned on the rest [33]. After a sufficient number of iterations (called burn-in time), the samples generated via Gibbs sampling obey their joint distribution [33]. The Gibbs sampling samples \( d_{n,1}, ..., d_{n,K} \) as

- for \( i = 1 \) to \( N_s \)
  - for \( iter = 1 \) to \( L \)
    * for \( k = 1 \) to \( K \)
      \[ d_{n,k}^{\text{cur}} = P(d_{n,k} | D_{0:n,k-1}, d_{n,k+1:n,K}, D_{0:n-1}^{(i)}, y_{1:n}) \]  
    * end for
  - end for
  - \( D_{0:n}^{(i)} = \{ d_{n,1}^{\text{cur}}, d_{n,2}^{\text{cur}}, ..., d_{n,K}^{\text{cur}} \} \)
\* end for

where \( \mathcal{D}_{n,k} = \{d_{n,j} : j \leq k \} \) and \( L \) is the burn-in time. Note that \( \mathcal{D}^{\text{cum}}_{n,0,k-1} \) denotes the samples generated up to the current iteration while \( \mathcal{D}^{\text{q cum}}_{n,k,1:k} \) denotes those for the previous iteration. The conditional probability density used in (26) is given by

\[
P(d_{n,k} | \mathcal{D}_{n,0:k-1}, \mathcal{D}_{n,k+1:k}, \mathcal{D}^{(i)}_{0:n-1}, y_{1:n}) = CP(y_{n} | \mathcal{D}_{n}, \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(d_{n,k} | g_{n-1,k})
\]

where \( C \) is the normalizing constant. (see the derivation in Appendix A.) Note that the first term on the right-hand side of (27) is viewed as the likelihood term, while the second term is the prior term specified by the transition probability of the Markov process. The likelihood term can be easily obtained as

\[
P(y_{n} | \mathcal{D}_{n}, \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) = \mathcal{N}(B_{n}A(\mathcal{D}_{n})g_{n-1,k}, (B_{n}A(\mathcal{D}_{n}))^T + W_{n},)
\]

where

\[
\hat{g}_{n-1}^{(i)} = E[g_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}]
\]

\[
P_{n-1}^{(i)} = \text{Cov}[g_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}],
\]

see the derivation in Appendix B. Note that \( \hat{g}_{n-1}^{(i)} \) and \( P_{n-1}^{(i)} \) can be obtained as a result of the prediction step of the Kalman filter when we apply the Kalman filter to \( y_{n} \) assuming that \( \mathcal{D}_{0:n-1} = \mathcal{D}^{(i)}_{0:n-1} \). Note also that for all \( N_{a} \) samples, we need to update the \( N_{a} \) Kalman filters in parallel. The prediction update equations of the \( i \) th Kalman filter are given by

\[
\hat{g}_{n-1}^{(i)} = F_{n}g_{n-1,k}^{(i)}
\]

\[
P_{n-1}^{(i)} = F_{n}P_{n-1,k}^{(i)}F_{n}^T + W_{n}.
\]

Finally, from (27) and (28), the conditional density used for Gibbs sampling is given by

\[
P(d_{n,k} | \mathcal{D}_{n,0:k-1}, \mathcal{D}_{n,k+1:k}, \mathcal{D}^{(i)}_{0:n-1}, y_{1:n}) = \mathcal{C}\mathcal{N}(B_{n}A(\mathcal{D}_{n})\hat{g}_{n-1,k}, (B_{n}A(\mathcal{D}_{n}))^T + W_{n}) P(d_{n,k} | d_{n-1,k}^{(i)}).
\]

### B. Importance weight calculation

Once we draw the samples for \( \mathcal{D}_{n} \) from the importance density, we need to assign the importance weight to each sample. We can show from (22) and (25) that the importance weight \( w_{n}^{(i)} \) can be recursively calculated as [30]

\[
w_{n}^{(i)} = w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

If we employ the posterior distribution as the importance density, i.e., \( Q(\mathcal{D}_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n}) = P(\mathcal{D}_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n}) \), the importance weight update rule becomes

\[
w_{n}^{(i)} = w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

\[
= w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

\[
= w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

\[
= w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

\[
= w_{n-1}^{(i)} \frac{P(y_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n-1}) P(\mathcal{D}^{(i)}_{0:n-1})}{Q(\mathcal{D}^{(i)}_{0:n-1}, y_{1:n})}.
\]

