Efficient Bayesian Tracking of Multiple Sources of Neural Activity: Algorithms and Real-Time FPGA Implementation

by

Lifeng Miao

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2013 by the
Graduate Supervisory Committee:

Chaitali Chakrabarti, Co-Chair
Antonia Papandreou-Suppappola, Co-Chair
Junshan Zhang
Daniel Bliss
Narayan Kovvali

ARIZONA STATE UNIVERSITY

August 2013

ABSTRACT

Electrical neural activity detection and tracking have many applications in medical research and brain computer interface technologies. In this thesis, we focus on the development of advanced signal processing algorithms to track neural activity and on the mapping of these algorithms onto hardware to enable real-time tracking. At the heart of these algorithms is particle filtering (PF), a sequential Monte Carlo technique used to estimate the unknown parameters of dynamic systems.

First, we analyze the bottlenecks in existing PF algorithms, and we propose a new parallel PF (PPF) algorithm based on the independent Metropolis-Hastings (IMH) algorithm. We show that the proposed PPF-IMH algorithm improves the root mean-squared error (RMSE) estimation performance, and we demonstrate that a parallel implementation of the algorithm results in significant reduction in inter-processor communication. We apply our implementation on a Xilinx Virtex-5 field programmable gate array (FPGA) platform to demonstrate that, for a one-dimensional problem, the PPF-IMH architecture with four processing elements and 1,000 particles can process input samples at 170 kHz by using less than 5% FPGA resources. We also apply the proposed PPF-IMH to waveform-agile sensing to achieve real-time tracking of dynamic targets with high RMSE tracking performance.

We next integrate the PPF-IMH algorithm to track the dynamic parameters in neural sensing when the number of neural dipole sources is known. We analyze the computational complexity of a PF based method and propose the use of multiple particle filtering (MPF) to reduce the complexity. We demonstrate the improved performance of MPF using numerical simulations with both synthetic and real data. We also propose an FPGA implementation of the MPF algorithm and show that the implementation supports real-time tracking.

For the more realistic scenario of automatically estimating an unknown number of time-varying neural dipole sources, we propose a new approach based on the probability hypothesis density filtering (PHDF) algorithm. The PHDF is implemented using particle filtering (PF-PHDF), and it is applied in a closed-loop to first estimate the number of dipole sources and then their corresponding amplitude, location and orientation parameters. We demonstrate the improved tracking performance of the proposed PF-PHDF algorithm and map it onto a Xilinx Virtex-5 FPGA platform to show its real-time implementation potential.

Finally, we propose the use of sensor scheduling and compressive sensing techniques to reduce the number of active sensors, and thus overall power consumption, of electroencephalography (EEG) systems. We propose an efficient sensor scheduling algorithm which adaptively configures EEG sensors at each measurement time interval to reduce the number of sensors needed for accurate tracking. We combine the sensor scheduling method with PF-PHDF and implement the system on an FPGA platform to achieve real-time tracking. We also investigate the sparsity of EEG signals and integrate compressive sensing with PF to estimate neural activity. Simulation results show that both sensor scheduling and compressive sensing based methods achieve comparable tracking performance with significantly reduced number of sensors.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors, Dr. Chaitali Chakrabarti and Dr. Antonia Papandreou-Suppappola for giving the wisdom, strength, support and knowledge in my Ph.D study and life. They patiently provided the guidance and encouragement necessary for me to proceed through the doctorial program and complete my dissertation. They also gave me great freedom to pursue independent work and provided advice when my steps faltered. I have been feeling amazingly fortunate that I could work with Dr. Chaitali Chakrabarti and Dr. Antonia Papandreou-Suppappola in the past years.

My sincere thanks also go to my committee, Dr. Leon Iasemidis, Dr. Junshan Zhang, Dr. Daniel Bliss and Dr. Narayan Kovvali for their support and guidance on my research. Their insightful suggestions and comments are greatly appreciated.

I am deeply indebted to my colleagues at Arizona State University for their stimulating discussions and continuous assistance. I would specially like to thank Jun Zhang, Lanping Deng, Chi-li Yu, Bhavana Chakraborty, Debejyo Chakraborty, Ying Li, Wenfan Zhou, Meng Zhou, Chengen Yang, Ming Yang, Siyuan Wei, Hsing-Min Chen.

I also thank my friends for providing support and friendship that I needed. Special thanks go to Peng Wang, Lv Tang, Qing Zhang, Lin Bo, Cheng Chen, Stefanos Michael, Yang Yang, Tingfang Du. You are always the sources of laughter, joy, and support.

Especially, I am grateful to my family for their unconditional love and care. Without their encouragement and support, I would not have made it this far. Special thanks to the newest additions to my family, Qian Xu, my wife as well as her wonderful family who all have been supportive and caring.

Most importantly, none of this would have been achieved without my best friend, soul-mate, and wife. She has been a true and great supporter and has unconditionally loved me during my good and bad times. I truly thank Qian for sticking by my side, understanding and encouraging me both academically and personally. During the past several years, we both learned a lot about life, and strengthened our determination to each other.

This work was supported by National Science Foundation (NSF) under Grant No. 0830799.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF ACRONYMS

Chapter 1

Introduction

1.1 Motivation

Detection and tracking of electrical neural activity can improve our understanding of how the human brain functions. Neural tracking techniques have been established as powerful tools for studying both normal and abnormal neural activity [1]. For instance, neural activity tracking techniques have helped to improve the understanding and treatment of serious neurological disorders such as epilepsy and Parkinson's disease. More specifically, these techniques are used to distinguish between different kinds of seizures based on the location and orientation of the seizure foci, thus resulting in improving the outcomes of epilepsy surgery [2]. For patients with Parkinson's disease, the techniques are used to identify and locate the area of the brain where the symptoms are generated in order to improve the performance of deep brain stimulation treatments [3].

In addition to clinical use, neural tracking has applications in brain computer interface (BCI) and augmented cognition [4]. The brain exhibits measurable changes in electrical activity when responding to certain stimulus. Using neural tracking techniques, these changes can be detected and used to control a computer system, for example, to direct a cursor or control a robot arm [5]. In augmented cognition, accurately tracking brain neural activity can be used to determine whether a person is asleep, awake, tired or angry and can be applied to prevent people from falling asleep behind the wheel [6].

Such advances in neural activity tracking have been possible partly because of advances in brain scanning technology, including magnetoencephalography (MEG) and electroencephalography (EEG) [7, 8, 1]. The human brain consists of a large number of neurons that have a resting state characterized by a cross-membrane voltage difference. When an electromagnetic signal is transferred from one neuron to another, a chemical postsynaptic potential is created that can be modeled as a localized current dipole [9]. When thousands of neighboring neurons are simultaneously in this postsynaptic excitation state, localized current is generated that creates an electromagnetic field outside the skull. The magnetic field can be recorded as an MEG signal using a superconducting quantum interference device; the corresponding electric potential can be recorded as an EEG signal using multiple electrodes placed at different locations on the scalp.

Among the available functional imaging techniques, MEG and EEG have temporal resolutions below 100 ms [10]. The excellent time resolution allows us to explore the timing of basic neural processes at the level of cell assemblies. However, the spatial resolution of an EEG/MEG system is limited by the relatively small number of spatial measurements (a few hundred versus tens of thousands in functional magnetic resonance imaging (fMRI)) [7]. Achieving high spatial resolution requires accurate solution to the EEG/MEG inverse problem; the inverse problem is the estimation of the localized current dipole model from EEG/MEG measurements. In this study, we develop advanced signal processing algorithms and hardware architectures to accurately solve the EEG/MEG inverse problem in real-time. Added to this complexity is the portability constraints of the device when used in the wearable ambulatory EEG (AEEG) mode that is used to improve the quality and accuracy of brain disorder tests [4, 11, 12, 13]. In this study, we propose sensor scheduling and data compression methods to reduce the number of sensors required and thus reduce the power consumption due to these devices.

1.2 Problems addressed

Several methods have been applied to solve the EEG/MEG inverse problem, including the multiple signal classification (RAP MUSIC) approach [14, 15], spatial filters or beamformers [16], and Bayesian methods [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. As Bayesian methods have been extensively used for tracking multiple objects, they provide suitable candidates for online tracking of current dipole sources [24, 26] and are investigated in this study. Among sequential Bayesian estimation techniques, Kalman filtering (KF) can provide optimal parameter estimates for linear systems under additive Gaussian noise [28]. Although KF was used to solve the EEG inverse problem in [18, 19], KF is not applicable for highly nonlinear and non-Gaussian systems.

Particle filtering (PF) is a sequential Bayesian technique that provides more accurate performance for estimating parameters of nonlinear and non-Gaussian dynamic systems [29, 30, 31, 32]. Since the EEG/MEG measurement model is highly nonlinear, PF is more suitable for solving the EEG/MEG inverse problem. In [19, 20, 21], PF was applied to estimate the locations of neural dipole sources. In an effort to improve estimation performance, a Rao-Blackwellized PF and beamforming PF were applied in [22, 23]. However, these systems are based on software implementations and cannot meet the stringent requirements of real-time processing. In this work, we

develop efficient implementations of PF that map well onto a parallel hardware platform and enable real time tracking of neural dipole sources. Specifically, we address the following 5 problems.

1. Parallel PF for real-time performance

There are three major operations in PF processing: particle generation, weight calculation, and resampling. As shown in [33, 34], the bottleneck in real-time PF implementation is the resampling operation. Several modifications of the resampling algorithm, such as residual-systematic resampling and threshold based resampling, were proposed to reduce computational complexity [35, 36, 37]. The threshold based resampling algorithm in [35] was modified to obtain the compact resampling algorithm that helped improve tracking performance in [38, 39, 40]. A systematic resampling algorithm with non-normalized weights was proposed in [37] to enable pipelined implementation of PF. In [41], a particle-tagging quantization scheme was used to make the number of particles a power of two and thus reduce the hardware complexity of the PF residual resampling algorithm. The aforementioned resampling algorithms are all modified versions of the systematic resampling algorithm [31] or residual resampling algorithm [42]. For both algorithms, resampling cannot be computed unless knowledge of all particle weights is available, and thus poses a considerable challenge for pipelined implementation. In order to eliminate this bottleneck, independent Metropolis-Hastings (IMH) resampling can be employed as it can start as soon as the first particle weight is available [43, 44]. In this work, we propose a parallel algorithm for PF that integrates the parallel PF (PPF) algorithm in [45] with the IMH sampler. When mapped to hardware, this algorithm reduces the communication between the parallel processing element and the central unit significantly. We present a pipelined and parallel architecture to implement the proposed PPF-IMH algorithm and map it onto a Xilinx field-programmable gate array (FPGA) hardware platform to show its ability for supporting real-time processing.

