# Respiratory Motion Estimation With Hybrid Implementation of Extended Kalman Filter

Suk Jin Lee, *Student Member, IEEE*, Yuichi Motai, *Member, IEEE*, and Martin Murphy

***Abstract*—The extended Kalman filter (EKF) can be used for the purpose of training nonlinear neural networks to perform desired input–output mappings. To improve the computational requirements of the EKF, Puskorius *et al.* proposed the decoupled EKF (DEKF) as a practical remedy for the proper management of computational resources. This approach, however, sacrifices computational accuracy of estimates because it ignores the interactions between the estimates of mutually exclusive weights. To overcome such a limitation, therefore, we proposed hybrid implementation based on EKF (HEKF) for respiratory motion estimation, which uses the channel number for the mutually exclusive groups and the coupling technique to compensate the computational accuracy. Moreover, the authors restricted to a DEKF algorithm in which the weights connecting the inputs to a node are grouped together. If there are multiple input training sequences with respect to the time stamp, the complexity can increase by the power of the input channel number. To improve the computational complexity, we split the complicated neural network into a couple of simple neural networks to adjust separate input channels. The experimental results validated that the prediction overshoot of the proposed HEKF was improved by 62.95% in the average prediction overshoot values. The proposed HEKF showed a better performance of 52.40% improvement in the average of the prediction time horizon. We have evaluated that the proposed HEKF can outperform DEKF by comparing the prediction overshoot values, the performance of the tracking estimation value, and the normalized root-mean-squared error.**

***Index Terms*—Estimate, extended Kalman filter (EKF), multilayer perceptron (MLP), recurrent neural network (RNN), tracking.**

## I. Introduction

THE problem of predicting moving objects with a given reference trajectory is a common estimation problem [1]–[5]. Kalman filters can be widely used in many industrial electronics for state estimation and prediction [6]–[14]. Due to increasingly complex dynamical systems, a variety of methodologies has been proposed based on the Kalman filter and its hybrid approach [14]–[19]. The recurrent neural network (RNN) can also be one of the estimation methods for predictive control in many application systems [20]–[32]. Here, RNN is a class of neural network where connections between units exhibit dynamic temporal behavior with their synaptic weights. Owing to this dynamic behavior, RNN can implement dynamical nonlinear multivariable discrete-time systems of arbitrary complexity [33]–[36].

Target-tracking estimation can be one of the applications of RNN because of its adaptive learning, which is an ability to learn how to do tasks based on the data given for training or initial experience [20], [21], [25], [26], [64]. For example, RNN can be used for respiratory motion prediction for real-time motion adaptation in the medical application [37]–[42]. Because of the self-organized characteristic of neural networks, it can have a built-in capability to adapt their synaptic weights to change based on the given samples in the specific circumstance. Thus, it can provide a better performance in comparison to the conventional methods of respiratory motion prediction [43]–[47]. Intrinsically, a training algorithm for RNN became an issue in improving the performance of dynamical systems with respect to the specific environment [48].

There are several algorithms available in training the weights of recurrent networks based on streams of input–output data. Basically, the most widely used are the back-propagation-through-time algorithm [49]–[51] and the real-time recurrent learning (RTRL) algorithm [51]–[54], which are both based on the computation of the gradient of an output error measure with respect to network weights. However, the calculation of the dynamic derivatives of the recurrent network's outputs with respect to its weights by RTRL is computationally expensive since these derivatives cannot be computed by the same back-propagation mechanism that was employed in the training of multilayer perceptron (MLP) networks [55].

As an alternative or improvement of the gradient descent-based methodology, several authors have noted that the extended Kalman filter (EKF) can also be used for the purpose of training networks to perform the desired input–output mappings [15]–[19]. Note that the predictor–corrector property is an intrinsic property of the Kalman filter, its variants, and extensions. Thus, whereas, in traditional applications of the Kalman filter for sequential state estimation, the roles of predictor and corrector are embodied in the Kalman filter itself, in supervised-training applications, these two roles are split between the RNN and the EKF. Here, the RNN, in which the input training samples are applied to the recurrent MLP (RMLP) as the excitation, performs the role of the predictor, and the EKF, in which the training samples of the desired response are applied

Manuscript received December 21, 2010; revised March 20, 2011; accepted May 5, 2011. Date of publication May 27, 2011; date of current version June 19, 2012. This work was supported in part by the Dean's Office of the School of Engineering, Virginia Commonwealth University, and in part by NSF ECCS under Grant 1054333.

S. J. Lee and Y. Motai are with the Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: leesj9@mymail.vcu.edu; ymotai@vcu.edu).

M. Murphy is with the Department of Radiation Oncology, Virginia Commonwealth University, Richmond, VA 23298 USA (e-mail: mjmurphy@vcu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2011.2158046

to the EKF as the observable data in providing the supervision, performs the role of the corrector [55].

In comparison to the gradient descent algorithms, EKF-based algorithms for recurrent networks do not require batch processing, making them more suitable for online use. To improve the computational requirements of the EKF, Puskorius *et al.* proposed decoupled EKF (DEKF) as a practical remedy for the proper management of computational resources [15]. The author in [15] restricted to a DEKF algorithm in which the weights connecting the inputs to a node are grouped together. *This approach, however, sacrifices computational complexity and estimation accuracy since DEKF defines a node as the mutually exclusive weight group. If there are multiple input training sequences with respect to the time stamp, the complexity can increase by the power of the input channel number*. To overcome these limitations, we do not adopt the mutually exclusive weight groups. Instead, we adopt the channel number for the mutually exclusive groups to propose the coupling technique to compensate the computational accuracy using multiple sensory channel inputs. We call this newly proposed method as hybrid motion estimation based on EKF (HEKF).

The contribution of this paper is twofold. First, we propose a new approach to split the whole RMLP with the complicated neuron number into a couple of RMLPs with the simple neuron number to adjust separate input channels. Second, we present a new method for respiratory motion estimation using EKF, which adapts the coupling technique using multiple channel inputs for the mutually exclusive groups to compensate the computational accuracy, instead of mutually exclusive weight groups.

This paper is organized as follows. In Section II, the theoretical background of the proposed algorithm is briefly discussed. In Section III, the proposed hybrid implementation based on EKF for RNN with multiple sensory channel inputs is presented in detail. Section IV presents and discusses the experimental results of the proposed filter design method—efficient estimation of the measurements, optimized group number for RMLP, prediction overshoot analysis, prediction time horizon, and computational complexity of HEKF and DEKF. A summary of the performance of the proposed method is presented in Section V.

## II. Background

### A. RNN

An RNN is a class of neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. A network with a rich representation of past outputs is a fully connected RNN, known as the Williams–Zipser network, as shown in Fig. 1[48]. This network consists of three layers: the input, processing, and output layers. For each neuron $i$ $(i = 1, 2, \ldots, N)$, the elements $u_j$ of the input vector $(j = 1, 2, \ldots, M + N + 1)$ to a neuron $u$ are as follows:

$$u_j^T(k) = [x(k-1), \ldots, x(k-M), 1, y_1(k-1), \ldots, y_N(k-1)] \tag{1}$$

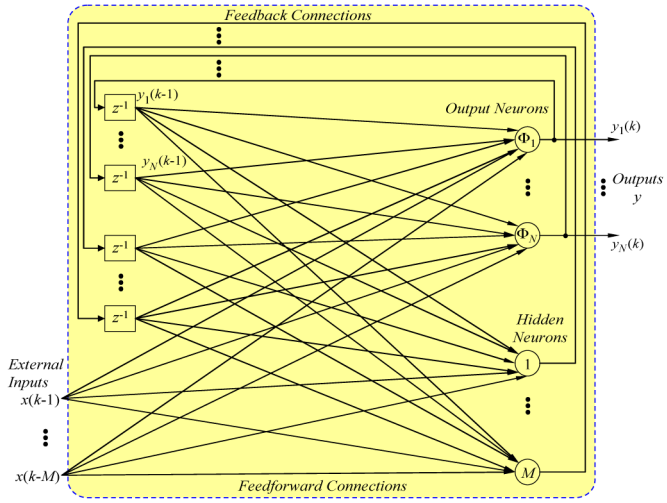


Fig. 1. Fully connected RNN with external inputs.

where $M$ is the number of external inputs, $N$ is the number of feedback connections, $(\cdot)^T$ denotes the vector transpose operation, and the $(M + N + 1) \times 1$ dimensional vector $u$ comprises both the external and feedback inputs to a neuron, as well as the unity valued constant bias input. Equation (1) is weighted and then summed to produce an internal activation function of a neuron $v$ as follows:

$$v_i(k) = \sum_{l=1}^{M+N+1} w_{i,l}(k) u_l(k) \tag{2}$$

where $w$ represents weights. Finally, (2) is fed through a nonlinear activation function $\Phi$ to form the output of the $i$th neuron $y_i$. Here, the function $\Phi$ is a monotonically increasing sigmoid function with slope $\beta$, as, for instance, the logistic function

$$\Phi(v) = \frac{1}{1 + e^{-\beta v}}. \tag{3}$$

At the time instant $k$, for the $i$th neuron, its weights form an $(M + N + 1) \times 1$ dimensional weight vector $w_i^T(k) = [w_{i,1}(k), \ldots, w_{i,M+N+1}(k)]$. One additional element of the weight vector $w$ is the bias input weight. After feeding (2) into (3) using the function $\Phi$, the output of the $i$th neuron $y_i$ can be formed as follows:

$$y_i(k) = \Phi\left(v_i(k)\right), \qquad i = 1, 2, \ldots, N. \tag{4}$$

In an RNN architecture, the feedback brings the delayed outputs from hidden and output neurons back into the network input vector $u(k)$, as shown in Fig. 1. Due to this recursive function at each time instant, the network is presented with the raw possibly noisy external input data $x(k), x(k-1), \ldots, x(k-M)$ from Fig. 1, input elements (1) and the filtered data $y_1(k-1), \ldots, y_N(k-1)$ from the network output. Intuitively, this filtered input history helps in improving the processing performance of RNNs, as compared with feedforward networks. Therefore, an RNN should be able to process signals corrupted by additive noise even in the case when the noise distribution is varying over time.