Note that \( P(\mathcal{D}_{n} | \mathcal{D}^{(i)}_{0:n-1}, y_{1:n}) \) is Gaussian with the mean \( \hat{g}_{n}^{(i)} \) and the covariance matrix \( P_{n}^{(i)} \). It should also be noted that
TABLE I

COMPUTATIONAL COMPLEXITY REQUIRED FOR EACH STEP OF THE PROPOSED MST ALGORITHM

<table>
<thead>
<tr>
<th>Step</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalman prediction</td>
<td>( \sim O(M) )</td>
</tr>
<tr>
<td>Gibbs sampling</td>
<td>( \sim O(MK^{N^3}) )</td>
</tr>
<tr>
<td>Importance weight update</td>
<td>( \sim O(M) )</td>
</tr>
<tr>
<td>Kalman measurement update</td>
<td>( \sim O(M^2) )</td>
</tr>
</tbody>
</table>

\( \hat{g}_{n|n-1}^{(i)} \) and \( P_{n|n-1}^{(i)} \) can be efficiently calculated using the measurement update stage of the Kalman filter under the assumption that \( D_{0:n} = D_{0:n}^{(i)} \). For the given \( D_{0:n} \), the measurement update equations for the Kalman filter are given by

\[
\hat{g}_{n|n-1}^{(i)} = \hat{g}_{n-1|n-1} + K_n^{(i)} \left( y_n - B_n \Lambda_{\{D_{n|n}^{(i)}\}} \hat{g}_{n|n-1}^{(i)} \right)
\]

\( P_{n|n-1}^{(i)} = (I - K_n^{(i)} B_n \Lambda_{\{D_{n|n}^{(i)}\}}) P_{n|n-1}^{(i)} \)

\( K_n^{(i)} = P_{n|n-1}^{(i)} \left( B_n \Lambda_{\{D_{n|n}^{(i)}\}} \right)^H (B_n \Lambda_{\{D_{n|n}^{(i)}\}}) + W_n \)^{-1}

Then, the estimate of \( D_n \), \( D_n^{*} \) is obtained by

\[
D_n^{*} = \arg \max_{D_n} \sum_{i=1}^{N_N} \frac{w_n^{(i)}}{(n)^M \left| P_{n|n}^{(i)} \right|} \delta(D_n, D_n^{(i)}) \left( \max_{g_n} P(g_n|D_{0:n}; y_{1:n}) \right)
\]

(47)

\[
= \arg \max_{D_n} \sum_{i=1}^{N_N} \frac{w_n^{(i)}}{(n)^M \left| P_{n|n}^{(i)} \right|} \delta(D_n, D_n^{(i)}).
\]

(48)

The maximizer \( D_n^{*} \) can be easily found by selecting the sample that maximizes \( \frac{w_n^{(i)}}{(n)^M \left| P_{n|n}^{(i)} \right|} \) among \( D_n^{(1)}, \ldots, D_n^{(N_N)} \).

Once the estimate \( D_n^{*} \) is found, the estimate of \( g_n \) can be obtained by

\[
g_n^{*} = \arg \max_{g_n} P(g_n|D_{0:n}; y_{1:n})
\]

(49)

where \( g_n^{*} \) can also be easily obtained from the measurement update of the Kalman filter under the assumption that \( D_{0:n} = D_{0:n}^{*} \).

D. Algorithm summary

In Algorithm 1, we summarize the procedure for the \( n \)th sequential update of the proposed MST algorithm. Note that our algorithm sequentially updates the samples for the support set using the Gibbs sampler while the first and second-order statistics of the amplitude associated with the samples are updated through \( N_N \) Kalman filters. First, we obtain \( \hat{g}_{n|n-1}^{(i)} \) and \( P_{n|n-1}^{(i)} \) from the prediction update of the \( n \)th Kalman filter. Next, the samples for \( D_{n|n}^{(i)} \) are generated using the Gibbs sampler and the corresponding importance weights are updated. Then, the measurement update for the \( n \)th Kalman filter is performed. We repeat this procedure for all \( N_N \) samples and then compute the approximate MAP estimate according to (48) and (49).