2. Application of PPF-IMH to waveform agile sensing

An important application of particle filtering is waveform-agile sensing, where the waveform is adaptively configured at each time [46, 47, 48]. When waveform-agility is integrated into particle filtering, the computational complexity increases significantly. However, if PF can be implemented in parallel, real-time implementation of adaptive waveform design schemes is indeed possible. We demonstrate this by applying the proposed PPF-IMH to waveform-agile sensing. We improve the

dynamic target tracking performance in terms of root mean-square error (RMSE). We also implement the proposed integrated waveform-agile PPF-IMH system on an FPGA platform and show that it can be used for real-time processing applications.

3. Application of PPF-IMH to neural activity tracking for a known number of sources

We model the neural activity as a stochastic process and apply PPF-IMH to track the positions and moments of neural dipole sources. The number of dipole source parameters to be estimated increases considerably when tracking multiple neural activities. This can result in a significant increase in the PF computational complexity as the number of particles required must be increased to maintain certain level of tracking performance. In this study, we propose the use of multiple PFs [49] to reduce the computational complexity of the multiple dipole sources tracking problem. The algorithm uses separate, parallel and interactive sub-particle filters to track each dipole, thus dividing the high-dimensional, multiple-dipole source model system into multiple, low-dimensional, single-dipole source model systems [50]. Each sub-particle filters can then operate on fewer particles, so that the overall integrated system has a much higher implementation efficiency. We map the resulting algorithm on a Xilinx FPGA platform to demonstrate its applicability to real-time neural activity tracking. This work has resulted in the development of the first hardware prototype of a neural activity tracking system [50].

4. Development of PF-PHDF for tracking unknown number of neural dipole sources

The approaches mentioned thus far typically assume that neural activity can be represented by a fixed and known number of current dipole sources. However, this is not a realistic assumption: neural activity varies with time, so current dipole source models and their parameters should also vary with time. In [25, 26, 27], an approach was used to dynamically estimate the number of dipoles and their parameters at each time step by modeling them as random finite sets (RFS) [51]. In particular, the authors first estimated the number of sources by maximizing the marginal distribution of the current dipole set posterior density and then tracked their locations as the maxima of the RFS first moment or probability hypothesis density. This system was implemented using a PF, and each particle was sampled from a point process, resulting in particles with varying dimensions. The multiple PF tracker algorithm was thus impractical as it used a very large number of particles (in the order of 10^5 particles for five dipoles [25]). In this study, we propose a new

algorithm for estimating both the unknown number of neural dipole sources and their parameters for real EEG/MEG data, with significantly fewer particles, using the probability hypothesis density filter (PHDF). The PHDF was used in [51] to recursively estimate the number of objects and their parameters in order to overcome the PF bottleneck. However, the use of the PHDF for solving the EEG/MEG inverse problem is more challenging. Specifically, the PHDF requires that each measurement is generated exclusively from a single object. However, each EEG/MEG sensor measurement consists of contributions from all dipole sources. Also, the PHDF propagation equations are complex. Extending our initial work in [52], we propose a computationally less expensive algorithm in order to overcome these limitations. In particular, we first reduce the dimension of the EEG/MEG measurements using a threshold-based eigenvalue distilling algorithm; this is needed to facilitate separating the measurements into independent components corresponding to the different dipole sources [53]. We then use the PF implementation of the PHDF (PF-PHDF) [54] to estimate the time-varying number of dipoles at each time step before estimating their unknown parameters. This method simplifies the dual estimation problem to a known-number of dipole sources estimation problem and decreases the number of required particles, thereby reducing the computational complexity. Furthermore, use of windowing along with parallelization helps speed up processing. The algorithm is implemented on a Xilinx Virtex-5 FPGA platform. For a 4-processor architecture, the processing time for one iteration of PHDF was shown to be 48.52 μs, and the processing time for a windowed data segment of 100 samples, obtained using a 1 kHz sampling rate, took only 5.1 ms. Thus our EEG/MEG tracking system has enough computing power to perform real-time processing for up to 10 kHz sampling rate.

5. Efficient design of EEG tracking system

A typical EEG system contains hundreds of sensors, and these sensors consume a lot of power. In [55, 56], low power EEG sensors with power consumption of about 10 milliWatts per sensor were designed. In this study, we focus on reducing the number of sensors required in an EEG system in an effort to reduce the power cost. Our objective is to achieve comparable tracking performance using a reduced number of EEG sensors. We propose two methods to achieve this objective: sensor scheduling [57, 58] and compressive sensing [59, 60]. First, we propose a sensor scheduling algorithm which adaptively configures the EEG sensors at each time step using the minimum pre-

dicted mean-squared error (PMSE) or maximum signal-to-noise ratio (SNR) as the performance metric. Optimization is performed globally by searching over all available sensors. We show that the proposed sensor scheduling algorithm significantly reduces the number of sensors required with minimum estimation performance degradation. For example, using only 15 out of 32 sensors, the RMSE slightly increases from 6.28 mm to 6.41 mm. Next, we analyze the EEG data sparsity in the spatial domain and integrate compressive sensing technique with PF to track the parameters of the neural dipole sources. The RMSE tracking results of the proposed compressive PF are comparable with those of conventional methods. However, the number of required EEG sensors is now reduced from 238 to 50, resulting in significant reduction in the sensor power consumption.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

In Chapter 2, we first introduce PF and then analyze the bottlenecks of the hardware implementation of PF. We present a new parallel PF with IMH sampler (PPF-IMH) which can achieve real-time processing without performance degradation. Next, we implement the PPF-IMH algorithm on an FPGA platform and demonstrate its real-time performance.

In Chapter 3, we apply the PPF-IMH to waveform-agile sensing application and map the proposed sensing system onto an FPGA platform. We also demonstrate its tracking performance in terms of RMSE and present the hardware implementation results.

In Chapter 4, we apply the proposed PPF-IMH to track neural activity. We assume that the number of neural dipole sources is known and use PPF-IMH to estimate the parameters of dipole sources. Then, we propose the use of multiple PF to reduce the computational complexity of the multiple-dipole tracking problem. Estimation of the neural dipole parameters are shown for both synthetic and real MEG data, and hardware implementation results are presented to demonstrate the real-time processing capacity of the proposed multi-dipole tracking system.

In Chapter 5, we propose a new algorithm to track an unknown number of dipole sources. We model the dipole sources as RFS and propose a new PF-PHDF algorithm to estimate the number of dipole sources and their parameters in real-time. Software and hardware implementation results are presented to show the performance of the proposed system in terms of estimation error, processing time and hardware resource utilization.

In Chapter 6, sensor scheduling and compressive sensing are used to design an efficient EEG neural activity tracking system. Numerical simulation results show that the required number of EEG sensors for accurate neural source localization is significantly reduced by using both methods.

We conclude the report with a summary of our work along with a discussion on future directions in Chapter 7.

Chapter 2

Particle Filtering and Its Parallel Implementation

The particle filtering (PF) algorithm is a technique for implementing a recursive Bayesian filter by sequential Monte Carlo simulations. It yields high performance for estimating nonlinear and/or non-Gaussian dynamic system parameters. However, the PF algorithm is iterative and computationally complex. As a result, its real-time implementation is quite challenging. In this chapter, we analyze the bottlenecks of existing parallel PF algorithms, and we propose a new approach that integrates parallel PFs with independent Metropolis-Hastings (PPF-IMH) resampling algorithms. We also implement the PPF-IMH algorithm onto a field-programmable gate array (FPGA) platform and analyze its performance.

2.1 Bayesian tracking

Bayesian tracking techniques are a group of algorithms that can recursively estimate a set of parameters for a dynamic system, given noisy measurements and some prior knowledge. A dynamic system consists of two mathematical models. The first model, known as the state model (also known as the system model or dynamic model), describes the evolution of the state with time. The second model relates the noisy measurements to the state of the system and is called the measurement model. Generally, the state and measurement models for discrete-time systems can be expressed as,

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \nu_{k-1} \tag{2.1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{n}_k \tag{2.2}$$

where $\mathbf{x}_k$ is the vector of unknown system parameters at time step k, $\mathbf{z}_k$ is the vector of measurements at time step k, $f(\cdot)$ is a (possibly) nonlinear state evolution function, $h(\cdot)$ is a (possibly) nonlinear function that relates the state vector with the measurement vector, ν_k is the state modeling error vector, and $\mathbf{n}_k$ is the measurement noise vector. The tracking problem is to recursively estimate $\mathbf{x}_k$ from a set of measurements $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\}$ up to time k. From a Bayesian perspective, the tracking problem is to recursively construct the posterior probability density function

of the state $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, given the sequentially obtained measurements. It is assumed that the initial probability density function or the prior of the state vector $p(\mathbf{x}_0|\mathbf{z}_0) = p(\mathbf{x}_0)$, is available.

The iterative Bayesian approach consists of two steps to obtain the posterior probability density function $p(\mathbf{x}_k|\mathbf{z}_{1:k})$: prediction and update. Assuming that the probability density function $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ at time $k-1$ is available, the prediction step uses the state model in Equation (2.1) to obtain the prior probability density function of the state at time k following the Chapman-Kolmogorov equation [29]

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.3}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ represents the probabilistic evolution model of the state and is given by the state model in Equation (2.1).

At time step k, the update stage involves using the new available measurement $\mathbf{z}_k$ to obtain the posterior probability density function $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ via the Bayes theorem [29]

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \tag{2.4}$$

where $p(\mathbf{z}_k|\mathbf{x}_k)$ is the likelihood function defined by the measurement model in Equation (2.2) and

$$p(\mathbf{z}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})d\mathbf{x}_k \tag{2.5}$$

The prediction and update Equations (2.3) and (2.4) form the basis for the Bayesian tracking algorithm. Solutions to these equations are available only if the system and measurement models satisfy some conditions, as summarized next.

- When the state and measurement models of a system are linear and the modeling error and noise processes are Gaussian, then the Kalman filtering (KF) can provide optimal solutions to this problem [28].

- When the state-space of the system is a finite, discrete-valued sequence, then grid-based search methods can be used to estimate the unknown system state [32].

- When the posterior probability density function of a nonlinear dynamic system has a sufficient statistic, *Benes* [61] and Daum filters [62] can be applied to estimate the probability density of the system state.

However, in practice, these restrictive conditions are not always satisfied, and analytical approximations and suboptimal Bayesian methods can be employed to obtain the posterior probability density function, such as the extended Kalman filtering (EKF), unscented Kalman filtering (UKF) and approximate grid-based methods [32]. More generally, when the models are nonlinear and/or the processes in the models are non-Gaussian, the posterior probability density function can be estimated using particle filtering [29].

2.2 Particle filtering

The PF algorithm is a technique for implementing a recursive Bayesian filter by Monte Carlo (MC) simulations. It is a sequential MC method that is used to estimate the dynamic state parameters of nonlinear and/or non-Gaussian systems [29, 30]. The estimation is performed by approximating the posterior probability density function of the unknown state parameters at each time step given measurements up to that time step. Specifically, for a dynamic system described in Equations (2.1) and (2.2), the PF approximates the joint posterior probability density function of $\mathbf{x}_k$ at time k using a set of N random samples or particles, $\mathbf{x}_k^{(\ell)}$ and their corresponding weights, $w_k^{(\ell)}$, $\ell = 1, \ldots, N$, as:

$$p(\mathbf{x}_k|\mathbf{z}_k) \approx \sum_{\ell=1}^{N} w_k^{(\ell)}\, \delta(\mathbf{x}_k - \mathbf{x}_k^{(\ell)}).$$

where $\delta(\cdot)$ is the Dirac delta function. Using this approximation, the estimated state parameter vector can be obtained as $\hat{\mathbf{x}}_k \approx \sum_{\ell=1}^{N} w_k^{(\ell)}\, \mathbf{x}_k^{(\ell)}$.