### *B. EKF for RNNs (EKF-RNN)*

As mentioned in the previous section, the learning algorithm based on gradient descent, exemplified by the RTRL algorithm, is typically slow due to reliance on instantaneous estimates of gradients [15]. We can overcome this serious limitation by using the supervised training of a recurrent network which recursively utilizes information contained in the training data in a manner going back to the first iteration of the learning process. That is based on Kalman filter theory [55].

Consider a recurrent network built around a static MLP with $s$ weights and $p$ output nodes. Let the vectors $w(k)$, $v(k)$, and $u(k)$ denote the weights of the entire network, the recurrent activities inside the network, and the input signal applied to the network at time $k$, respectively. With adaptive filtering in mind, the system state model and measurement model equations for the network may be modeled as follows:

$$w(k+1) = w(k) + q(k) \tag{5}$$

$$d(k) = b\left(w(k), v(k), u(k)\right) + r(k) \tag{6}$$

where $q(k)$ and $r(k)$ are the process and measurement noise with the property of a multivariate zero-mean white noise with covariance matrices $Q$ and $R$, respectively. $d(k)$ is the observable data, and $b(\cdot,\cdot,\cdot)$ is the measurement function that accounts for the overall nonlinearity of the MLP from the input to the output layer.

For us to be able to apply the EKF algorithms as the facilitator of the supervised-learning task, we have to linearize the measurement equation (6) by retaining the first-order terms in the Taylor series expansion of the nonlinear part of the equation. With $b(w(k), v(k), u(k))$ as the only source of nonlinearity, we may approximate (6) as follows:

$$d(k) = B(k)w(k) + r(k) \tag{7}$$

where $B(k)$ is the $p \times s$ measurement matrix of the linearized model. The linearization consists of the partial derivatives of the $p$ outputs of the whole network with respect to the $s$ weights of the model as shown

$$B(k) = \begin{bmatrix} \frac{\partial b_1}{\partial w_1} & \frac{\partial b_1}{\partial w_2} & \cdots & \frac{\partial b_1}{\partial w_s} \\ \frac{\partial b_2}{\partial w_1} & \frac{\partial b_2}{\partial w_2} & \cdots & \frac{\partial b_2}{\partial w_s} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial b_p}{\partial w_1} & \frac{\partial b_p}{\partial w_2} & \cdots & \frac{\partial b_p}{\partial w_s} \end{bmatrix}. \tag{8}$$

The partial derivatives in (8) are evaluated at $w(k) = \hat{w}(k|k-1)$, where $\hat{w}(k|k-1)$ is the prediction of the weight vector $w(k)$ computed by EKF at time $k$, given the observed data up to time $k-1$.

For the purpose of our present discussion, the relevant equations in the EKF algorithm are the innovation process and the weight update equations as follows:

$$\alpha(k) = d(k) - b(k)\left(\hat{w}(k|k-1), v(k), u(k)\right) \tag{9}$$

$$\hat{w}(k+1|k) = \hat{w}(k|k-1) + G(k)\alpha(k) \tag{10}$$

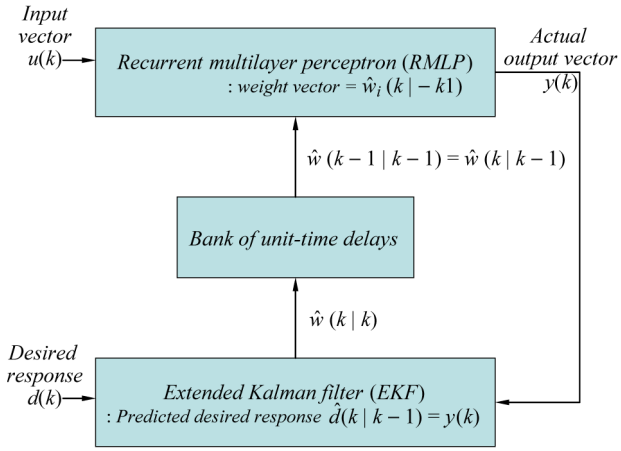


Fig. 2. Closed-loop feedback system embodying the RMLP and the EKF.

where $\alpha(k)$ is the $p \times 1$ matrix denoting the innovations defined as the difference between the desired response $d(k)$ for the linearized system and its estimation, $\hat{w}(k|k-1)$ is the $s \times 1$ vector denoting the estimate of the weight vector $w(k)$ at time $k$ given the observed data up to time $k-1$, and $\hat{w}(k|k)(= \hat{w}(k+1|k))$ is the filtered updated estimate of $w(k)$ on the receipt of the observable $d(k)$. $G(k)$ is the $s \times p$ matrix denoting the Kalman gain that is an integral part of the EKF algorithm.

Let $\Gamma(k)$, $P(k|k-1)$, and $P(k|k)$ be defined as the $p \times p$ matrix denoting the global conversion factor for the entire network, the $s \times s$ prediction-error covariance matrix, and the $s \times s$ filtering-error covariance matrix, respectively. In light of these new notations, we can write the EKF algorithms as follows:

$$\Gamma(k) = \left[B(k)P(k|k-1)B^T(k) + R(k)\right]^{-1} \tag{11}$$

$$G(k) = P(k|k-1)B^T(k)\Gamma(k) \tag{12}$$

$$P(k|k) = P(k|k-1) - G(k)B(k)P(k|k-1) \tag{13}$$

$$P(k+1|k) = P(k|k) + Q(k). \tag{14}$$

As shown in Fig. 2, with the weight vector set at its old predicted value $\hat{w}(k|k-1)$, the RMLP computes the actual output vector $y(k)$ in response to the input vector $u(k)$. After updating the old estimate of the weight vector by operating on the current desired response $d(k)$, the filtered estimate of the weight vector $\hat{w}(k|k)$ is computed in accordance with (10).

Note that, in the EKF-RNN of Fig. 2, the RNN performs the role of the predictor, and the EKF performs the role of the corrector.

## III. Multichannel Coupled EKF-RNN: Proposed Filter Design For Multiple Sensors

### *A. DEKF*

The computational requirement of the EKF is dominated by the need to store and update the filtering-error covariance matrix $P(k|k)$ at time step $k$. For an RNN containing $p$ output nodes and $s$ weights, the computational complexity of the EKF is $O(ps^2)$, and its storage requirement is $O(s^2)$. For a large $s$, these requirements may be highly demanding. In such