Algorithm 1 Update at the time step \( n \) for the proposed MST algorithm

Input : \( \{\hat{g}_{n-1|n-1}^{(1)}, \ldots, \hat{g}_{n-1|n-1}^{(N_N)}\}, \{P_{n-1|n-1}^{(1)}, \ldots, P_{n-1|n-1}^{(N_N)}\} \)

Output : \( \{\hat{g}_{n|n}^{(1)}, \ldots, \hat{g}_{n|n}^{(N_N)}\}, \{P_{n|n}^{(1)}, \ldots, P_{n|n}^{(N_N)}\}, D_n^{*}, g_n^{*} \)

1: for \( i = 1 \) to \( N_N \) do

2: Kalman prediction update according to (31) and (32).

3: The samples \( D_{n|n}^{(i)} = \{d_{n,1}^{(i)}, d_{n,2}^{(i)}, \ldots, d_{n,N_N}^{(i)}\} \) are drawn using Gibbs sampling.

4: The importance weight is updated according to (39).

5: Kalman measurement update according to (43) - (45).

6: end for

7: The importance weights are normalized : \( w_n^{(i)} = \sum_{j=1}^{N_N} w_n^{(j)} \)\(^{-1} \)

8: The sub-optimal MAP estimate of \( D_n \) and \( g_n \) is obtained according to (48) and (49).

E. Low-complexity implementation: Truncated prior

Table I provides the computational complexity required for performing each step of the proposed MST algorithm. We observe that most of the complexity of the proposed algorithm is due to the Gibbs sampling step, which calculates (33) for all \( M \) possible values of \( d_{n,k} \). To reduce the complexity, we can truncate the tail of the prior density \( P(d_{n,k}|d_{n-1,k}) \). Specifically, the density \( P(d_{n,k}|d_{n-1,k}) \) is forced to 0 over the range of \( d_{n,k} \) where \( P(d_{n,k}|d_{n-1,k}) < \text{smaller than the predefined threshold} \). This approximation is justified by the fact that the tail of the prior distribution \( P(d_{n,k}|d_{n-1,k}) \) is expected to decay rapidly toward zero as \( |d_{n,k} - d_{n-1,k}| \) increases. If the threshold is small enough, the performance loss due to the approximation would be negligible. Note that the complexity of our MST algorithm is reduced by a factor of \( r/M \), where \( r \) is the number of nonzero discrete bins remaining after the truncation of the density \( P(d_{n,k}|d_{n-1,k}) \). As the value of \( M \) tends to be large in typical recovery problems, the effect of complexity reduction is expected to be substantial.

IV. SIMULATIONS

In this section, we evaluate the performance of the proposed MST algorithm. First, we validate our algorithm using the synthetic data generated by our model. Next, we evaluate the performance in application to the channel estimation in millimeter wave (mmWave)-band communications.

A. Benchmark

We compare our algorithm with the conventional sparse estimation algorithms. To establish a baseline, we include the algorithms that use only the current observation vector including OMP [6] and BPDN [9]. We also compare our algorithm with the state-of-the-art algorithms using multiple measurement vectors, including dynamic regularized modified BPDN (reg-mod-BPDN) [26], streaming modified weighted-\( \ell_1 \) (streaming mod-wl1) [27], and Kalman filtered modified-CS (KF-ModCS) [16]. Note that these algorithms were originally
developed for Scenario III but have been recently modified to handle Scenario IV in [25].

To establish a performance metric, we measure the normalized mean square error (MSE) defined as

\[
\text{MSE} = 10 \log_{10} \frac{E[||s_n - \hat{s}_n||^2]}{E[||s_n||^2]},
\]

(50)

where \( \hat{s}_n \) is the estimate of \( s_n \). The algorithms described in [16], [26], [27] do not provide an explicit estimate of the support set \( D_n \). In this case, we obtain the estimate of the support by thresholding the estimated signal \( \hat{s}_n \), where the threshold is calculated as suggested in [25].

We also measure the support recovery rate of the algorithms by calculating the empirical frequency of the exact matching between the true support set and the estimated set.