There are different PF algorithms, depending on the choice of importance density used to compute the weights [31, 30]. One of the most commonly used algorithms is the sequential importance resampling (SIR) PF that consists of the following basic three steps:

1. *Particle generation*. The particles $\mathbf{x}_k^{(\ell)}$ are drawn from an importance density function

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)}, \mathbf{z}_{1:k}),$$

   where $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$.

2. *Weight computation*. The corresponding weights are calculated as

$$w_k^{(\ell)} \propto w_{k-1}^{(\ell)} \frac{p\left(\mathbf{z}_k|\mathbf{x}_k^{(\ell)}\right)\; p\left(\mathbf{x}_k^{(\ell)}|\mathbf{x}_{k-1}^{(\ell)}\right)}{q\left(\mathbf{x}_k^{(\ell)}|\mathbf{x}_{k-1}^{(\ell)}, \mathbf{z}_{1:k}\right)}$$

and then normalized so that $\sum_{\ell=1}^{N} w_k^{(\ell)}$=1. Note that the importance density is often chosen to be the prior density function $q(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)}, \mathbf{z}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)})$. This simplifies the weight computation to $w_k^{(\ell)} \propto w_{k-1}^{(\ell)} p(\mathbf{z}_k|\mathbf{x}_k^{(\ell)})$.

3. *R*esampling. The particles are resampled to avoid particle degeneracy, which occurs when most particle weights are close to zero, resulting in a poor representation of the posterior probability density function [30]. Resampling avoids degeneracy by eliminating particles with low importance weights and replicating particles with high importance weights.

Even with the simplified weight computation, the SIR PF can be very computationally intensive as the number of particles required is large. For example, in a radar tracking problem, a PF using $N \approx 1,000$ particles requires about $30N$ additions, $20N$ multiplications, and N exponential calculations per iteration. Thus, the overall computational complexity is very high.

Real-time implementation of the SIR PF is only possible through the use of pipelining and parallel processing. In [35], the authors investigated the hardware implementation of the SIR PF, and they pointed out that the main challenges are accelerating the process of resampling and reducing the computational intensity of the algorithm. In [33, 34], the authors also concluded that the systematic resampling step is the bottleneck of hardware implementation of the SIR PF. Since particle generation and weight calculation do not have any data dependencies, they can be easily pipelined. However, systematic resampling requires the knowledge of all normalized weights which makes it hard to be pipelined with other steps. In [37], the authors proposed a parallel architecture for the SIR PF, and they indicated that reducing the communication amount among different processing elements was the main issue of their parallel architecture.

2.3 Parallel PF with independent Metropolis-Hastings sampling

In this section, we propose a new particle filtering algorithm which makes use of Metropolis-Hastings sampling. As we demonstrate, the algorithm yields high tracking performance comparable to the PF with systematic resampling, and it also supports parallel and pipelined processing.

Algorithm 1 Metropolis-Hastings algorithm [44]

Choose a starting point $\left(\mathbf{x}_k^{(0)}, w_k^{(0)}\right)$

for $\ell = 1$ to N **do**

From $\mathbf{x}_k^{(\ell)}$, draw samples (and compute corresponding weights) $(\mathbf{x}_k^*, w_k^*)$ from $q\left(\mathbf{x}_k^* | \mathbf{x}_k^{(\ell)}\right)$

Compute probability $\alpha\left(\mathbf{x}_k^{(\ell)}, \mathbf{x}_k^*\right) = \min\left\{\frac{p\left(\mathbf{x}_k^*\right) q\left(\mathbf{x}_k^{(\ell)} | \mathbf{x}_k^*\right)}{p\left(\mathbf{x}_k^{(\ell)}\right) q\left(\mathbf{x}_k^* | \mathbf{x}_k^{(\ell)}\right)}, 1\right\}$

$$\left(\mathbf{x}_k^{(\ell+1)}, w_k^{(\ell+1)}\right) = \begin{cases} (\mathbf{x}_k^*, w_k^*), & \text{with probability } \alpha\left(\mathbf{x}_k^{(\ell)}, \mathbf{x}_k^*\right) \\ \left(\mathbf{x}_k^{(\ell)}, w_k^{(\ell)}\right), & \text{with probability } 1 - \alpha\left(\mathbf{x}_k^{(\ell)}, \mathbf{x}_k^*\right) \end{cases}$$

end for

2.3.1 Metropolis-Hastings Algorithm

We use the Metropolis-Hastings (MH) algorithm to perform PF resampling in order to overcome the bottleneck caused by systematic resampling. The MH resampling computation can start as soon as the first particle weight becomes available [44]. Specifically, the MH algorithm does not require all the particles as it can generate a Markov chain in which the current state $\mathbf{x}_k^{(\ell)}$ depends only on the previous state $\mathbf{x}_{k-1}^{(\ell)}$ [63]. In particular, the MH algorithm can draw samples from a desired probability density function $p(\mathbf{x}_k)$ given a proposal probability density function $q(\mathbf{x}_k)$. The steps of the MH algorithm are provided in Algorithm 1.

In Algorithm 1, the step of accepting the sample $\mathbf{x}_k^*$ can be implemented by first generating the uniform random variable u, taking values between 0 and 1, and then performing

$$\left(\mathbf{x}_k^{(\ell+1)}, w_k^{(\ell+1)}\right) = \begin{cases} \left(\mathbf{x}_k^*, w_k^*\right), & if \;\; u \leq \min\left\{w_k^*/w_k^{(\ell)}, 1\right\} \\ \left(\mathbf{x}_k^{(\ell)}, w_k^{(\ell)}\right), & if \;\; u > \min\left\{w_k^*/w_k^{(\ell)}, 1\right\} \end{cases}.$$

The independent Metropolis-Hastings (IMH) algorithm can be obtained when $q(x_k^* | \mathbf{x}_k^{(\ell)})$ is independent of $\mathbf{x}_k^{(\ell)}$ in Algorithm 1. Note that, as there is no need to wait for all the particles and their weights to become available [43], the IMH algorithm is suitable for pipelined hardware implementation.

2.3.2 PPF-IMH hardware implementation

We propose a parallel PF algorithm which can be mapped into a multiple processing element architecture. The processing elements (PEs) perform the major PF computational workload (particle generation, weight evaluation and resampling), and a central unit (CU) performs global computations and coordinates the activities of the PEs. If the PF is implemented by computing the systematic resampling in the CU, all the importance weights would have to be transferred, resulting in a huge communication overhead. In [45], a method to significantly reduce the amount of data communication overhead was proposed. The main idea was to divide the particles into several groups in each PE and use the average of each group as the new particle. However, this method results in an estimation performance degradation. To improve estimation performance while keeping the communication overhead low, we propose to use the IMH resampling in each PE before communication with the CU. Using the IMH resampling, the particles $\tilde{\mathbf{x}}_k^{(\ell)}$, $\ell = 1,\ldots,N$, are resampled to obtain $\mathbf{x}_k^{(\ell)}$ in order to more accurately represent the posterior probability density function. The information of the resampled particles is then sent to the CU. Also, since the IMH resampler can be easily pipelined with the other steps, the processing period is not increased.

The new PPF-IMH algorithm is described next in detail. We distribute M particles to P PEs , so N=M/P particles are assigned to each PE. The mth PE, $m = 1,\ldots,P$, executes the processing steps in Algorithm 2 (sampling, weights computation, and IMH resampling) to generate the resampled particle set $\mathbf{x}_{k,m}^{(\ell)}$, $\ell = 1,\ldots,N$. Note that in Algorithm 2, we use $(N+N_b)$ particles since at the end of the processing, we discard N_b samples from the start of the sequence as they may not converge to a good estimate [44].

Next, we present the one-dimensional grouping method or Algorithm 3 that is used to reduce the communication overhead. First, we find the local minima and local maxima of the mth PE as $x_{\min,m} = \min_\ell x_{k,m}^{(\ell)}$ and $x_{\max,m} = \max_\ell x_{k,m}^{(\ell)}$, respectively, and then transmit them to the CU. The CU then finds the global maxima x_{Max} and global minima x_{Min}, and sends them back to all the PEs. Based on x_{Max} and x_{Min}, the particles in each PE are divided into $G = \lceil (x_{\text{Max}} - x_{\text{Min}})/\delta \rceil$ groups where $\lceil a \rceil$ represents the smallest integer greater than a [45]. The parameter δ provides the range of each group; if δ is large, then the number of groups in each PE is small and thus the algorithm

precision is low. The mth PE calculates the average particle value $x_{\text{mean},j,m}$ and particle weight $w_{\text{mean},j,m}$ of group j , $j = 1,\ldots,G$, and transmits them to the CU. The CU uses these values to compute the particle replication factor ρ_j . It also ensures that the replication factor is an integer number by simple rounding-off operations and that $\sum_{j=1}^{G} \rho_j = N$.

This grouping method can be extended to multi-dimensional problem by operating Algorithm 3 on each of the dimensions of the particles. Assuming $\mathbf{x}_k^{(\ell)} = [x_{1,k}^{(\ell)}\ x_{2,k}^{(\ell)}\ \ldots\ x_{\mathfrak{D},k}^{(\ell)}]^T$ is a $\mathfrak{D}$ dimensional particle, then $\min_\ell\ \mathbf{x}_k^{(\ell)} = [\min_\ell x_{1,k}^{(\ell)}\ \min_\ell x_{\mathfrak{D},k}^{(\ell)}]^T$ and $\max_\ell\ \mathbf{x}_k^{(\ell)} = [\max_\ell x_{1,k}^{(\ell)}\ \max_\ell x_{\mathfrak{D},k}^{(\ell)}]^T$. Local extrema $\mathbf{x}_{\min,m} = \min_\ell\ \mathbf{x}_{k,m}^{(\ell)}$ and $\mathbf{x}_{\max,m} = \max_\ell\ \mathbf{x}_{k,m}^{(\ell)}$ of the mth PE are transmitted to the CU. They are used to compute the global extrema $\mathbf{x}_{\text{Max}}$ and $\mathbf{x}_{\text{Min}}$ which are sent back to the PE. Here, $\mathbf{x}_{\min,m}$, $\mathbf{x}_{\max,m}$, $\mathbf{x}_{\text{Max}}$ and $\mathbf{x}_{\text{Min}}$ are all $\mathfrak{D}$ dimensional vectors. For each dimension, particles are divided into G groups based on $\mathbf{x}_{\text{Max}}$ and $\mathbf{x}_{\text{Min}}$, and thus there are $G \times \mathfrak{D}$ groups. Then the average particle value $\mathbf{x}_{\text{mean},j,m}$ and particle weight $w_{\text{mean},j,m}$ of group j, $j = 1,\ldots,G \times \mathfrak{D}$, are calculated and transmitted to the CU for calculating the particle replication factor ρ_j.