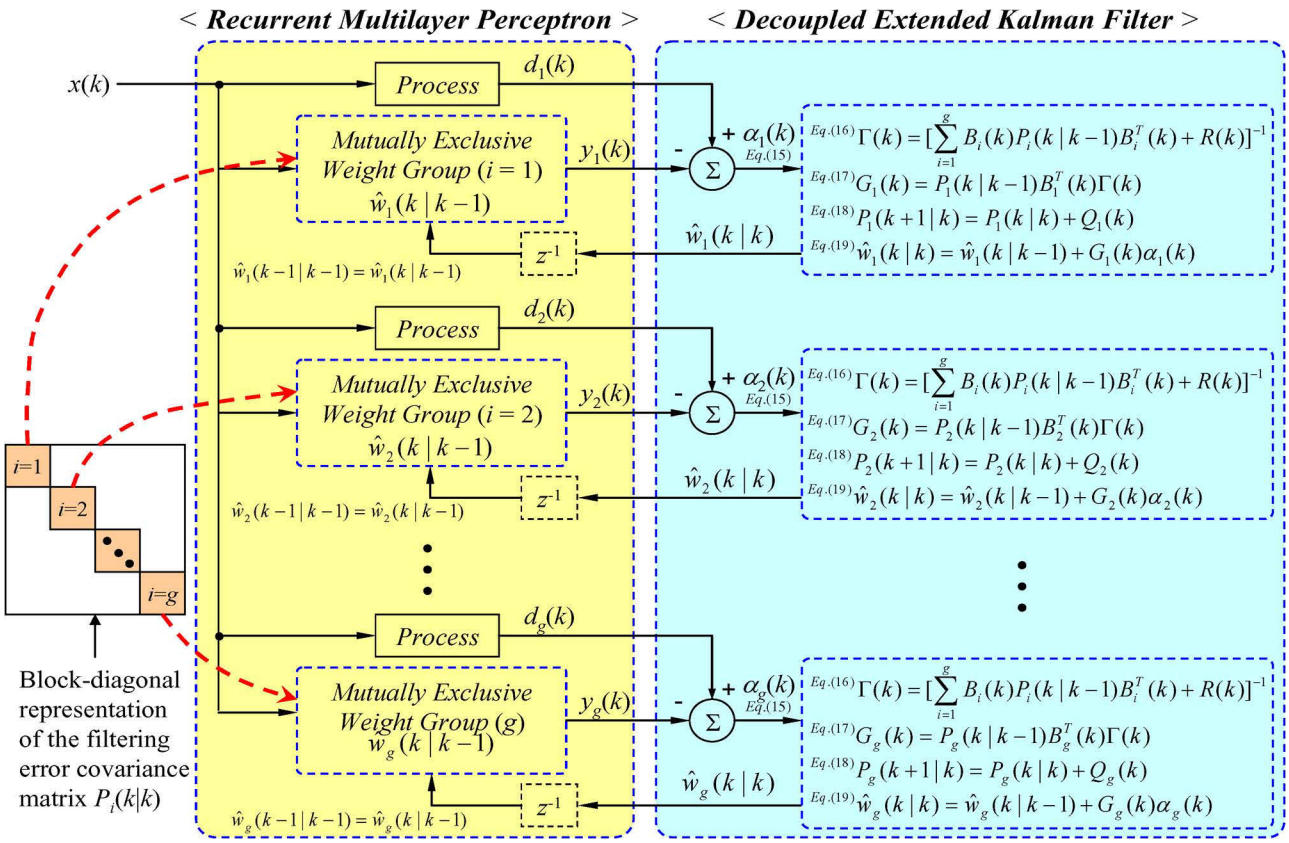


Fig. 3. DEKF for RNN. Each group is corresponding to mutually exclusive weight group. The concatenation of the filtered weight vector $\hat{w}_i(k|k)$ forms the overall filtered weight vector $\hat{w}(k|k)$.

situations, we need to look for a practical remedy for the proper management of computational resources, i.e., DEKF [15], [55].

The basic idea behind the DEKF is to ignore the interactions between the estimates of certain weights in the RNN. If the weights in the network are decoupled in such a way that we can create *mutually exclusive weight groups*, then the covariance matrix $P(k|k)$ is structured into a block-diagonal form, as shown at the bottom left of Fig. 3.

Let $g$ denote the designated number of mutually exclusive disjoint weight groups. Also, for $i = 1, 2, \ldots, g$, let $\hat{w}_i(k|k)$, $P_i(k|k)$, and $G_i(k)$ be defined as the filtered weight vector, the subset of the filtering-error covariance matrix, and the Kalman gain matrix for the group $i$, respectively. The concatenation of the filtered weight vectors $\hat{w}_i(k|k)$ forms the overall filtered weight vector $\hat{w}(k|k)$. In light of these new notations, we can now rewrite the DEKF algorithm for the $i$th weight group as follows:

$$\alpha_i(k) = d_i(k) - b_i(k) \left(\hat{w}_i(k|k-1), v_i(k), u_i(k)\right) \tag{15}$$

$$\Gamma(k) = \left[\sum_{i=1}^{g} B_i(k) P_i(k|k-1) \left(B_i(k)\right)^T + R(k)\right]^{-1} \tag{16}$$

$$G_i(k) = P_i(k|k-1) \left(B_i(k)\right)^T \Gamma(k) \tag{17}$$

$$\hat{w}_i(k+1|k) = \hat{w}_i(k|k-1) + G_i(k)\alpha_i(k) \tag{18}$$

$$P_i(k+1|k) = P_i(k|k) + Q_i(k) \tag{19}$$

$$P_i(k|k) = P_i(k|k-1) - G_i(k) B_i(k) P_i(k|k-1) \tag{20}$$

where $\alpha_i(k)$, $\Gamma(k)$, and $P_i(k+1|k)$ denote the difference between the desired response $d_i(k)$ for the linearized system and its estimation for the $i$th weight group, the global conversion factor for the entire network, and the prediction-error covariance matrix, respectively.

DEKF can reduce the computational complexity and its storage requirement of the EKF, but [15] restricts to a DEKF algorithm for which the weights are grouped by node. That sacrifices the computational accuracy because of omitting the interactions between the estimates of certain weights.

To verify the prediction accuracy, we used a certain marginal value that can be explained in detail in Section III-D. Fig. 4 shows the estimation of the respiratory motion with DEKF. As shown in Fig. 4, we can notice that the percentage of prediction overshoot based on the marginal value is over 35%, which means that we need a new approach to compensate the prediction accuracy with multiple input sequences. Therefore, we will show a hybrid motion estimation based on EKF (HEKF) in the next section, which uses the channel number for the mutually exclusive groups and the coupling technique to compensate the computational accuracy.

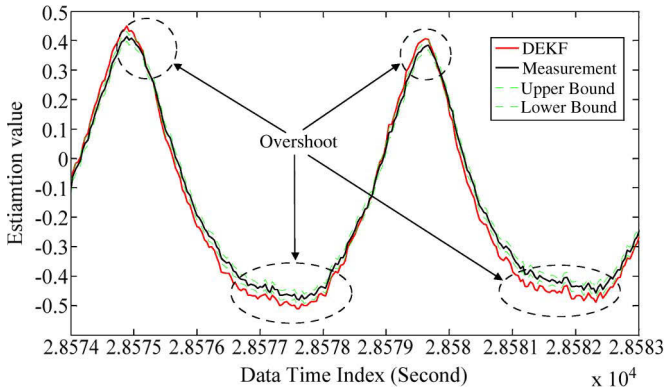


Fig. 4. Prediction overshoot with DEKF.

### *B. Hybrid Estimation Based on EKF for Neural Network (HEKF)*

We have extended the DEKF into hybrid motion estimation based on EKF (HEKF). The author in [15] restricted to a DEKF algorithm in which the weights connecting the inputs to a node are grouped together. If there are multiple input sequences with respect to time $k$, the complexity can increase by the power of the input number. To overcome computational complexity and estimation accuracy, we propose the coupling technique to compensate the computational accuracy using multiple sensory channel inputs. We refer to this newly proposed method as hybrid motion estimation based on EKF (HEKF).

There are two significant innovations for the proposed HEKF. The first innovation is to comprehensively organize the multiple channel sensory process by adapting the coupling technique. The second innovation is the multiple RMLPs with the simple neuron number for separate input channels. We first introduce the coupling matrix in (21) and then show the separate EKF process for each RMLP in (22)–(27).

Let $c$ denote the designated channel number for the mutually exclusive groups. Here, each group corresponds to an individual channel that is composed of a position vector sequence with respect to time $k$. Also, for $i = 1, 2, \ldots, c$, let $\hat{w}_i^{CP}(k|k)$ be defined as the filtered weight vector, and $P_i^{CP}(k|k)$ and $G_i^{CP}(k)$ are the subsets of the filtering-error covariance matrix and the Kalman gain matrix for the channel $i$ coupled with other channels, respectively.

Let $\Gamma^{CP}(k)$ and $P_i^{CP}(k|k-1)$ be defined as the $p \times p$ matrix denoting the global conversion factor for the coupled entire network and the $s \times s$ prediction-error covariance matrix for the coupled EKF, respectively. Here, we also need to define the degree of coupling $\mu_{ij}$ representing the degree to which component $(i)$ depends on one another $(j)$. Coupling matrix $\Pi$ is the $p \times p$ matrix containing all components of coupling degree. We can represent coupling matrix $(\Pi)$ and coupling degree $(\mu_{ij})$ as follows:

$$\Pi = \begin{bmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1p} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{p1} & \mu_{p2} & \cdots & \mu_{pp} \end{bmatrix} \quad \sum_{j=1}^{p} \mu_{ij} = 1, \text{ for all } i. \quad (21)$$

The closer to one the coupling degree is, the more tightly the channels $i$ and $j$ are coupled, i.e., *tight coupling*. If the coupling degree is close to zero, we can expect *loose coupling*. If $\mu_{ij}$ is corresponding to zero, there is no coupling with one another. For $i = 1, 2, \ldots, c$, let us define $\hat{w}_i^{CP}(k|k)$ as the filtered weight vector, and $P_i^{CP}(k|k)$ and $G_i^{CP}(k)$ are the subsets of the filtering-error covariance matrix and the Kalman gain matrix for the channel number $i$, respectively. In light of these new notations, we can write the hybrid motion estimation based on EKF (HEKF) as follows:

$$\alpha_i^{CP}(k) = d_i(k) - \left[\sum_{j=1}^{p} \mu_{ij} \times y_j\right] \quad (22)$$

$$\Gamma^{CP}(k) = \left[\sum_{i=1}^{p}\sum_{j=1}^{p} \mu_{ij} B_i(k) P_i^{CP}(k|k-1) \left(B_i(k)\right)^T + R(k)\right]^{-1} \quad (23)$$

$$G_i^{CP}(k) = P_i^{CP}(k|k-1) \left(B_i(k)\right)^T \Gamma^{CP}(k) \quad (24)$$

$$\hat{w}_i^{CP}(k+1|k) = \hat{w}_i^{CP}(k|k-1) + G_i^{CP}(k) \alpha_i^{CP}(k) \quad (25)$$

$$P_i^{CP}(k+1|k) = P_i^{CP}(k|k) + Q_i(k) \quad (26)$$

$$P_i^{CP}(k|k) = P_i^{CP}(k|k-1) - G_i^{CP}(k) B_i(k) P_i^{CP}(k|k-1) \quad (27)$$

where $\alpha_i^{CP}(k)$, $\Gamma^{CP}(k)$, and $P_i^{CP}(k+1|k)$ denote the difference between the desired response $d_i(k)$ for the linearized system and coupled estimations for the channel number $i$, the global conversion factor for the entire-coupled network, and the prediction-error covariance matrix for the coupled, respectively. In the case of HEKF, we have $c$ identical networks for $c$ input channels. Each input sequence is inserted into individual neural network process for each channel prediction, as shown in Fig. 5.