B. Experiments with synthetic data

1) Simulation setup: The synthetic data are generated based on our system model. The parameters for the prior dynamic model are given by \( F_n = \beta I \), \( V_n = \sqrt{1 - \beta^2} I \), and \( \beta = 0.9 \). The noise covariance matrix for the measurement model is given by \( V_n = \sigma_n^2 I \), where \( \sigma_n^2 \) is chosen for the given signal-to-noise ratio (SNR). The parameters \( M \) and \( N \) are set to 200 and 50, respectively. Each element of the sensing matrix \( B_n \) is randomly generated according to \( \mathcal{CN}(0, 1) \). Note that \( \hat{g}_{0|0} \) and \( P_{0|0} \) are initialized by the zero vector and the identity matrix respectively. The total number of samples generated by the Gibbs sampler is set to \( N_s = 40 \). For the proposed method, we employ the truncated prior with the threshold \( th = 10^{-4} \). In the reg-mod-BPDN, streaming mod-wl1, and KF-ModCS, the regularization parameters are set to \( \gamma = \sigma_n \sqrt{\log N} \) and \( \lambda = 0.001 \). As suggested in [25], the threshold \( \eta \) used to obtain the support estimate is set to 0.05. We generate 5000 measurement vectors and evaluate the performance metrics for all algorithms under consideration.

2) Simulation results: Fig. 2 shows the support recovery rate as a function of the sparsity level \( K \) when the SNR is set to 30 dB. The prior decaying parameter \( \alpha \) is set to 0.05, 0.1, and 0.2, in Fig. 2 (a), (b), and (c), respectively. A smaller \( \alpha \) implies slower variation in the support set. We observe that

Fig. 2. Support recovery rate versus the sparsity level \( K \) with (a) \( \alpha = 0.05 \), (b) \( \alpha = 0.1 \), and (c) \( \alpha = 0.2 \). The performance is evaluated using synthetic data generated based on our system model.
the proposed method consistently achieves a higher support recovery rate over the whole range of sparsity level $K$. Since OMP and BPDN rely on only a single measurement, they do not perform as well as the other algorithms. Though reg-mod-BPDN, streaming mod-wl, and KF-modCS exploit the temporal correlation over the multiple measurement vectors, they do not use the explicit probabilistic model for the time-varying support; thus their performance is inferior to that of our algorithm. As $\alpha$ increases, the performance gap between other algorithms decreases but our algorithm still significantly outperforms the other algorithms.

Fig. 3 shows the MSE performance of the algorithms when the sparsity level $K$ is set to 15. Note that the proposed method achieves a significantly lower MSE than the other algorithms over the entire SNR range of interest. With $\alpha = 0.2$, the proposed method yields a performance gain of more than 5 dB to achieve an MSE of $-40$ dB. This implies that the proposed method better estimates the sparse signal when the support and amplitude are simultaneously time-varying. Note that as the support set changes faster with higher values of $\alpha$, the performance gap between the proposed method and other algorithms increases.

Next, we study the effect of the truncated prior on the performance of the MST algorithm. Table II shows how the MSE performance and computational complexity (FLOPs) change for different values of the truncation parameter $r$. We set $K = 15$ and $\alpha = 0.2$, and SNR to 30 dB. The threshold is chosen such that the truncated prior density has $r$ nonzero bins. Clearly, as $r$ decreases, the computational complexity decreases at the cost of performance loss. Table II shows that the truncated prior with $r = 4$ offers almost a fivefold reduction in computational complexity with negligible performance loss.

We also analyze how the performance of our scheme is affected by the number of samples $N_s$ generated by the Gibbs sampler. Fig. 4 shows that the MSE performance improves as the number of samples increases. As long as the number of samples is above 40, the MSE performance does not degrade any further. This implies that only 40 samples are sufficient to approximate the posterior distribution of the support set for the setup considered.
TABLE II
MSE PERFORMANCE AND COMPUTATIONAL COMPLEXITY WITH VERSUS WITHOUT THE TRUNCATED PRIOR.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MSE (dB)</th>
<th>Complexity (Mega FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed MST with truncated prior ( (r=2) )</td>
<td>-34.62</td>
<td>603</td>
</tr>
<tr>
<td>Proposed MST with truncated prior ( (r=4) )</td>
<td>-34.31</td>
<td>904</td>
</tr>
<tr>
<td>Proposed MST with truncated prior ( (r=8) )</td>
<td>-34.60</td>
<td>1,328</td>
</tr>
<tr>
<td>Proposed MST with truncated prior ( (r=16) )</td>
<td>-34.62</td>
<td>2,176</td>
</tr>
<tr>
<td>Proposed MST without truncated prior</td>
<td>-34.62</td>
<td>5,673</td>
</tr>
</tbody>
</table>

C. mmWave channel estimation

In this subsection, we evaluate our algorithm in application to the channel estimation problem of mmWave-band communication systems.