The PPF-IMH algorithm has advantages both in terms of algorithm and hardware performance. In each PE, the particles $\mathbf{x}_k^{(\ell)}$, $\ell = 1,\ldots,N$ are resampled using the IMH; thus, particles with high weights are replicated and particles with low weights are discarded. As a result, the remaining particles represent the posterior probability density function more accurately, resulting in improved performance. The PPF-IMH also results in reduced communication overhead. Specifically, in a traditional parallel architecture, M weights and M index factors have to be shared between the PEs and the CU, and, in the worst case scenario, there could be $M/2$ inter-PE communications [35]. For comparison, in the PPF-IMH, only the mth PE range factors $\mathbf{x}_{\min,m}$, $\mathbf{x}_{\max,m}$, $\mathbf{x}_{\text{Min}}$, and $\mathbf{x}_{\text{Max}}$, the average weights $w_{\text{mean},j,m}$, $j = 1,\ldots,G \times \mathfrak{D}$, and the replication factors ρ_j, $j = 1,\ldots,G \times \mathfrak{D}$ need to be transferred between the mth PE and the CU. Also, there is no inter-PE communication. As a result, the communication is reduced to $(2G \times \mathfrak{D} \times P) + (4 \times P)$, where G is the number of groups in each PE, $\mathfrak{D}$ is the vector dimension and P is the number of PEs. Also, since the IMH resampler does not need all the normalized weights, resampling can start once the first weight is computed. Thus, the computation time of the PPF-IMH method increases very mildly when compared to the parallel PF (PPF) algorithm in [45].

Algorithm 2 Parallel Particle Filtering with IMH

Input $\mathbf{z}_k$ and initial set $\mathbf{x}_0^{(\ell)} \sim p(\mathbf{x}_0)$, $\ell = 1,\ldots,N$

for $k = 1$ to K time step **do**

Sampling {*Generate particles and weights*}
for $\ell = 1$ to $(N+N_b)$ **do**
$J(\ell) \sim \mathscr{U}[1,N]$ {*Sample from a discrete uniform distribution between* 1 *and* N}
$\tilde{\mathbf{x}}_k^{(\ell)} \sim p\left(\mathbf{x}_k^{(\ell)} | \mathbf{x}_{k-1}^{(J(\ell))}\right)$
Calculate $\tilde{w}_k^{(\ell)} = p\left(\mathbf{z}_k | \tilde{\mathbf{x}}_k^{(\ell)}\right)$
end for

IMH resampling
Initialize the chain $\left(\bar{\mathbf{x}}_k^{(1)}, \bar{w}_k^{(1)}\right) = \left(\tilde{\mathbf{x}}_k^{(1)}, \tilde{w}_k^{(1)}\right)$
for $\ell = 2$ to $(N+N_b)$ **do**
$u \sim \mathscr{U}(0,1)$

$$\alpha\left(\bar{\mathbf{x}}_k^{(\ell-1)}, \tilde{\mathbf{x}}_k^{(\ell)}\right) = \min\left\{\tilde{w}_k^{(\ell)} / \bar{w}_k^{(\ell-1)}, 1\right\}$$

$$\left(\bar{\mathbf{x}}_k^{(\ell)}, \bar{w}_k^{(\ell)}\right) = \begin{cases} \left(\tilde{\mathbf{x}}_k^{(\ell)}, \tilde{w}_k^{(\ell)}\right), & u \le \alpha\left(\bar{\mathbf{x}}_k^{(\ell-1)}, \tilde{\mathbf{x}}_k^{(\ell)}\right) \\ \left(\bar{\mathbf{x}}_k^{(\ell-1)}, \bar{w}_k^{(\ell-1)}\right), & u > \alpha\left(\bar{\mathbf{x}}_k^{(\ell-1)}, \tilde{\mathbf{x}}_k^{(\ell)}\right) \end{cases}$$

end for

Assign $\left\{\left(\mathbf{x}_k^{(\ell)}, w_k^{(\ell)}\right), \ell = 1,\ldots,N\right\}$ to $\left\{\left(\bar{\mathbf{x}}_k^{(\ell)}, \bar{w}_k^{(\ell)}\right), \ell = (N_b+1),\ldots,(N+N_b)\right\}$

end for

2.3.3 PPF-IMH FPGA implementation

The overall block diagram of the proposed PPF-IMH hardware implementation architecture is shown in Figure 2.1 which consists of four PEs and one CU. Local PF processing steps, such as particle generation, weight evaluation and IMH resampling, are executed in each PE. Global processing steps, such as computing global range and replication factors, are executed in the CU. Each PE communicates with the CU, but there is no communication among PEs. Figure 2.1 also shows the data that is transferred between the PE and the CU.

Algorithm 3 Grouping method

Given particles and weights of the mth PE $\left(x_{k,m}^{(\ell)}, w_{k,m}^{(\ell)}\right)$, $m = 1, \ldots, P$
*F*ind the local extrema at the mth PE
for $m = 1$ to P **do**
  $x_{\min,m} = \min_{\ell} x_{k,m}^{(\ell)}$
  $x_{\max,m} = \max_{\ell} x_{k,m}^{(\ell)}$
  Transmit $x_{\min,m}$ and $x_{\max,m}$ to the CU
end for

*F*ind global extrema in the CU
$x_{\text{Min}} = \min_{m} x_{\min,m}$
$x_{\text{Max}} = \max_{m} x_{\max,m}$
Send x_{Min} and x_{Max} back to the PEs

*D*ivide particles into groups based on global extrema
*C*alculate the averages for each group in the PEs
for $j = 1$ to G **do**
  $x_{\text{mean},j,m} = \frac{1}{N_j} \sum_{\ell \in \text{Group}_j} x_{k,m}^{(\ell)}$
  $w_{\text{mean},j,m} = \frac{1}{N_j} \sum_{\ell \in \text{Group}_j} w_{k,m}^{(\ell)}$
  Send $w_{\text{mean},j,m}$ to the CU
  Calculate replication factor ρ_j based on $w_{\text{mean},j,m}$
  Send ρ_j to each PE (operate in the CU)
end for

2.3.4 Processing element architecture

The PE block diagram is shown in Figure 2.2. The PE processes the input particles and executes the sampling, weighting and IMH sampling steps. After sampling, the particles are stored in the particle memory (PMEM), and the replicated particle index factors are stored in the replicated particle index memory (RMEM). Using the index from RMEM, each PE reads the resampled particles from PMEM, computes the local range factors $\mathbf{x}_{\max,m}$, $\mathbf{x}_{\min,m}$ and transmits them to the CU. After receiving the global range factors $\mathbf{x}_{\text{Min}}$, $\mathbf{x}_{\text{Max}}$, the resampled particles are divided into G groups, and the average particles $\mathbf{x}_{\text{mean},j,m}$ and average weights $w_{\text{mean},j,m}$ for the jth group are calculated. Next, the average weights of each group $w_{\text{mean},j,m}$ are sent to the CU to compute the replication factor ρ_j. The mean particles $\mathbf{x}_{\text{mean},j,m}$ are read from the mean particle memory (MPMEM) and sent to the sampling unit for generating particles in the next time step.

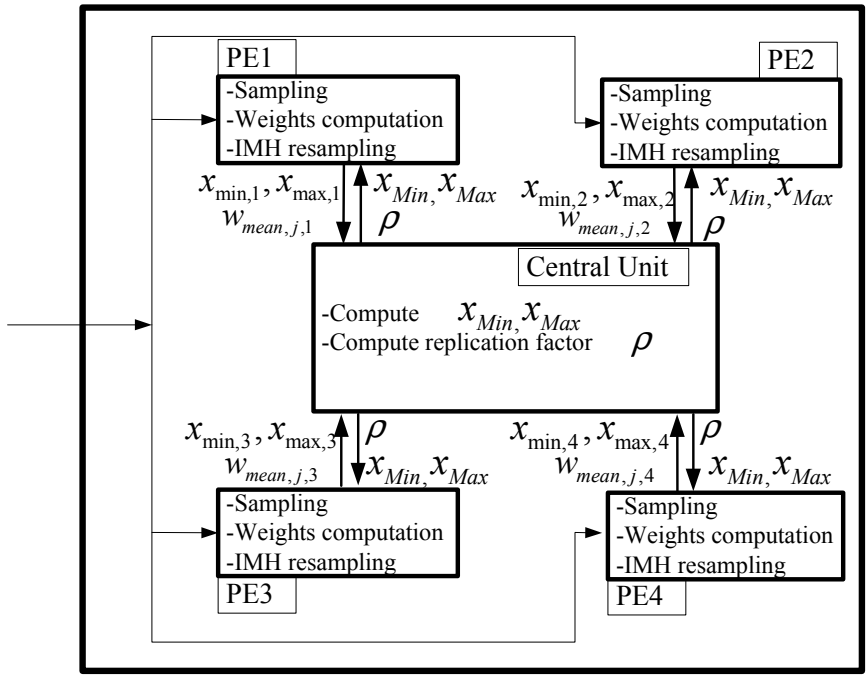


Figure 2.1: PPF-IMH architecture with four PEs: PE1, PE2, PE3, and PE4. The mth PE, $m = 1,\ldots,4$ sends the average weights $w_{\text{mean},j,m}$, local minima $\mathbf{x}_{\min,m}$, local maxima $\mathbf{x}_{\max,m}$ to the CU and CU sends global minima $\mathbf{x}_{\text{Min}}$, global maxima $\mathbf{x}_{\text{Max}}$ and replication factor ρ back to the PEs.

Figure 2.3 shows the IMH sampler architecture. When computing the acceptance probability, we use the modified method in [64] to avoid division computation. In particular, in our case, we accept particles following the procedure

$$\left(\mathbf{x}_k^{(\ell)}, w_k^{(\ell)}\right) = \begin{cases} \left(\tilde{\mathbf{x}}_k^{(\ell)}, \tilde{w}_k^{(\ell)}\right), & u\, w_k^{(\ell-1)} \leq \tilde{w}_k^{(\ell)} \\ \left(\mathbf{x}_k^{(\ell-1)}, w_k^{(\ell-1)}\right), & u\, w_k^{(\ell-1)} > \tilde{w}_k^{(\ell)}, \end{cases}$$

where u is a uniform random variable between 0 and 1. Specifically, the weight of a newly generated particle is first compared with the product of the uniformly distributed random variable u and the weight of the last accepted particle in the chain. If the new particle weight is larger, it remains in the chain and its index is assigned to a new replicate index labeled r_i; otherwise, it is replicated once more, and the replicate index r_i remains unchanged.