### *C. Adaptive Coupling Matrix*

At the beginning of this section, we define that the coupling matrix $\Pi$ of (21) is the $p \times p$ matrix containing all components of coupling degree. To make the time-adaptive coupling system, we need to adjust the coupling matrix $\Pi(k)$ with respect to time $k$. Here, we define $H(k)$ as the error-gain matrix as follows:

$$H(k) = \left(G^{CP}(k)\right)^T P^{CP}(k|k) G^{CP}(k) \quad (28)$$

where $G^{CP}(k)$ and $P^{CP}(k|k)$ are the $s \times p$ matrix denoting the Kalman gain and the $s \times s$ filtering-error covariance matrix, respectively. As would be expected, $H(k)$ can get the $p \times p$ matrix denoting global error gain value. Now, we can define $\Delta(k)$ as the difference of the consecutive global error gain values as follows:

$$\Delta(k) = H(k) - H(k-1). \quad (29)$$

Finally, we can adjust the adaptive coupling matrix as follows:

$$\Pi(k) = \Pi(k-1) + \Delta(k) \quad (30)$$

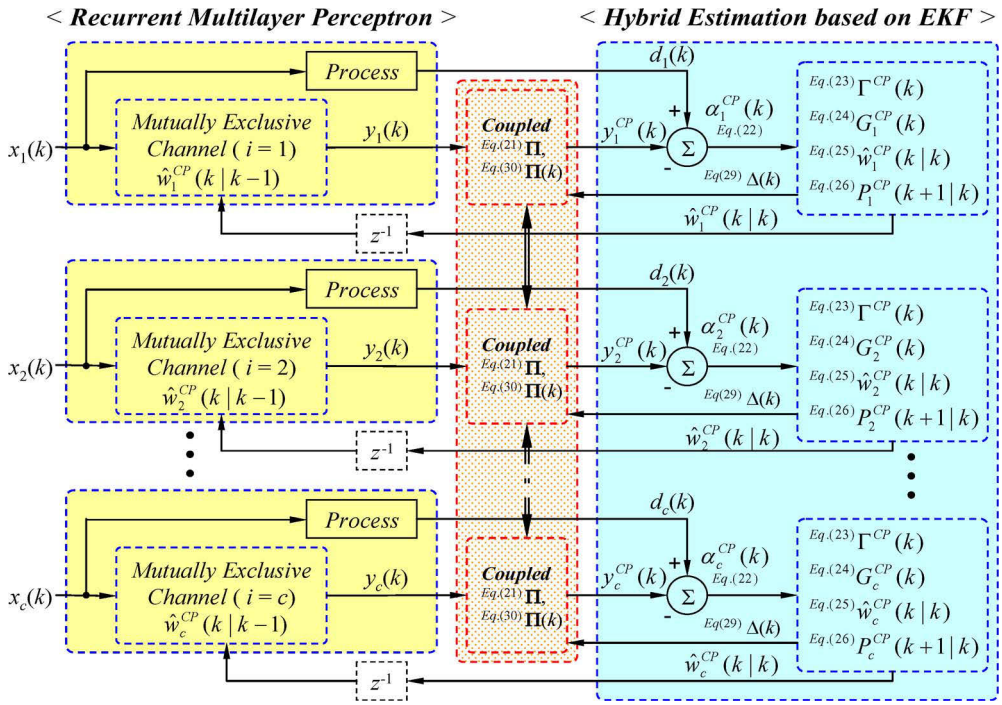


Fig. 5. Hybrid motion estimation based on EKF (HEKF) for RNN. Each group corresponds to an individual channel that is composed of $(x, y, z)$ position sequence with respect to the time step $k$.

where the current coupling matrix values can be increased or decreased depending on the difference value (29) in comparison with the previous coupling matrix values.

### D. Optimized Group Number for RMLP

In the DEKF algorithm, the weights connecting the inputs to a node are grouped together, whereas each group in the HEKF algorithm corresponds to the individual channel that is composed of a position vector sequence with respect to time $k$. In order to analyze the group number, we can incorporate Fisher linear discriminant on the discriminant analysis, which employs the *within-class scatter* value $(S_W)$ and the *between-class scatter* value $(S_B)$ in the given samples [56].

We have a set of $nD$-dimensional samples, which correspond to the filtering-error covariance matrices ($P_i$ and $P_i^{CP}$) defined in (20) and (27) for each group $i$. Let $m_i$ denote the $D$-dimensional sample mean for group $i$, and then, define $m_i$ as follows:

$$m_i = \frac{1}{n_i} \sum_{j=1}^{n_i} P_i(j) \tag{31}$$

where $n_i$ is the component number of group $i$.

To obtain the optimization objective function, we define the scatter values $S_i$ and $S_W$ as

$$S_i = \sum_{P \in P_i} (P - m_i)^2 \tag{32}$$

$$S_W = \sum_{k=1}^{g} S_k \tag{33}$$

where $g$ is the number of group in the given samples.

We define the *between-class scatter* value $S_B$ as follows:

$$S_B = \sum_{i=1}^{g} \sum_{j=1}^{g} |m_i - m_j|^2 \quad (i \neq j) \tag{34}$$

where $g$ is the number of group in the given samples and $m_i$ is not identical to $m_j$.

In terms of $S_B$ and $S_W$, the objective function $J(\cdot)$, called as the discriminant criterion, can be written as

$$J(g) = \arg\min_{g} \frac{S_W}{S_B}. \tag{35}$$

This introduced criterion expects that the *within-class scatter* value should be minimized and that the *between-class scatter* value should be maximized in the given number. Under the minimizing eq. (35), we can get the optimized number of group $(g)$ for RMLP by choosing the smallest $J(\cdot)$ with optimized group number $(g)$. This value can be used to test the optimized number of RMLP between HEKF and DEKF. We can evaluate whether HEKF or DEKF could be more discriminated by comparing the objective function values $J(\cdot)$ as the discriminant degree at the selected $(g)$.

### E. Prediction Overshoot Analysis

Here, we evaluate the performance of overshoot for the prediction values. We define overshoot for cases in which the predicted output exceeds a certain marginal value with confidence levels corresponding to the tolerances. We would like to derive such marginal value based on the estimation process of the uncertainty point estimators or predictors [57]–[63].

We noted in (4) in the previous section that, generally, a neural network model can be represented as a nonlinear regressive function as follows:

$$y_i(k) = \Phi(x_i, \theta) + \varepsilon_i, \qquad i = 1, 2, \ldots, n \tag{36}$$

where $x_i$ (with dimension $M \times 1$) is the input vector and $\theta$ (with dimension $s \times 1$) is a set of neural network true weights. It is assumed that $\varepsilon_i$ is independent and identically distributed with a normal distribution $N(0, \sigma^2)$. Let us define $\hat{\theta}$ as the least square estimation of $\theta$. In a small neighborhood $\theta$, the linear Taylor series expansion for the model (36) can be shown as follows [62]:

$$\hat{y}_i(k) = \Phi(x_i, \theta) + \Phi_0^T(\hat{\theta} - \theta), \qquad i = 1, 2, \ldots, n \tag{37}$$

where

$$\Phi_0^T = \left[ \frac{\partial \Phi(x_i, \theta)}{\partial \theta_1} \quad \frac{\partial \Phi(x_i, \theta)}{\partial \theta_2} \quad \cdots \quad \frac{\partial \Phi(x_i, \theta)}{\partial \theta_s} \right]. \tag{38}$$

To construct marginal values for nonlinear regressive models in neural networks, the standard asymptotic theory should be applied. For the linear model in (37), an approximate marginal value $(\gamma)$ with $100(1-\alpha)$ confidence can be obtained [58], [62]

$$\gamma = \pm t_{1-\alpha/2,n} \hat{\sigma} \sqrt{1 + \sum_{i=1}^{n} \left( F_i \cdot F_i^T \right)} \tag{39}$$

where $t_{1-\alpha/2,n}$ is the $1 - \alpha/2$ quantile of a $t$-distribution function with $n$ degrees of freedom, $\hat{\sigma}$ is the standard deviation estimator, and $F_i$ is the Jacobian matrix of the neural network outputs with respect to weights, respectively. $\hat{\sigma}$ and $F_i$ are calculated as follows:

$$\hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left( y_i - \Phi(x_i, \hat{\theta}) \right)^2} \tag{40}$$

$$F_i = \left[ \frac{\partial \Phi(x_i, \hat{\theta})}{\partial \theta_1} \quad \cdots \quad \frac{\partial \Phi(x_i, \hat{\theta})}{\partial \theta_s} \right]. \tag{41}$$

In the experimental Section IV-E, we use this marginal value to judge whether the predicted outcomes exceed or not and to know how many overshoots occur.