1) System description: The mmWave-band communication is considered as a means to meet the ever-increasing throughput demand of the next-generation communication systems. As there exists a large amount of bandwidth that has not been explored by the current communication systems, mmWave communications can be a promising solution to support the required high data throughput. Compared to the conventional communication systems using microwave radio waves, the mmWave band experiences high atmospheric attenuation when a transmit signal is absorbed by gas and humidity. Additionally, significant path loss would occur when the signal is blocked by obstacles such as buildings, foliage, and a user’s body. To overcome this problem, beamforming techniques are widely used, where two communication entities transmit and receive the signals with appropriately adjusted phase and amplitude using an array of antennas. Suppose that the transmitter and receiver are equipped with \( N_t \) and \( N_r \) antennas, respectively. To perform beamforming, the channel-state information should be known, which is estimated through the beam training step. In this step, the transmitter transmits the known pilot symbols using the \( M_t \) transmit (Tx) beams and the receiver receives them using the \( M_r \) receive (Rx) beams. Based on the \( N_h = M_t M_r \) measurements obtained using the combinations of \( M_t \) Tx and \( M_r \) Rx beams, the receiver estimates the channel. Fig. 5 depicts the frame structure for the beam training. The duration in which the pilot symbols are transmitted is called a beam training block. As the mmWave channel changes in time, the beam training block appears repeatedly every \( N_p \) symbols to update the channel estimate periodically. Let us denote \( M_t \) \( N_t \times 1 \) beamforming vectors as \( \{ f_1, ..., f_{M_t} \} \) and \( M_r \) \( N_r \times 1 \) combining vectors as \( \{ c_1, ..., c_{M_r} \} \). When the \( j \)th beamforming vector \( f_j \) is used for the transmitter and the \( j \)th beamforming vector \( c_j \) is used for the receiver during the \( n \)th beam training block, the received signal is expressed as

\[
r_n^{(j,i)} = c_j^H H_n f_i x_n^{(i,j)} + e_j^H n_n^{(i,j)},
\]

where \( H_n \) is the \( N_r \times N_r \) channel matrix, \( x_n^{(i,j)} \) is the known symbol, and \( n_n^{(i,j)} \) is the \( N_r \times 1 \) noise vector. For convenience, we let \( x_n^{(i,j)} = 1 \). We assume that the channel \( H_n \) does not change within a beam training block but does so across different blocks. The mmWave channel matrix \( H_n \) can be represented in the discrete angular domain

\[
H_n = A^{(r)} H_n^{(v)} \left( A^{(t)} \right)^H,
\]

where

\[
A^{(r)} = \left[ a^{(r)}(-1), a^{(r)} \left( -1 + \frac{2}{G_r} \right), ..., a^{(r)} \left( -1 + \frac{2 G_r - 1}{G_r} \right) \right],
\]

\[
A^{(t)} = \left[ a^{(t)}(-1), a^{(t)} \left( -1 + \frac{2}{G_t} \right), ..., a^{(t)} \left( -1 + \frac{2 G_t - 1}{G_t} \right) \right],
\]

and

\[
a^{(r)}(\theta) = \frac{1}{\sqrt{G_r}} \left[ 1, e^{\frac{2\pi d_r \theta}{\lambda}}, e^{\frac{2\pi 2d_r \theta}{\lambda}}, ..., e^{\frac{2\pi (N_r - 1)d_r \theta}{\lambda}} \right]^T,
\]

\[
a^{(t)}(\theta) = \frac{1}{\sqrt{G_t}} \left[ 1, e^{\frac{2\pi d_t \theta}{\lambda}}, e^{\frac{2\pi 2d_t \theta}{\lambda}}, ..., e^{\frac{2\pi (N_t - 1)d_t \theta}{\lambda}} \right]^T,
\]