The group-and-mean unit is used to divide the particles into different groups, based on the global ranges, and to calculate particle and weight averages in each group. For the one-dimensional problem, the architecture of this unit is shown in Figure 2.4. First, using the global range factors

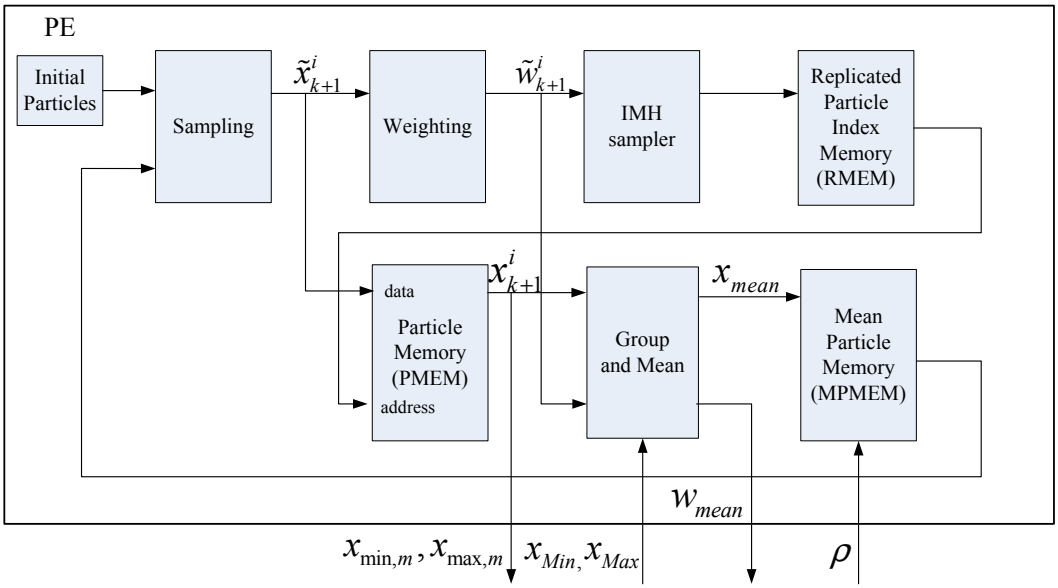


Figure 2.2: Block diagram of a PE.

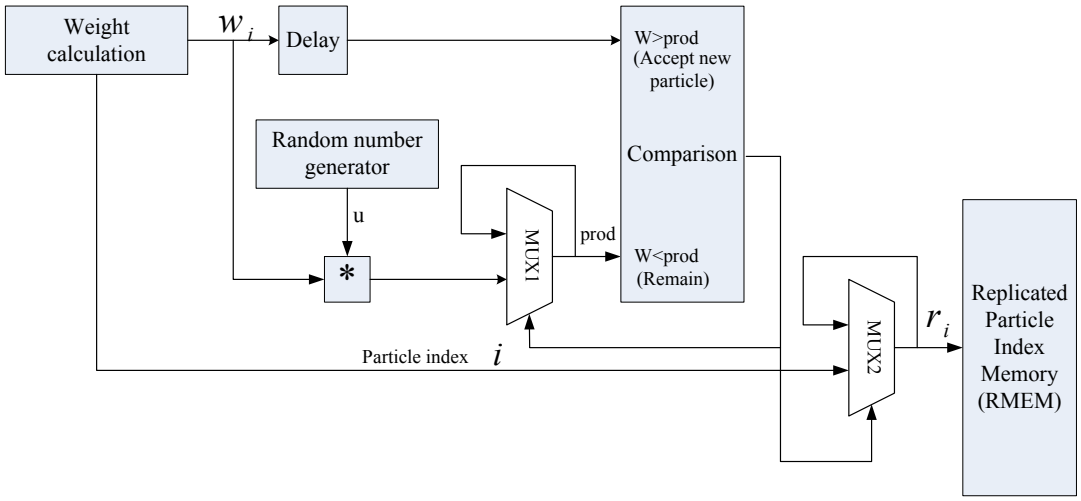


Figure 2.3: Block diagram of the IMH sampler.

x_{Min}, x_{Max}, and the number of groups G, the range for each group, $\delta = (x_{\text{Max}} - x_{\text{Min}})/G$ is computed. Then, the thresholds γ of each group are generated based on δ as $\gamma_j = x_{\text{Min}} + (j-1)\delta$, $j = 1, \ldots, G$. Each particle is then compared to the thresholds and placed in the corresponding group. The particle values are accumulated, and the number of particles is counted in each group. Finally, the mean value $x_{\text{mean},j}$ and the mean weight $w_{\text{mean},j}$ are computed for each group. For the

multi-dimensional problem, since the computations for each dimension are independent, we apply this procedure to each dimension in dimension in parallel.

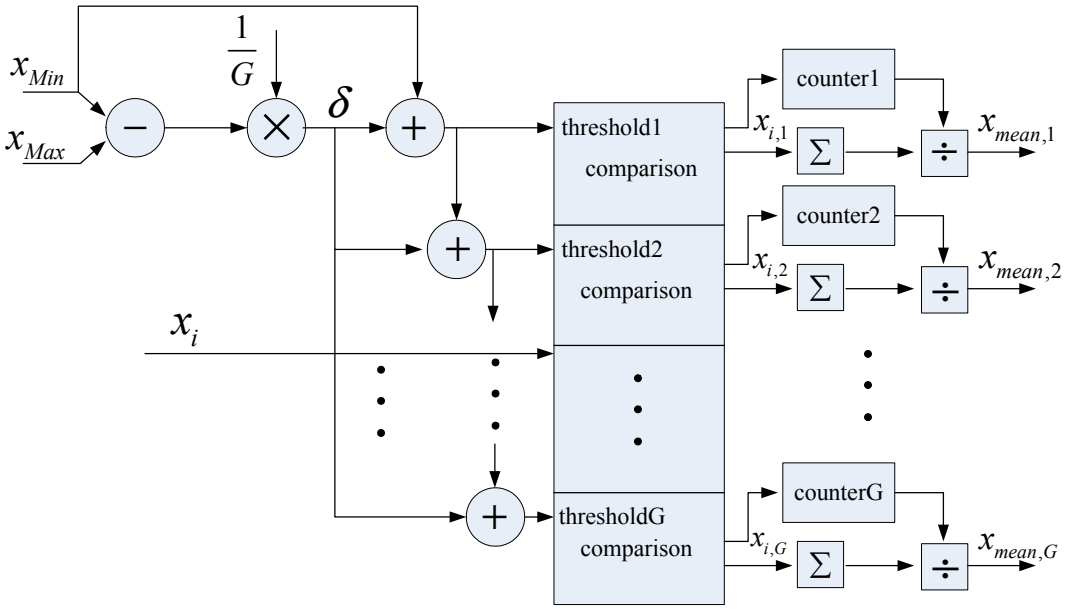


Figure 2.4: Block diagram of the group-and-mean unit.

2.3.5 Central unit architecture

The CU executes global computations such as global range and replication factor computations. Its architecture is shown in Figure 2.5. Two comparators and multiplexers (MUXs) are used to generate $\mathbf{x}_{\text{Min}}$ and $\mathbf{x}_{\text{Max}}$. If the new local range $\mathbf{x}_{\text{Min}}$ is smaller than the last accepted global range $\mathbf{x}_{\text{Min}}$, we assign $\mathbf{x}_{\text{min},m}$ to $\mathbf{x}_{\text{min}}$, or keep the last value of $\mathbf{x}_{\text{Min}}$; a similar procedure is used to find $\mathbf{x}_{\text{Max}}$. We use an accumulator and a multiplier to compute the replication factor. The accumulator inputs, $w_{\text{mean},j,m}$, are normalized to guarantee that $\sum_{j=1}^{G\times\mathfrak{D}} \rho_j = N$. Thus, after each iteration, the number of PE particles is unchanged.

2.4 Algorithm performance results

We demonstrate the performance of our proposed PPF-IMH system using two dynamic state-space examples that have been previously used in the literature for comparison. The first system, state-space Model 1, depends on a one dimensional (1-D) dynamic state parameter x_k and is described

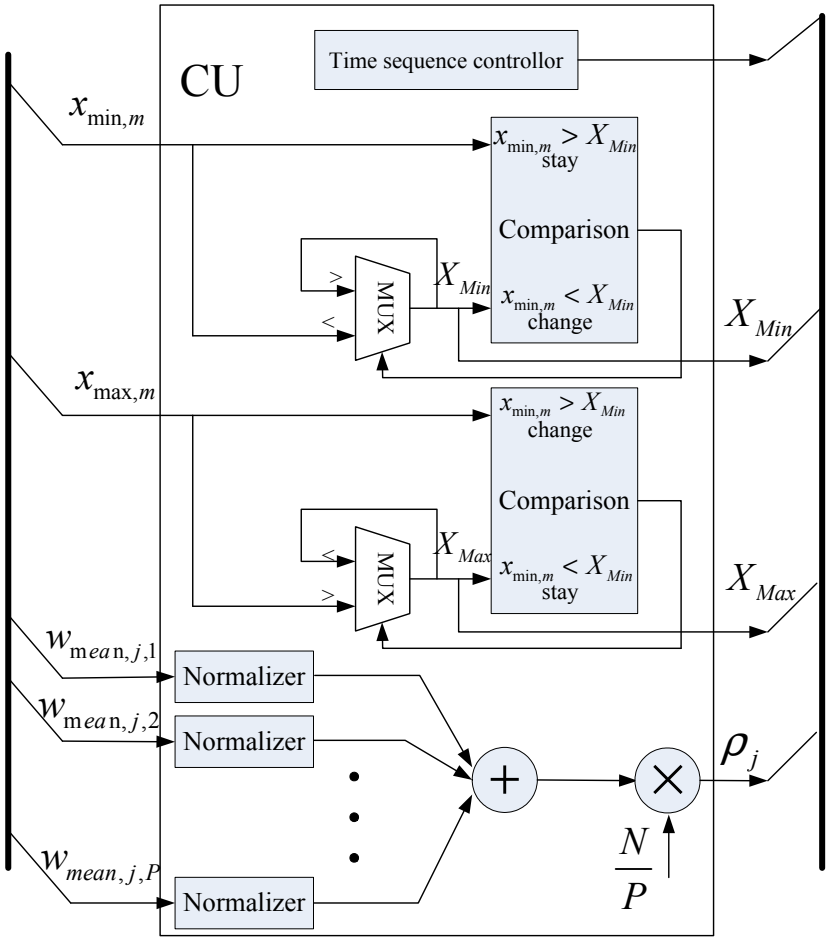


Figure 2.5: Block diagram of the CU.

by the following equations [65]

$$\begin{aligned} x_{k+1} &= 1+\sin(0.04\pi k)+0.5x_k+v_k \\ z_k &= \begin{cases} 0.2x_k^2+n_k, & \text{if } k \le 30 \\ 0.5x_k-2+n_k, & \text{if } k > 30 \end{cases} . \end{aligned} \tag{2.6}$$

Here, v_k is a random process modeled by a Gamma random variable with shape parameter 3 and scale parameter 2, and n_k is a zero-mean, additive white Gaussian noise random variable with variance 10^{-5}. The second example, state-space Model 2, is also a 1-D state space system that is

described by [30]

$$\begin{aligned} x_{k+1} &= 0.5x_k + 25\frac{x_k}{1+x_k^2} + 8\cos(1.2k) + v_k \\ z_k &= \frac{1}{20}x_k^2 + n_k \end{aligned} \tag{2.7}$$

where v_k and n_k are zero-mean, Gaussian random variables with variances $\sigma_v^2 = 10$ and $\sigma_n^2 = 1$, respectively.