### *F. Comparisons on Computational Complexity and Storage Requirement*

The computational requirements of the DEKF, described in Section III-A, are dominated by the need to store and update the filtering-error covariance matrix $P(k|k)$ at each time step $n$. For an RNN containing $p$ output nodes and $s$ weights, the computational complexity of the DEKF assumes the following orders:

computational complexity

$$O\left( p^2 s + p \sum_{i=1}^{g} s_i^2 \right) \tag{42}$$

storage requirement

$$O\left( \sum_{i=1}^{g} s_i^2 \right) \tag{43}$$

where $s_i$ is the size of the state in group $i$, $s$ is the total state size, and $p$ is the number of output nodes [17].

The computational requirements of the HEKF are also determined by the need to store and update the filtering-error covariance matrix $P^{CP}$ at each time step $n$. In the HEKF, it also needs to calculate the coupling matrix that contains all components of coupling degree, which means that we need additional $p^2$ computation at each time step $n$. Therefore, the computational complexity of the HEKF assumes the following orders:

computational complexity

$$O\left( p^2(s+1) + p \sum_{i=1}^{c} s_i^2 \right) \tag{44}$$

storage requirement

$$O\left( p^2 + \sum_{i=1}^{c} s_i^2 \right) \tag{45}$$

where $s_i$ is the size of the state in channel $i$.

Note that the HEKF algorithm needs additional computation to calculate the coupling matrix, whereas the total computational complexity depends on the channel number $c$. The total computational complexity of the DEKF algorithm can be determined by the group number $g$. Here, we need to consider the group and channel numbers. If the group number is greater than the channel number $(g > c)$ and the output node number is smaller than the size of the state in group $i (p < s_i)$, the HEKF algorithm can improve computational complexity in comparison to the DEKF algorithm. Note that this complexity analysis does not include the computational requirements for the matrix of dynamic derivatives.

When we compare HEKF and DEKF, the computational complexity of DEKF is recalculated as multiple channel numbers. When we use multiple channel numbers, the computational complexity of the DEKF assumes the following orders:

computational complexity

$$O\left( p^2 sc + cp \sum_{i=1}^{g} s_i^2 \right) \tag{46}$$

storage requirement

$$O\left( c \sum_{i=1}^{g} s_i^2 \right) \tag{47}$$

where $c$ is the channel number. The computational complexities of the DEKF shown in (46) and (47) are larger than that of HEKF, shown in (44) and (45).

When we implement the proposed HEKF method by comparing it to the DEKF method, it is required to evaluate how much is the additional computational time. For the comparison on

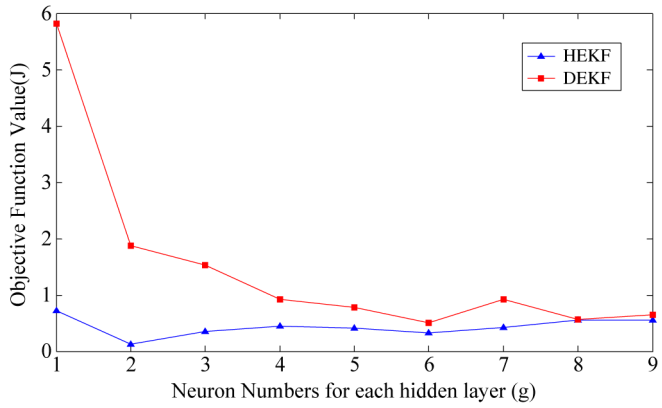


Fig. 6. Comparison of objective function values between the HEKF and the DEKF. With this figure, we can expect to choose the selected neuron number for HEKF or DEKF to be more optimized. Also, the discriminant criterion itself tests whether HEKF or DEKF is less enormous.

computational complexity, we have evaluated the performance of average CPU time in experimental Section IV-F. Here, we used three RMLPs for each channel in HEKF, whereas we used one RMLP in DEKF.

## IV. Experimental Results

### A. Motion Data Captured

We used three channel sets of patient breathing data to evaluate the filter performance. Each set of data consisted of chest displacements recorded continuously in three dimensions at a sampling frequency of 26 Hz. The recordings lasted anywhere from 5 min to 1.5 h of the average time at the Georgetown University CyberKnife treatment facility. These records were arbitrarily selected to represent a wide variety of breathing patterns, including highly unstable and irregular examples. Each patient's breathing record was used to independently train and test the predictive accuracy of the filter.

### B. Optimized Group Number for RMLP

*1) Optimized Group Number:* With respect to the selected group number $(g)$ to implement the RMLP, we used an MLP with two hidden layers, where the first hidden layer is recurrent and the second one is not. We increased the number of hidden units for the first and second hidden layers according to the group number to calculate the objective function value in comparing two different methods. In order to analyze the group number for RMLP, we incorporated objective function (35) in Section III-C.

As shown in Fig. 6, HEKF is optimized when the group number is two, whereas DEKF is optimized when the group number is six. Therefore, we choose the neuron number as two for HEKF and six for DEKF.

*2) Discriminant Criterion to Compare HEKF and DEKF:* Using Fisher linear discriminant on the discriminant analysis in Section III-C, we can expect that HEKF or DEKF could be more optimized by comparing them with the objective function values $J(\cdot)$. Fig. 6 shows the objective function values $J(\cdot)$ defined in (35). HEKF itself has fewer values than DEKF; thus, HEKF has more discriminated or further discriminant degree in comparison to DEKF across any group numbers selected, which means that HEKF has less error than DEKF.

### C. Prediction Overshoot Analysis

To evaluate the performance of overshoot for the prediction values, we derived the marginal value $(\gamma)$ using (39) in Section III-D. In this section, we would like to use this marginal value to judge whether the predicted outcomes exceed or not and to know how many overshoots occur. With the marginal value $(\gamma)$, we can define the upper and lower bounds by adding the marginal value to the measurement value and by subtracting the marginal value from the measurement value, respectively.

Fig. 7 shows the comparison of prediction overshoots between the HEKF and the DEKF. As shown in Fig. 7, most of the estimation values of the HEKF align between the upper and lower bounds. After the transient state, we can notice that the average percentage of prediction overshoot for HEKF is 3.72%, whereas the average percentage of prediction overshoot for DEKF is 18.61%. As shown in Table I, most of the prediction overshoot of the HEKF is within 5%, except for data sets DB00, DB02, and DB03. We have also noticed that DEKF is slightly better than HEKF in the case of data sets DB13 and DB14, which include some discontinuities as well as the system noise because of the irregular patient breathing and the system latency during the breathing record [40]. We think that these lacks of continuity could decrease the Kalman filter gain during the target prediction. In spite of these defect, however, the proposed HEKF can improve the average prediction overshoot by 62.95% in comparison to DEKF.

### D. Comparison on Estimation Performance

We evaluate the target estimation by comparing the proposed HEKF described in Section III-B with the alternative DEKF described in Section III-A.

*1) Tracking Position Estimation:* Fig. 8 shows the average target position estimation of the 3-D Euclidian distance between the predicted value and the measurement values with respect to the data time index given by the original CyberKnife data set. The unit of the vertical axes in Figs. 8 and 9 is dimensionless for the amplitude, i.e., the target estimation corresponds to the 3-D position, has the range of $[-1, +1]$, corresponding to the real measurement data set range $[-1.4735 \times 10^3, -1.5130 \times 10^3]$. As shown, the position estimation values of the HEKF align closer to the measurement values than the DEKF values.

*2) Position Error Value:* We would like to compare the performance of tracking errors with respect to the data time index across the entire measurement period between the HEKF and the DEKF. The error value in Fig. 9 was calculated by the subtraction of the 3-D Euclidian distance between the predicted values and the measurement values in the data time index.