where \( G_r \) and \( G_t \) are the number of bins of a uniformly quantized angular grid. In addition, \( d_r \) and \( d_t \) are the distances between the adjacent antennas for the receiver and transmitter, respectively, and \( \lambda \) is the signal wavelength. The mmWave channel tends to have only a few dominant components in the angular domain, which implies that most channel gains in \( H_n^{(v)} \) are close to zero. Hence, we can assume that most elements of \( H_n^{(v)} \) are zero except for a few ones. We construct the received measurement vector as

\[
y_n = \left[ r_n^{(1,1)}, ..., r_n^{(M_r,1)}, ..., r_n^{(1,M_t)}, ..., r_n^{(M_r,M_t)} \right]^T.
\]
the vectorization operation, \( A \) where \( H \) of the sparse vector that the channel estimation boils down to the estimation of two matrices \( y \). Fig. 5. Frame structure of beam training.

Then, we can show that [35]

\[
y_n = B_n s_n + w_n, \quad \text{where} \quad ()^T \text{denotes the conjugation operation, vec()} \text{denotes the vectorization operation, } A \otimes B \text{ is the Kronecker product of two matrices } A \text{ and } B, \text{ and } w_n \text{ is the noise vector. Note that the channel estimation boils down to the estimation of the sparse vector } s_n \text{ based on the measurement vectors } y_1, ..., y_n. \text{Moreover, the support set of } s_n \text{ is determined by the nonzero channel gains in } H_n^{(v)}. \text{As the row vectors of } H_n^{(v)} \text{represent the channel gains in the angle of arrival (AoA) and the column vectors represent those in the angle of departure (AoD), the support set } D_n \text{ of } s_n \text{ is determined by the support set associated with AoA, } D_n^{AoA} \text{ and that associated with AoD, } D_n^{AoD}. \text{While the transmitter or the receiver is under mobility, the support sets, } D_n^{AoA} \text{ and } D_n^{AoD} \text{change across different beam training blocks. Owing to the physical constraint of the transmitter and the receiver’s motion, } D_n^{AoA} \text{ and } D_n^{AoD} \text{ would change slowly with } n \text{ with high temporal correlation. Thus, we can describe the temporal evolution of } D_n^{AoA} \text{ and } D_n^{AoD} \text{using our prior dynamic sparsity model in (14) as}
\]

\[
\Pr(D_n|D_{n-1}) = \prod_{k=1}^{K} \Pr(d_{n,k}^{AoD}|d_{n-1,k}^{AoD}) \prod_{k=1}^{K} \Pr(d_{n,k}^{AoA}|d_{n-1,k}^{AoA}).
\]

We assume that the channel gains associated with different paths are independent and \( D_n^{AoA} \) and \( D_n^{AoD} \) are independent with each other. We also assume that each channel gain follow the exponentially decaying conditional distribution, i.e.,

\[
\Pr(d_{n,k}^{AoD} = m_j|d_{n-1,k}^{AoD} = m_i) = \frac{1}{C} \alpha^{m_j - m_i}, \quad \text{(58)}
\]

\[
\Pr(d_{n,k}^{AoA} = m_j|d_{n-1,k}^{AoA} = m_i) = \frac{1}{C} \alpha^{m_j - m_i}. \quad \text{(59)}
\]

The exponent \( \alpha \) can be obtained from the empirical data samples based on the maximum likelihood criterion. The proposed MST method is employed to estimate the sparse vector \( s_n \) based on the multiple measurement vectors \( y_1, ..., y_n \).