The estimation performance is computed using the root mean-squared error (RMSE) as

$$\text{RMSE} = \left(\frac{1}{K} \sum_{k=1}^{K} \frac{1}{\text{MC}} \sum_{l=1}^{\text{MC}} (\hat{\mathbf{x}}_{k,l} - \mathbf{x}_k)^2 \right)^{1/2}.$$

Here, $K = 30$ is the simulation path length, MC $= 100$ is the number of Monte Carlo simulations, $\mathbf{x}_k$ is the true state k and $\hat{\mathbf{x}}_{k,l}$ is the estimated state parameter in the lth Monte Carlo iteration at time k.

2.4.1 Effect of number of groups

In the proposed PPF-IMH algorithm, we divide the particles in each PE into G groups and use the average of each group as the new particle. The choice of G is crucial as it impacts the estimation accuracy. Figure 2.6 shows the RMSE tracking performance with respect to G for Model 1. Here the number of particles is chosen to be 1,000 and 2,000, and the number of PEs is chosen to be 1, 2 and 4. In all cases, we can see that as G increases, the RMSE decreases. But when G is greater than an optimal value G_{opt}, then there is no significant improvement in the RMSE. The G_{opt} value depends on the number of particles in each PE. From Figure 2.6, we can see that for N=1,000 particles, when P=4 PEs, then $G_{\text{opt}} \approx 10$, when P=2 then $G_{\text{opt}} \approx 15$ and when P=1 then $G_{\text{opt}} \approx 20$. For N=2,000 particles, when P=4 then $G_{\text{opt}} \approx 15$; this is similar to the case of N=1,000 and P=2. Furthermore, since for large G the hardware resource utilization is also higher, here we choose G=10.

2.4.2 Estimation performance

We use a parallel architecture with P=4 PEs for numerical simulations, where each PE processes 250 particles. We apply the parallel algorithm in [45] and also the new PPF-IMH algorithm to the systems described by Model 1 and Model 2. The corresponding estimation results are shown

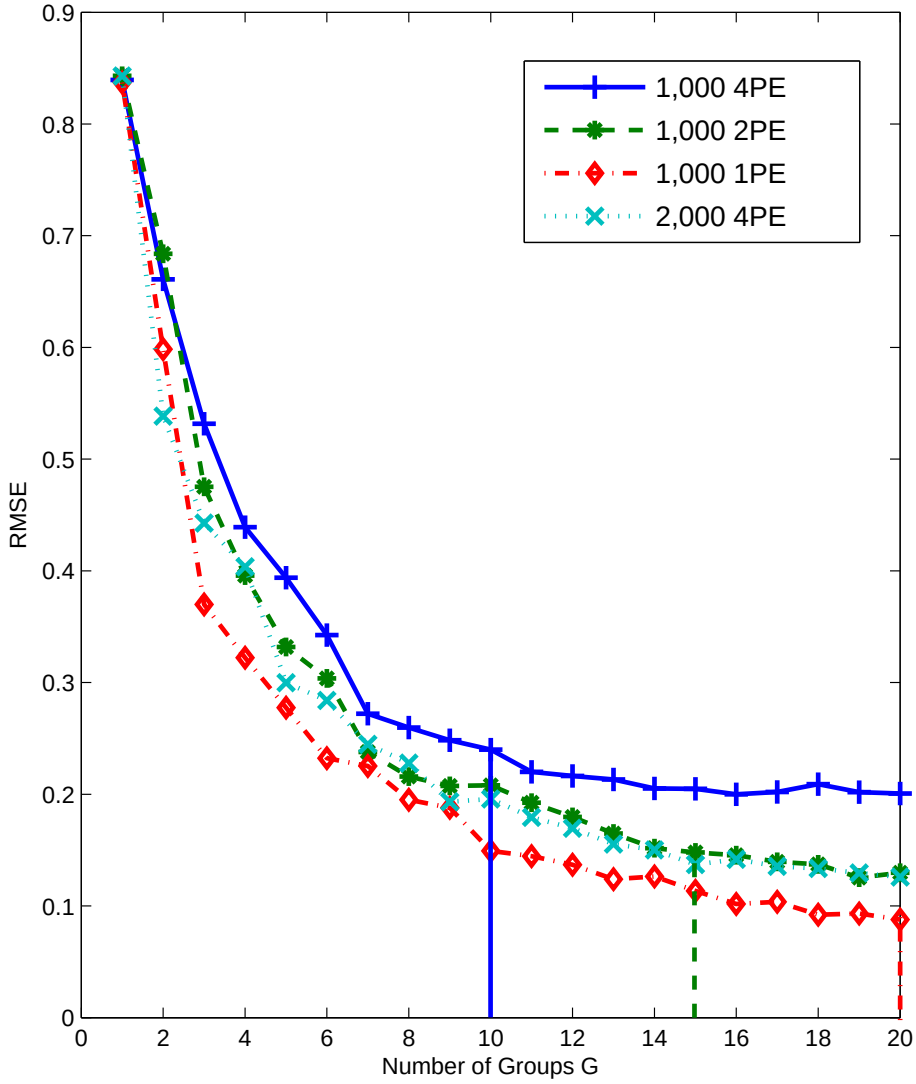


Figure 2.6: RMSE performance for varying number of groups G and number of PEs.

in Figure 2.7 and Figure 2.8. Table 2.1 shows the RMSE performance for the two models. We can see that the RMSE performance of the PPF-IMH algorithm is significantly better than the parallel algorithm in [45] for both models. In addition, the RMSE performance of the PPF-IMH is close to the PF with systematic resampling, which means that the performance degradation due to parallelization in [45] is compensated by IMH resampling.

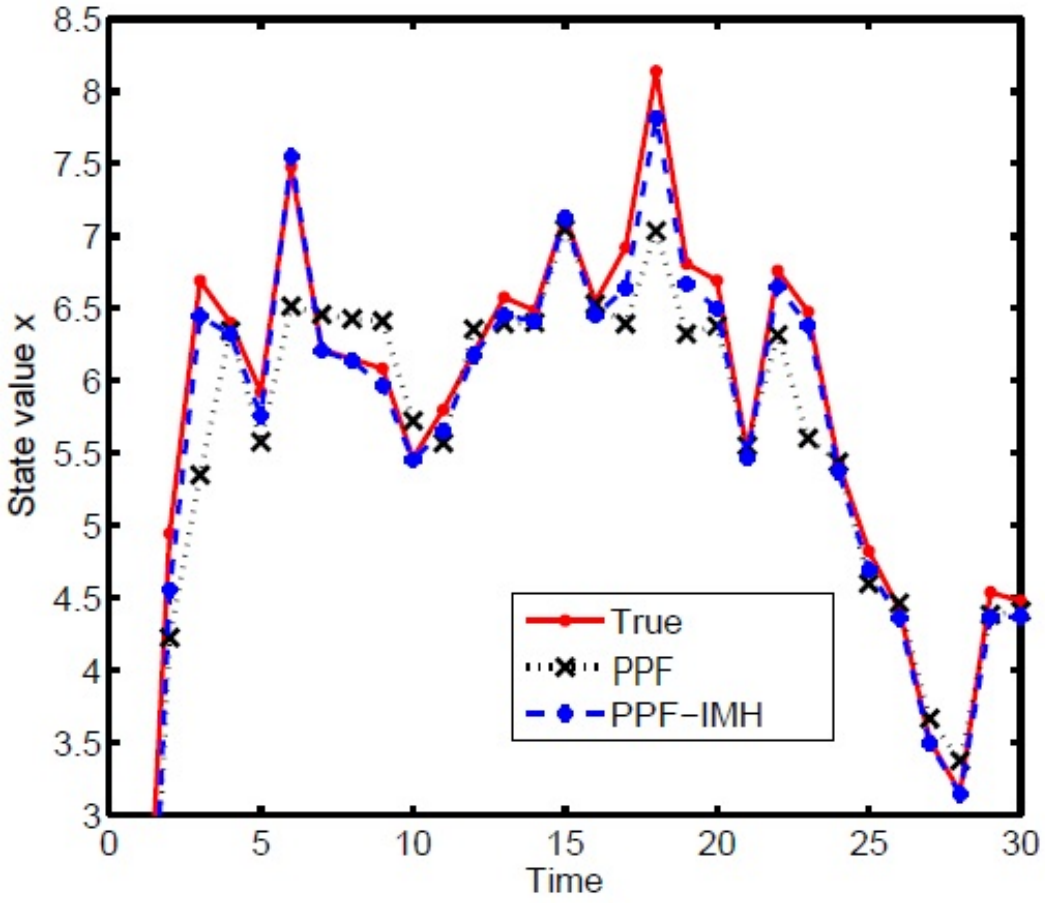


Figure 2.7: Comparison of estimation performance for Model 1 using the PPF algorithm in [45] and the proposed PPF-IMH algorithm.

Table 2.1: Comparison of RMSE performances.