Fig. 9 shows that the error value of the HEKF is smaller than that of the DEKF across the data time index 25 200–25 350 s. At the beginning of tracking estimation, we notice that both approaches have several overshoots across the data time because

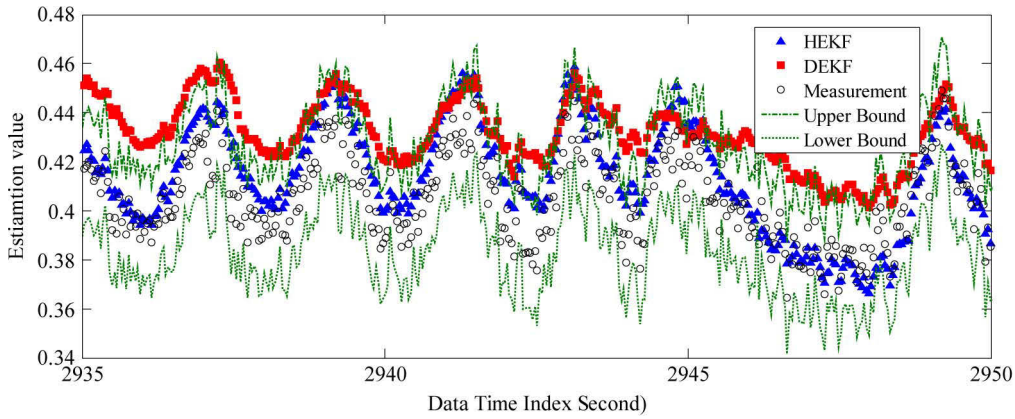


Fig. 7. Comparison of prediction overshoot between the HEKF and the DEKF. In this figure, we can notice that most of the estimation values of the HEKF align between the upper and lower bounds, whereas the values of DEKF do not. After the transient state, we can evaluate that the average percentage of prediction overshoot for the HEKF is 3.72%, whereas the average percentage of prediction overshoot for the DEKF is 18.61%. Thus, HEKF has a smaller prediction overshoot value than DEKF.

TABLE I
PREDICTION OVERSHOOT ANALYSIS (HEKF VERSUS DEKF)

| Datasets | # Total frames | HEKF (#Overshoot Frame / #Total Frame: %) | DEKF (#Overshoot Frame / #Total Frame: %) | Improvement (%) |
|---|---|---|---|---|
| DB00 | 79078 | 17.09 | 31.10 | 45.06 |
| DB01 | 145336 | 3.23 | 7.72 | 58.13 |
| DB02 | 140704 | 8.65 | 39.03 | 77.84 |
| DB03 | 175896 | 5.24 | 6.35 | 17.52 |
| DB04 | 93653 | 4.44 | 27.78 | 84.03 |
| DB05 | 100739 | 4.14 | 26.55 | 84.41 |
| DB06 | 159855 | 1.66 | 32.51 | 94.89 |
| DB07 | 110417 | 0.12 | 41.50 | 99.70 |
| DB08 | 225785 | 1.49 | 32.67 | 95.44 |
| DB09 | 144149 | 0.27 | 0.40 | 31.75 |
| DB10 | 185697 | 4.29 | 15.31 | 71.98 |
| DB11 | 108327 | 0.95 | 12.01 | 92.11 |
| DB12 | 129503 | 0.03 | 2.16 | 98.46 |
| DB13 | 146145 | 0.53 | 0.51 | -4.15 |
| DB14 | 134683 | 3.59 | 3.49 | -2.94 |

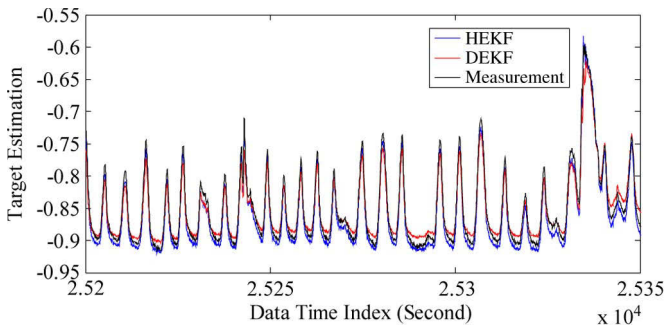


Fig. 8. Target estimation between the HEKF and the DEKF. This figure shows that the position estimation values of the HEKF align closer to the measurement values than the DEKF values.

of the unstable initialization of the original data set. After the steady state, the error value of the HEKF aligns more close to zero point. Two significant position errors are shown in DEKF, whereas the position error is negligible in HEKF.

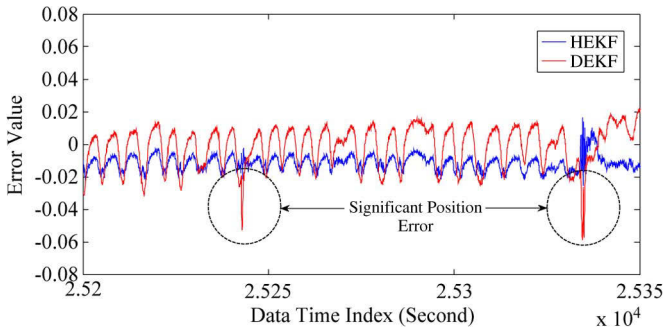


Fig. 9. Comparison of the position error between the HEKF and the DEKF. This figure shows two significant position errors in DEKF, whereas the position error is negligible in HEKF.

### *E. Error Performance Over Prediction Time Horizon*

Prediction time horizon is the term to indicate the consecutive estimate interval that is used to predict the future sensory signal. We would like to compare the error performance among various prediction time horizons between the HEKF and the DEKF in Table II. For the comparison, we used a normalization that is the normalized root-mean-squared error (NRMSE) between the predicted and actual signals over all of the samples in the test data set as follows [40]:

$$NRMSE = \sqrt{\sum_i (y_i - \hat{y}_i)^2 \Big/ \sum_i (y_i - \hat{y}_i)^2} \qquad (48)$$

where $y_i$ is the $i$th measurement, $\hat{y}_i$ is the estimation of the $i$th measurement, and $m_y$ is the mean of all of the measurements. This metric is dimensionless, and it allows us to compare prediction accuracy for different signals of widely varying amplitude.

As shown in Table II, the error performance in the proposed HEKF has improved for all of the data sets by 26.65% in the average of the prediction time horizon for 38.46 ms. The prediction interval time has increased, and the calculated NRMSE has increased. Notice that seven data sets are shown in bold font since the improvement of error performance for the proposed method maintained over 25%, with 50% across the prediction time horizons in data sets DB01, DB03, DB07, and

TABLE II
ERROR PERFORMANCE AMONG PREDICTION TIME HORIZONS (HEKF VERSUS DEKF)

| | Prediction Time Horizon | | | | | | |
|---|---|---|---|---|---|---|---|
| | 38.46*ms* | 115.38 *ms* | 192.3 *ms* | 269.23 *ms* | 346.15 *ms* | 423.07 *ms* | 500 *ms* |
| DB00 | 0.0666/0.0706 | 0.0714/0.0768 | 0.0740/0.0782 | 0.0752/0.0847 | 0.0790/0.0875 | 0.0812/0.0850 | 0.0848/0.0890 |
| **DB01** | **0.0326/0.0739** | **0.0365/0.0771** | **0.0420/0.0876** | **0.0463/0.1015** | **0.0466/0.1085** | **0.0504/0.1087** | **0.0820/0.1134** |
| DB02 | 0.0961/0.1347 | 0.1128/0.1395 | 0.1306/0.1419 | 0.1331/0.1450 | 0.1333/0.1540 | 0.1349/0.1637 | 0.1458/0.1821 |
| **DB03** | **0.0535/0.0896** | **0.0545/0.0917** | **0.0560/0.1122** | **0.0576/0.1260** | **0.0593/0.1342** | **0.0616/0.1348** | **0.0796/0.1519** |
| **DB04** | **0.0440/0.0661** | **0.0503/0.0674** | **0.0589/0.0719** | **0.0613/0.0724** | **0.0638/0.0775** | **0.0668/0.0991** | **0.0672/0.1138** |
| **DB05** | **0.0468/0.0789** | **0.0546/0.0830** | **0.0563/0.0863** | **0.0574/0.0907** | **0.0583/0.0947** | **0.0675/0.0966** | **0.0696/0.0987** |
| DB06 | 0.0265/0.0304 | 0.0279/0.0338 | 0.0304/0.0338 | 0.0340/0.0366 | 0.0361/0.0382 | 0.0388/0.0399 | 0.0409/0.0449 |
| **DB07** | **0.0311/0.0864** | **0.0423/0.0941** | **0.0442/0.0957** | **0.0501/0.0959** | **0.0555/0.0993** | **0.0608/0.1333** | **0.0755/0.1444** |
| DB08 | 0.0555/0.0606 | 0.0590/0.0636 | 0.0621/0.0691 | 0.0737/0.0783 | 0.0763/0.0792 | 0.0816/0.0880 | 0.0866/0.0921 |
| **DB09** | **0.1018/0.1123** | **0.1104/0.1305** | **0.1460/0.1712** | **0.1825/0.2283** | **0.1875/0.2928** | **0.1886/0.3428** | **0.1926/0.3556** |
| DB10 | 0.1010/0.1064 | 0.1078/0.1146 | 0.1133/0.1238 | 0.1251/0.1344 | 0.1342/0.1474 | 0.1495/0.1638 | 0.1786/0.1906 |
| DB11 | 0.0731/0.0987 | 0.1106/0.1237 | 0.1209/0.1293 | 0.1358/0.1377 | 0.1574/0.1586 | 0.1588/0.1638 | 0.1770/0.1797 |
| **DB12** | **0.0424/0.0916** | **0.0459/0.0958** | **0.0470/0.0977** | **0.0474/0.0998** | **0.0504/0.1021** | **0.0505/0.1034** | **0.0630/0.1045** |
| DB13 | 0.0651/0.0788 | 0.0668/0.0811 | 0.0678/0.0815 | 0.0687/0.0839 | 0.0691/0.0908 | 0.0710/0.0948 | 0.0732/0.1036 |
| DB14 | 0.0440/0.0455 | 0.0456/0.0509 | 0.0482/0.0511 | 0.0490/0.0519 | 0.0505/0.0520 | 0.0515/0.0538 | 0.0541/0.0575 |

DB12. Compared to the patient of the CyberKnife data set in the latest paper [47], the proposed HEKF showed a better NRMSE performance across all variable prediction interval times (e.g., at the prediction time horizon of 500 ms, there is a 422% NRMSE improvement).