2) Simulation setup: Next, we provide the system parameters used in the simulations. We set the number of antennas to 16 for both the transmitter and receiver, i.e., \( N_t = N_r = 16 \). We consider uniform linear arrays of antennas. The channel gains are generated from the auto-regressive model in (15) with the parameters \( F_n = \beta I, V_n = \sqrt{1 - \beta^2} I \), and \( \beta = 0.5 \). The resolutions for AoA and AoD are set to \( G_1 = G_r = 32 \) and thus the dimension \( M \) of \( s_n \) becomes 1024. The numbers of Tx and Rx beams used for beam training are set to \( M_t = M_r = 16 \). Thus, the dimension of \( y_n \) becomes \( N = M_t \times M_r = 256 \). The beam transmission period \( N_p \) is set to 1000 symbols (see Fig. 5). The symbol duration is set to 4.46\( \mu \)s following the 5G NR standard [36]. We assume that the UE moves at a speed of 5.2 m/s and the channel is generated using the Ray tracing simulator [37]. The prior decaying parameter \( \alpha \) is set to 0.2, which is empirically shown to describe the time-varying mmWave channel well. The total number of particles used for the proposed method is set to \( N_s = 40 \). We compare the proposed MST algorithm with the conventional sparse estimation schemes including OMP, BPDM, Reg-mod-BPDM, Streaming-mod-w11, and KF-ModCS. We measure both channel estimation MSE and bit error rate (BER) of the mmWave system to evaluate the performance of the proposed MST algorithm. We generate the data symbols using the binary phase shift keying (BPSK) modulation, considering the single-layer multi-input multi-output (MIMO) transmission. The Tx and Rx beamforming vectors for data transmission are given by the right and left singular vectors of the channel matrix \( H_n \) associated with the largest singular value [38]. The maximum likelihood detector is used to demodulate the data symbols.

3) Simulation result: Fig. 6 shows the MSE performance as a function of SNR for several channel estimation algorithms under consideration. The number of paths \( K \) is set to \( K = 3 \) and \( K = 5 \) in Fig. 6 (a) and (b), respectively. We observe that the proposed MST method yields a performance gain of more than 2 dB over the other methods to achieve an MSE of \(-40 \text{ dB}\). As the number of paths, \( K \) increases from 3 to 5, the performance gap between our algorithm with other methods increases, especially in a high-SNR regime. Fig. 7 also shows the BER performance of the proposed MST algorithm. Note that the performance gain of the proposed MST algorithm over the other algorithms is also validated in terms of BER performance. This shows that the proposed method offers promising channel estimation performance, particularly for high-mobility scenarios.
V. CONCLUSIONS

In this paper, we addressed the problem of tracking the time-varying support of sparse signals. The variations of the support and amplitude of the sparse signal were described by our dynamic prior model. A sequential Monte-Carlo estimation of the support and the amplitude was derived. We applied Rao-Blackwellization to track both support and amplitude separately in a computationally efficient manner. The samples for the support set were generated using the Gibbs sampler and the amplitude variable was linearly estimated using the Kalman filter for all generated samples. In combination with some low-complexity implementation, our method successfully tracked the sparse signals with the time-varying support at a reasonable complexity. The experimental results showed that our method significantly outperforms the existing sparse recovery algorithms using multiple measurement vectors.

APPENDIX A
DERIVATION OF (27)

Using Bayes’ rule, We factorize (27) as

\[
P(d_{n,k} | D_n \setminus k, D_{0:n-1}, y_{1:n}) = P(y_n | d_{n,k}, D_{0:n-1}, y_{1:n-1}) \times P(d_{n,k} | c_{n-1,k})
\]

(61)
APPENDIX B

DERIVATION OF (28)

First, we can show that

\[
P(y_n | D_{0:n}, y_{1:n-1}) = \int P(y_n, g_n | D_{0:n}, y_{1:n-1}) dg_n
\]

\[
= \int P(y_n | g_n, D_{0:n}) \times P(g_n | D_{0:n}, y_{1:n-1}) dg_n.
\]

(63)

The first term on the right side of (63) is given by

\[
P(y_n | g_n, D_{0:n}) = \mathcal{C}N(B_n \Lambda(D_n) g_n, W_n),
\]

and the second term is expressed as [39]

\[
P(g_n | D_{0:n}, y_{1:n-1}) = \mathcal{C}N(\hat{g}_n | 1:n-1, P_n | 1:n-1).
\]

(65)

Using (64) and (65), the integration in (63) can be obtained in a closed form as

\[
P(y_n | D_{0:n}, y_{1:n-1}) = \int \mathcal{C}N(B_n \Lambda(D_n) g_n, W_n) \times \mathcal{C}N(\hat{g}_n | 1:n-1, P_n | 1:n-1) dg_n
\]

\[
= \mathcal{C}N(B_n \Lambda(D_n) \hat{g}_n | 1:n-1, D_n),
\]

(66)

where

\[
D_n = (B_n \Lambda(D_n)) P_n | 1:n-1 (B_n \Lambda(D_n))^H + W_n.
\]

REFERENCES