Algorithms	RMSE for Model 1	RMSE for Model 2
Systematic resampling	0.24	4.06
Parallel algorithm in [45]	0.36	6.19
Proposed PPF-IMH algorithm	0.26	4.34

2.5 Hardware performance results

The PPF-IMH hardware architecture for the system state estimation in Model 1 was implemented using Verilog HDL and synthesized on Xilinx Virtex-5 device (XC5VSX240T). The design was verified using Modelsim. Both the P=1 PE serial architecture and the P=4 PE parallel architectures

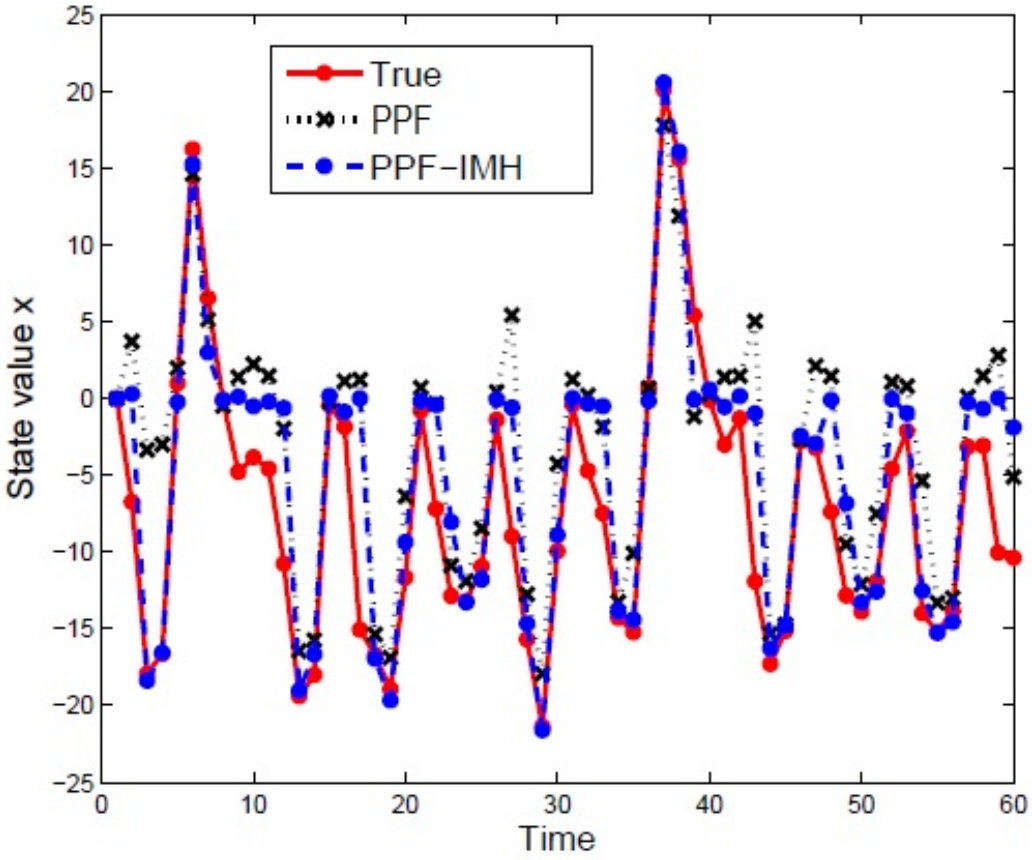


Figure 2.8: Comparison of estimation performance for Model 2 using the PPF algorithm in [45] and the proposed PPF-IMH algorithm.

were implemented. The RMSE values for the P=1 and P=4 PE architectures are 0.2686 and 0.3172, respectively. The RMSE is higher than the MATLAB generated numerical results which is 0.26 for P=4 because of the 14 bits fixed-point FPGA implementation.

2.5.1 Resource utilization

Table 2.2 summarizes the P=1 and P=4 PE architecture resource utilization. The sinusoidal and exponential functions in the system equations described by Model 1 and Model 2 are implemented using CORDIC units, and the rest of the units are implemented using DSP cores. For the P=4 PE implementation, the PE and CU occupied slice utilizations are 408 (1%) and 420 (1%), respectively. Our resource usage is fairly low; for example, only about 5% of the slice resource is used in a Xilinx Virtex-5 FPGA. Thus, such an implementation can support a much larger number of particles or

multiple PFs which are required in biomedical signal processing applications [50].

Table 2.2: Resource utilization comparison.

Unit	Occupied Slices	Slice Registers	Slice LUTs	Block RAM	DSP48Es
P=1 processing element	398 (1%)	1,292 (1%)	1,363 (1%)	5 (1%)	10 (1%)
P=4 processing elements	2,052 (5%)	5,749 (3%)	6,352 (4%)	18 (3%)	46 (4%)

2.5.2 Execution time

Figure 2.9 shows the timing for one iteration of the proposed method for a system using N=1,000 particles and P=4 PEs. For our implementation, $L_s = 21$ cycles is the sampling step delay determined by the sinusoid calculation time, $L_w = 24$ cycles is the weighting latency determined by the time for calculating the exponential functions, $L_r = 2$ cycles is the latency of the global range calculation, $L_m = 18$ cycles is the time for computing the average value, and $L_\rho = 20$ cycles is the time for calculating the replication factor. Thus, one PPF-IMH iteration takes $L_s + L_w + N + L_r + N + L_m + L_\rho$ = 585 cycles. For a system clock rate of 100 MHz, the total processing period for one iteration is T_{total}=5.85 μs.

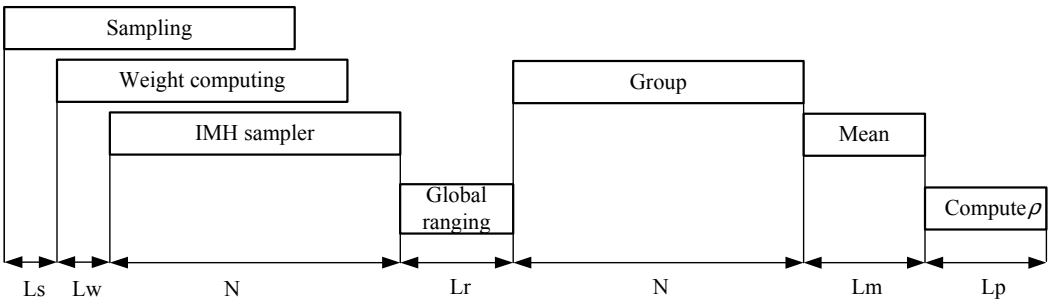


Figure 2.9: Execution time of proposed PPF-IMH method.

2.5.3 Communication overhead

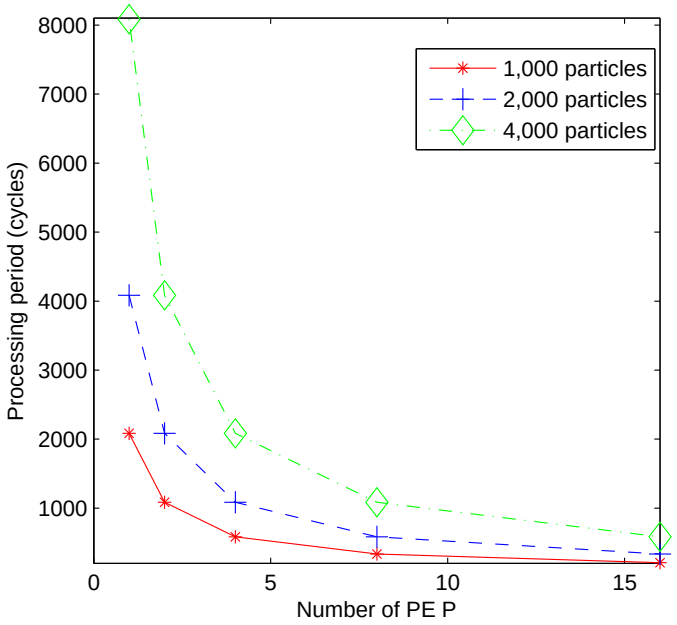
The communication overhead of the proposed algorithm for a system using N=1,000 particles, P=4 PEs and G=10 groups is 96 bytes. This is a significant reduction compared with the traditional algorithm whose communication overhead is 2,500 bytes.

2.5.4 Scalability

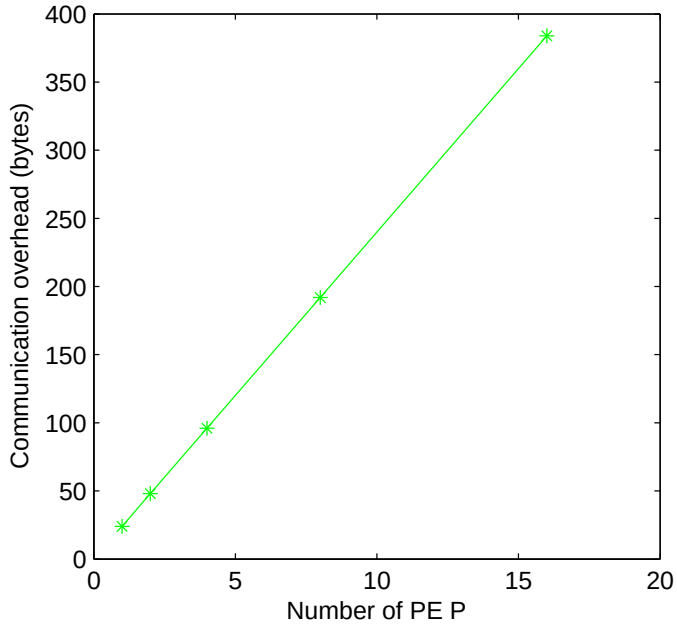
Figures 2.10a and 2.10b show the execution time and communication overhead, respectively, for one processing iteration with respect to P, the number of PEs in the proposed parallel architecture. The processing period curve saturates when P is large because there is no significant speedup when M/P approaches the constant latency L. In this case, the latency is given by $L = L_s + L_w + L_r + L_m + L_\rho = 85$ cycles. From Figure 2.6, the RMSE performance slightly decreases as P increases. Thus, for a PF with $N = 1,000$ particles, the P=4 PE architecture yields high performance.

In many applications such as in biomedical signal processing, the dimension of the state space can be very large [50]. Consequently, a very large number of particles is required for satisfactory performance. In such cases, the processing time can be further reduced by using more PEs. Figure 2.10a shows the processing period for N=2,000 and N=4,000 particles. For these cases, an architecture using P=8 PEs provides a better choice.

From Figure 2.10b, we can see that the communication overhead curve increases linearly with respect to P, and the slope is equal to $2G+4$, where G is the number of groups in each PE. Thus, a lower value of G is more desirable for lower communication overhead. Unfortunately, a lower value of G results in degraded RMSE performance, and thus the choice of G is a a compromise between RMSE performance and communication overhead.



(a)



(b)

Figure 2.10: Scalability of the proposed parallel architecture.

Chapter 3

Application of Parallel Particle Filtering to Waveform-agile Sensing

An important application of the particle filtering (PF) algorithm is waveform-agile sensing, where the waveform is adaptively configured at each time step. The sensing performance has been shown to increase when the parameters of transmitted waveforms (in active sensing) are adaptively designed or the parameters of observed waveforms (in passive sensing) are optimally selected at each time step [46, 47, 48]. However, as the waveform parameters need to be adaptively updated at each time step, the computational complexity of waveform design can be very high. When waveform-agility is integrated into particle filtering, the computational complexity can become unmanageable. However, if the PF can be implemented in parallel, then real-time implementation of adaptive waveform design schemes will become feasible. In Chapter 2, we proposed a new parallel PF with independent Metropolis-Hastings (PPF-IMH) sampling which enables real-time processing of the traditional PF. In this chapter, we apply the proposed PPF-IMH to waveform-agile sensing to achieve real-time tracking of targets with high RMSE tracking performance.

3.1 Waveform-agile sensing algorithm

The dynamic system described by the state-space Equations (2.1) and (2.2) assumes that measurements $\mathbf{z}_k$ are observed at time step k. In certain applications, these measurements could be determined by transmitted waveform $s_k(t;\theta_k)$ with known fixed parameters θ_k. One possible way to improve the estimation performance of the state parameters is to adaptively control the transmit waveform parameters θ_k at each time step k. Specifically, waveform-agile sensing is a closed-loop feedback optimization procedure that allows adaptive selection of the waveform parameters to be transmitted at the next time-step in order to optimize a cost function [46, 47, 48]. As our objective here is to accurately estimate the dynamic state $\mathbf{x}_k$, we can choose the cost function to be the mean-square error (MSE) for the next time step.