### *F. Comparisons on Computational Complexity*

We would like to evaluate how much additional computational time is required when we implement the proposed HEKF method by it comparing with the DEKF method. For HEKF, we used three RMLPs for each channel, whereas we used one RMLP for DEKF, where the neuron number for the first and second hidden layers is two for HEKF and six for DEKF, respectively. Regarding CPU experimental time, we have evaluated the overall performance of the average CPU time using a PC with Pentium core 2.4 GHz and with 3.25-GB RAM.

Table III shows the performance of the CPU time used. As shown in Table III, the HEKF method needs more time compared with DEKF. We think that the actual difference of the CPU time used in Table III mainly comes from the calculation of the coupling matrix and the separate neural network for channel number. Although 30.07% more time is required to implement the proposed HEKF, it is a modest tradeoff to consider the better performance than the better computational time under the condition that PC speed is improving these days.

TABLE III
CPU TIME USED IN THE TARGET ESTIMATION

| Datasets | Recording time (minutes) | CPU Time used (Millisecond / #Total Frame) | |
|---|---|---|---|
| | | HEKF | DEKF |
| DB00 | 50.80 | 9.4306 | 7.1737 |
| DB01 | 93.36 | 9.7759 | 7.2836 |
| DB02 | 90.39 | 10.8872 | 7.1532 |
| DB03 | 113.00 | 10.8578 | 6.9824 |
| DB04 | 60.16 | 10.0511 | 7.1556 |
| DB05 | 64.85 | 10.3541 | 7.3941 |
| DB06 | 102.83 | 10.5332 | 7.1505 |
| DB07 | 70.93 | 9.4372 | 6.7484 |
| DB08 | 145.21 | 11.2489 | 7.1755 |
| DB09 | 92.67 | 10.3379 | 7.0038 |
| DB10 | 119.55 | 11.3506 | 7.2783 |
| DB11 | 69.72 | 9.5831 | 7.0640 |
| DB12 | 85.34 | 9.6143 | 6.8265 |
| DB13 | 93.88 | 11.2510 | 7.4613 |
| DB14 | 86.52 | 9.5256 | 7.5890 |

## V. CONCLUSION

In this paper, we have presented respiratory motion estimation with hybrid implementation of EKF, called as HEKF. Our new method has two main contributions in improving the traditional EKF-based RNN target tracking. The first contribution is to present a new approach in splitting the whole RMLP with the complicated neuron number into a couple of RMLPs with the simple neuron number to adjust separate input channels. The second contribution is to comprehensively organize the multiple channel sensory process by adapting the coupling technique using multiple channel inputs for the mutually exclusive groups to compensate the computational accuracy.

The experimental results have validated that the prediction overshoot of the proposed HEKF was improved for 13 data sets among 15 data sets by 62.95%. The proposed HEKF showed a better performance of 52.40% NRMSE improvement in the average of the prediction time horizon. We have evaluated that the proposed HEKF can outperform DEKF by comparing the performance of the tracking estimation values, NRMSE, and prediction overshoot analysis. Moreover, HEKF has more discriminated degree in comparison to DEKF across any group

numbers selected, which means that HEKF has less error than DEKF. Even though the provided method needed more computational time compared with the previous method, the experimental results showed that it improved NRMSE around 24.72% across the overall prediction time horizon.

## ACKNOWLEDGMENT

The work reported here would not be possible without the help of J. Williamson.

## REFERENCES

[1] J. Tan and N. Kyriakopoulos, "Implementation of a tracking Kalman filter on a digital signal processor," *IEEE Trans. Ind. Electron.*, vol. 35, no. 1, pp. 126–134, Feb. 1988.

[2] H.-W. Kim and S.-K. Sul, "A new motor speed estimator using Kalman filter in low-speed range," *IEEE Trans. Ind. Electron.*, vol. 43, no. 4, pp. 498–504, Aug. 1996.

[3] B. Terzic and M. Jadric, "Design and implementation of the extended Kalman filter for the speed and rotor position estimation of brushless dc motor," *IEEE Trans. Ind. Electron.*, vol. 48, no. 6, pp. 1065–1073, Dec. 2001.

[4] M. Barut, S. Bogosyan, and M. Gokasan, "Speed-sensorless estimation for induction motors using extended Kalman filters," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 272–280, Feb. 2007.

[5] S.-H. P. Won, W. W. Melek, and F. Golnaraghi, "A Kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system," *IEEE Trans. Ind. Electron.*, vol. 57, no. 5, pp. 1787–1798, May 2010.

[6] M. Chueh, Y. L. W. Au Yeung, K.-P. C. Lei, and S. S. Joshi, "Following controller for autonomous mobile robots using behavioral cues," *IEEE Trans. Ind. Electron.*, vol. 55, no. 8, pp. 3124–3132, Aug. 2008.

[7] Y. Motai and A. Kosaka, "Hand–eye calibration applied to viewpoint selection for robotic vision," *IEEE Trans. Ind. Electron.*, vol. 55, no. 10, pp. 3731–3741, Oct. 2008.

[8] W.-S. Ra, H.-J. Lee, J. B. Park, and T.-S. Yoon, "Practical pinch detection algorithm for smart automotive power window control systems," *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1376–1384, Mar. 2008.

[9] K. Szabat and T. Orlowska-Kowalska, "Performance improvement of industrial drives with mechanical elasticity using nonlinear adaptive Kalman filter," *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1075–1084, Mar. 2008.

[10] A. G. Beccuti, S. Mariethoz, S. Cliquennois, S. Wang, and M. Morari, "Explicit model predictive control of dc–dc switched-mode power supplies with extended Kalman filtering," *IEEE Trans. Ind. Electron.*, vol. 56, no. 6, pp. 1864–1874, 2009.

[11] K. Szabat, T. Orlowska-Kowalska, and M. Dybkowski, "Indirect adaptive control of induction motor drive system with an elastic coupling," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4038–4042, Oct. 2009.

[12] C. Mitsantisuk, S. Katsura, and K. Ohishi, "Kalman-filter-based sensor integration of variable power assist control based on human stiffness estimation," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3897–3905, Oct. 2009.

[13] N. Salvatore, A. Caponio, F. Neri, S. Stasi, and G. L. Cascella, "Optimization of delayed-state Kalman-filter-based algorithm via differential evolution for sensorless control of induction motors," *IEEE Trans. Ind. Electron.*, vol. 57, no. 1, pp. 385–394, Jan. 2010.

[14] M. Charkhgard and M. Farrokhi, "State of charge estimation for lithium-ion batteries using neural networks and extended Kalman filter," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4178–4187, Dec. 2010.

[15] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 279–297, Mar. 1994.

[16] R. J. Williams, "Training recurrent networks using the extended Kalman filter," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 1992, vol. 4, pp. 241–246.

[17] D. W. Ruck, S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley, "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 6, pp. 686–691, Jun. 1992.

[18] S. Murtuza and S. F. Chorian, "Node decoupled extended Kalman filter based learning algorithm for neural network," in *Proc. IEEE Int. Symp. Intell. Control*, Aug. 1994, pp. 364–369.

[19] S. Li, D. C. Wunsch, E. O'Hair, and M. G. Giesselmann, "Extended Kalman filter training of neural networks on a SIMD parallel machine," *J. Parallel Distrib. Comput.*, vol. 62, no. 4, pp. 544–562, Apr. 2002.

[20] T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Trajectory control of robotic manipulators using neural networks," *IEEE Trans. Ind. Electron.*, vol. 38, no. 3, pp. 195–202, Jun. 1991.

[21] H. Tai, J. Wang, and K. Ashenayi, "A neural network-based tracking control system," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 504–510, Dec. 1992.

[22] T. Fukuda and T. Shibata, "Theory and applications of neural networks for industrial control systems," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 472–489, Dec. 1992.

[23] M. Saad, P. Bigras, L.-A. Dessaint, and K. Al-Haddad, "Adaptive robot control using neural networks," *IEEE Trans. Ind. Electron.*, vol. 41, no. 2, pp. 173–181, Apr. 1994.

[24] T. W. S. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE Trans. Ind. Electron.*, vol. 45, no. 1, pp. 151–161, Feb. 1998.

[25] P. Payeur, H. Le-Huy, and C. M. Gosselin, "Trajectory prediction for moving objects using artificial neural networks," *IEEE Trans. Ind. Electron.*, vol. 42, no. 2, pp. 147–158, Apr. 1995.

[26] C.-H. Lu and C.-C. Tsai, "Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine," *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1366–1375, Mar. 2008.

[27] B. M. Wilamowski, N. J. Cotton, O. Kaynak, and G. Dundar, "Computing gradient vector and Jacobian matrix in arbitrarily connected neural networks," *IEEE Trans. Ind. Electron.*, vol. 55, no. 10, pp. 3784–3790, Oct. 2008.

[28] M. Wlas, Z. Krzemiriski, and H. A. Toliyat, "Neural-network-based parameter estimations of induction motors," *IEEE Trans. Ind. Electron.*, vol. 55, no. 4, pp. 1783–1794, 2008.

[29] J. Mazumdar and R. G. Harley, "Recurrent neural networks trained with backpropagation through time algorithm to estimate nonlinear load harmonic currents," *IEEE Trans. Ind. Electron.*, vol. 55, no. 9, pp. 3484–3491, Sep. 2008.

[30] S. Cong and Y. Liang, "PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3872–3879, Oct. 2009.

[31] T. Orlowska-Kowalska and M. Kaminski, "Effectiveness of saliency-based methods in optimization of neural state estimators of the drive system with elastic couplings," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4043–4051, Oct. 2009.

[32] C. Y. Lai, F. L. Lewis, V. Venkataramanan, X. Ren, S. Sam Ge, and T. Liew, "Disturbance and friction compensations in hard disk drives using neural networks," *IEEE Trans. Ind. Electron.*, vol. 57, no. 2, pp. 784–792, Feb. 2010.

[33] I. J. Leontaritis and S. A. Billings, "Input–output parametric models for non-linear systems Part I: Deterministic non-linear systems," *Int. J. Control*, vol. 41, no. 2, pp. 303–328, Feb. 1985.

[34] S. Chen and S. A. Billings, "Representations of non-linear systems: The NARMAX model," *Int. J. Control*, vol. 49, no. 3, pp. 1013–1032, Mar. 1989.

[35] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[36] O. Nerrand, P. Roussel-Ragot, L. Personnaz, and G. Dreyfus, "Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms," *Neural Comput.*, vol. 5, no. 2, pp. 165–199, Mar. 1993.

[37] G. C. Sharp, S. B. Jiang, S. Shimizu, and H. Shirato, "Prediction of respiratory tumour motion for real-time image-guided radiotherapy," *Phys. Med. Biol.*, vol. 49, no. 3, pp. 425–440, Feb. 2004.

[38] S. S. Vedam, P. J. Keall, A. Docef, D. A. Todor, V. R. Kini, and R. Mohan, "Predicting respiratory motion for four-dimensional radiotherapy," *Med. Phys.*, vol. 31, no. 8, pp. 2274–2283, Aug. 2004.

[39] M. Kakar, H. Nyström, L. R. Aarup, T. J. Nøttrup, and D. R. Olsen, "Respiratory motion prediction by using the adaptive neuro fuzzy inference system (ANFIS)," *Phys. Med. Biol.*, vol. 50, no. 19, pp. 4721–4728, Oct. 2005.

[40] M. J. Murphy and S. Dieterich, "Comparative performance of linear and nonlinear neural networks to predict irregular breathing," *Phys. Med. Biol.*, vol. 51, no. 22, pp. 5903–5914, Nov. 2006.

[41] Q. Ren, S. Nishioka, H. Shirato, and R. I. Berbeco, "Adaptive prediction of respiratory motion for motion compensation radiotherapy," *Phys. Med. Biol.*, vol. 52, no. 22, pp. 6651–6661, Oct. 2007.

[42] D. Ruan, J. A. Fessler, and J. M. Balter, "Real-time prediction of respiratory motion based on local regression methods," *Phys. Med. Biol.*, vol. 52, no. 23, pp. 7137–7152, Dec. 2007.
[43] F. Ernst, A. Schlaefer, S. Dieterich, and A. Schweikard, "A fast lane approach to LMS prediction of respiratory motion signals," *Biomed. Signal Process. Control*, vol. 3, no. 4, pp. 291–299, Oct. 2008.
[44] J. H. Goodband, O. C. L. Haas, and J. A. Mills, "A comparison of neural network approaches for on-line prediction in IGRT," *Med. Phys.*, vol. 35, no. 3, pp. 1113–1122, Mar. 2008.
[45] D. Putra, O. C. L. Haas, J. A. Mills, and K. J. Burnham, "A multiple model approach to respiratory motion prediction for real-time IGRT," *Phys. Med. Biol.*, vol. 53, no. 6, pp. 1651–1663, Mar. 2008.
[46] F. Ernst and A. Schweikard, "Predicting respiratory motion signals for image-guided radiotherapy using multi-step linear methods (MULIN)," *Int. J. CARS*, vol. 3, no. 1/2, pp. 85–90, Jun. 2008.
[47] M. J. Murphy and D. Pokhrel, "Optimization of adaptive neural network to predict breathing," *Med. Phys.*, vol. 36, no. 1, pp. 40–47, Jan. 2009.
[48] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures, and Stability*. New York: Wiley, 2001.
[49] F. J. Pineda, "Generalization of backpropagation to recurrent neural networks," *Phys. Rev. Lett.*, vol. 59, no. 19, pp. 2229–2232, Nov. 1987.
[50] P. J. Werbos, "Backpropagation through time: What it does and how to do it?," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[51] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Netw.*, vol. 6, no. 5, pp. 1212–1228, Sep. 1995.
[52] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, Summer 1989.
[53] R. J. Williams and D. Zipser, "Experimental analysis for the real-time recurrent learning algorithm," *Connection Sci.*, vol. 1, no. 1, pp. 87–111, 1989.
[54] G. Kechriotis and E. S. Manolakos, "Training fully recurrent neural networks with complex weights," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 41, no. 3, pp. 235–238, Mar. 1994.
[55] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Pearson, 2009.
[56] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2001.
[57] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 229–232, Jan. 1996.
[58] J. T. Gene Hwang and A. Adam Ding, "Prediction intervals for artificial neural networks," *J. Amer. Statist. Assoc.*, vol. 92, no. 438, pp. 748–757, Jun. 1997.
[59] T. Heskes, "Practical confidence and prediction intervals," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, pp. 176–182.
[60] D. L. Shrestha and D. P. Solomatine, "Machine learning approaches for estimation of prediction interval for the model output," *Neural Netw.*, vol. 19, no. 2, pp. 225–235, 2006.
[61] D. J. Olive, "Prediction intervals for regression models," *Comput. Statist. Data Anal.*, vol. 51, no. 6, pp. 3115–3122, Mar. 2007.
[62] A. Khosravi, S. Nahavandi, and D. Creighton, "Constructing prediction intervals for neural network metamodels of complex systems," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2009, pp. 1576–1582.
[63] J. G. Ramírez, *Statistical Intervals: Confidence, Prediction, Enclosure*. Cary, NC: SAS Inst. Inc., 2009.
[64] A. Bhattacharya and C. Chakraborty, "A shunt active power filter with enhanced performance using ANN-based predictive and adaptive controllers," *IEEE Trans. Ind. Electron.*, vol. 58, no. 2, pp. 421–428, Feb. 2011.

**Suk Jin Lee** (S'11) received the B.S. degree in electronic engineering and the M.S. degree in telematics engineering from Pukyong National University, Busan, Korea, in 2003 and 2005, respectively. He is currently working toward the Ph.D. degree in the School of Engineering, Virginia Commonwealth University, Richmond.

In 2007, he was a Visiting Research Scientist with the GW Center for Networks Research, The George Washington University, Washington, DC. His research interests include network protocols, neural networks, target estimation, and classification.

**Yuichi Motai** (M'01) received the B.Eng. degree in instrumentation engineering from Keio University, Tokyo, Japan, in 1991, the M.Eng. degree in applied systems science from Kyoto University, Kyoto, Japan, in 1993, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2002.

He is currently an Assistant Professor of electrical and computer engineering with Virginia Commonwealth University, Richmond. His research interests include the broad area of sensory intelligence, particularly in medical imaging, pattern recognition, computer vision, and sensory-based robotics.

**Martin Murphy** received the Sc.B. degree in physics from Brown University, Providence, RI, in 1973 and the Ph.D. degree in physics from the University of Chicago, Chicago, IL, in 1980.

In 1992, he joined the initial development team for the CyberKnife, which is a revolutionary image-guided radiosurgical system, first as the Director of System Development at Accuracy Incorporated and later as a Senior Research Scientist at Stanford University, Stanford, CA. He is currently an Associate Professor with the Department of Radiation Oncology, School of Medicine, Virginia Commonwealth University, Richmond.