We assume that the waveform $s_k(t;\theta_k)$ to be transmitted at time step k has a parameter vector θ_k that can be adaptively selected. The received waveform is analyzed to obtain the measurement vector $\mathbf{z}_k$ in Equation (2.2), and, consequently, the measurement noise vector $\mathbf{n}_k$ in Equation (2.2) is assumed to have a covariance matrix $\mathbf{R}(\theta_k)$ that depends on θ_k. Using $\mathbf{z}_k$, we can obtain an estimate

of the target state, $\hat{\mathbf{x}}_k$; thus the estimation error depends on the choice of θ_k. The proposed PPF-IMF approach can be applied to provide an efficient implementation of waveform-agile sensing. In particular, we will use the proposed PPF-IMF formulation to draw particles $\mathbf{x}_k^{(\ell)}$ from an importance density $q(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)}, \mathbf{z}_k, \theta_0, \ldots, \theta_k)$, estimate the posterior probability density function and adaptively choose the waveform parameter $\hat{\theta}_k$ that optimizes the predicted MSE in estimating $\mathbf{x}_k$ [47].

The covariance matrix for the target state estimate at time step k is given by

$$\mathbf{P}(\theta_k) = E_{\mathbf{x}_k, \mathbf{z}_k|\mathbf{z}_{1:k-1}}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T(\mathbf{x}_k - \hat{\mathbf{x}}_k)]$$

where $E[\cdot]$ is the expectation operator and $\hat{\mathbf{x}}_k$ is the estimate of $\mathbf{x}_k$ given the measurement sequence $\mathbf{z}_{1:k-1}$. The covariance matrix of error estimation can be approximated by the posterior Cramér-Rao lower bound (PCRLB) [66, 67, 68, 69]

$$\mathbf{P}(\theta_k) \approx \text{PCRLB}(\theta_k) \tag{3.1}$$

that depends on the waveform parameter vector θ_k. The PCRLB can be computed from the predicted Fisher information matrix $\mathbf{I}_k$ using [66]

$$\text{PCRLB}(\theta_k) = \mathbf{I}_k^{-1}(\theta_k).$$

For a system with linear state-transition model, Equation (2.2) can be rewritten as $\mathbf{x}_k = \mathbf{F}\,\mathbf{x}_{k-1} + \nu_k$ and the process noise ν_k has covariance matrix given by $\mathbf{Q}$. In this case, the predicted Fisher information matrix $\mathbf{I}_k$ can be represented as

$$\begin{aligned}
\mathbf{I}_k(\theta_k) &= \mathbf{Q}^{-1} + E\left[\tilde{\mathbf{H}}_k^T \mathbf{R}^{-1}(\theta_k)\tilde{\mathbf{H}}_k\right] - \mathbf{Q}^{-1}\mathbf{F}\left(\mathbf{I}_{k-1}(\theta_{k-1}) + \mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}\right)^{-1}\mathbf{F}^T\mathbf{Q}^{-1} \\
&= \left(\mathbf{Q}^{-1} + F^T\mathbf{I}_{k-1}^{-1}(\theta_{k-1})\mathbf{F}\right)^{-1} + E\left[\tilde{\mathbf{H}}_k^T \mathbf{R}^{-1}(\theta_k)\tilde{\mathbf{H}}_k\right].
\end{aligned}$$

Here, $\tilde{\mathbf{H}}_{k+1} = [\nabla_{\mathbf{x}_{k+1}} h_{k+1}(\mathbf{x}_{k+1})]^T$, where $h(\cdot)$ is the measurement function given in Equation (2.2) and ∇ denotes gradient operation. The specific representation of the measurement noise covariance $\mathbf{R}$ is related to the waveform type parameters, and will be describe in Equation (3.5). Thus, the covariance matrix of error estimation can be calculated iteratively as

$$\mathbf{P}(\theta_k) \approx \left(\left(\mathbf{Q}^{-1} + \mathbf{F}^T\mathbf{P}(\theta_{k-1})\mathbf{F}\right)^{-1} + E\left[\mathbf{H}_k\mathbf{R}^{-1}(\theta_k)\mathbf{H}_k\right]\right)^{-1} \tag{3.2}$$

where $\mathbf{H}_k = \nabla_{\mathbf{x}_k} h_k(\mathbf{x}_k)$.

The optimal waveform to be transmitted at the next time step is then obtained by optimizing the predicted MSE using $\mathbf{P}(\theta_k)$ in Equation (3.2). The waveform-agile sensing problem can thus be stated as the selection of the waveform parameter

$$\hat{\theta}_k = \arg\min_{\theta_k} \mathrm{Tr}\big(\mathbf{P}(\theta_k)\big)\,,$$

where $\hat{\theta}_k$ is the optimally chosen waveform parameter vector and $\mathrm{Tr}(\cdot)$ is the matrix trace.

3.2 Waveform-agile tracking application

We consider a waveform-agile tracking application problem, where a target's position and velocity in a two dimensional (2-D) Cartesian coordinate system need to be estimated. The target is tracked using a phased-array radar system, transmitting waveforms from a class of generalized frequency-modulated (GFM) waveforms with Gaussian envelope [70]. A Gaussian-windowed GFM waveform, at time step k, is given by

$$s_k(t;\theta_k) = (\pi\alpha_k^2)^{-1/4}\; e^{-0.5(t/t_r)^2/\alpha_k^2}\; e^{j2\pi\beta_k\xi(t/t_r)}, \tag{3.3}$$

where α_k is the shape parameter of the Gaussian envelope, β_k is the frequency modulation (FM) rate, $\xi(t/t_r)$ is the time-varying phase function, and $t_r = 1$ s is a reference time. The waveform parameter vector that can be configured is given by $\theta_k = [\alpha_k \;\; \beta_k]^T$. An example of waveforms we use are linear FM (LFM) waveforms; these are waveforms with quadratic phase function $\xi(t/t_r) = (t/t_r)^2$.

The target state at time k can be represented as $\mathbf{x}_k = [x_k \; y_k \; \dot{x}_k \; \dot{y}_k]^T$, where (x_k, y_k) and $(\dot{x}_k, \dot{y}_k)$ are the position and velocity of the target, respectively, in 2-D Cartesian coordinates. For this system, the state-transition is linear, so Equation (2.1) can be rewritten as $\mathbf{x}_k = \mathbf{F}\;\mathbf{x}_{k-1} + \nu_k$. Here, the process noise ν_k has covariance matrix given by $\mathbf{Q}$. The state transfer function $\mathbf{F}$ and $\mathbf{Q}$ are given by

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \delta_t & 0 \\ 0 & 1 & 0 & \delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Q} = q \begin{bmatrix} \delta_t^2/3 & 0 & \delta_t^2/2 & 0 \\ 0 & \delta_t^2/3 & 0 & \delta_t^2/2 \\ \delta_t^2/2 & 0 & \delta_t & 0 \\ 0 & \delta_t^2/2 & 0 & \delta_t \end{bmatrix} \tag{3.4}$$

where δ_t is the step interval and q is the intensity factor. If the radar is located at position (0, 0), the nonlinear relation between $\mathbf{x}_k$ and $\mathbf{z}_k$ is given by

$$\begin{aligned}\mathbf{z}_k &= [r_k \;\; \dot{r}_k \;\; \varphi_k]^T + \mathbf{n}_k = h(\mathbf{x}_k) + \mathbf{n}_k \\ &= \left[\left(x_k^2 + y_k^2\right)^{1/2} \quad (\dot{x}_k x_k + \dot{y}_k y_k)/r_k \quad \arctan\left(y_k/x_k\right)\right]^T + \mathbf{n}_k\end{aligned}$$

where $\mathbf{n}_k$ is measurement noise with zero mean and covariance matrix $\mathbf{R}_k(\theta_k)$.

In order to compute the covariance matrix of error estimation in (3.2), we first compute $\tilde{\mathbf{H}}_k = [\nabla_{\mathbf{x}_k} h_k(\mathbf{x}_k)]^T$ as

$$\nabla_{\mathbf{x}_k} h_k(\mathbf{x}_k) = \begin{bmatrix} \frac{\partial y_k}{\partial x_k} & \frac{\partial \dot{r}_k}{\partial x_k} & \frac{\partial \varphi_k}{\partial x_k} \\ \frac{\partial y_k}{\partial y_k} & \frac{\partial \dot{r}_k}{\partial y_k} & \frac{\partial \varphi_k}{\partial y_k} \\ \frac{\partial y_k}{\partial \dot{x}_k} & \frac{\partial \dot{r}_k}{\partial \dot{x}_k} & \frac{\partial \varphi_k}{\partial \dot{x}_k} \\ \frac{\partial y_k}{\partial \dot{y}_k} & \frac{\partial \dot{r}_k}{\partial \dot{y}_k} & \frac{\partial \varphi_k}{\partial \dot{y}_k} \end{bmatrix} = \begin{bmatrix} \frac{2x_k}{cy_k} & \frac{2f_c}{c}\left(\frac{\dot{x}_k}{y_k} - \frac{\dot{r}_k x_k}{y_k^2}\right) & -\frac{y_k}{y_k^2} \\ \frac{2y_k}{cy_k} & \frac{2f_c}{c}\left(\frac{\dot{y}_k}{y_k} - \frac{\dot{r}_k y_k}{y_k^2}\right) & \frac{x_k}{y_k^2} \\ 0 & \frac{2f_c}{c}(x_k/y_k) & 0 \\ 0 & \frac{2f_c}{c}(y_k/y_k) & 0 \end{bmatrix}$$

where f_c is the carrier frequency of the waveform and c is the waveform speed of propagation in the medium. Assuming high signal-to-noise ratio (SNR), the noise covariance matrix $\mathbf{R}(\theta_k)$ can be approximated by the CRLB, which corresponds to a 3×3 matrix that for the GFM waveform in (3.3) is given by [70]

$$\mathbf{R}(\theta_k) = \eta_k \begin{bmatrix} \frac{1}{2\alpha_k^2} + g(\beta_k) & 2\pi d(\beta_k) & 0 \\ 2\pi d(\beta_k) & (2\pi)^2 \alpha_k^2/2 & 0 \\ 0 & 0 & \psi \end{bmatrix}. \tag{3.5}$$

Here, η_k is the SNR, ψ is determined by the radar array properties and is independent of waveform parameter θ_k and

$$\begin{aligned} g(\beta_k) &= (2\pi\beta_k)^2 \int_{-\infty}^{\infty} \frac{1}{\alpha_k\sqrt{\pi}} \exp\left(-t^2\, \xi^2(t)/\alpha_k^2\right) dt \\ d(\beta_k) &= (2\pi\beta_k)^2 \int_{-\infty}^{\infty} \frac{t}{\alpha_k\sqrt{\pi}} \exp\left(t^2\, \xi'(t)/\alpha_k^2\right) dt\,. \end{aligned}$$

We can optimally choose the waveform parameter θ_k to minimize the predicted MSE (PMSE) using the following three steps.

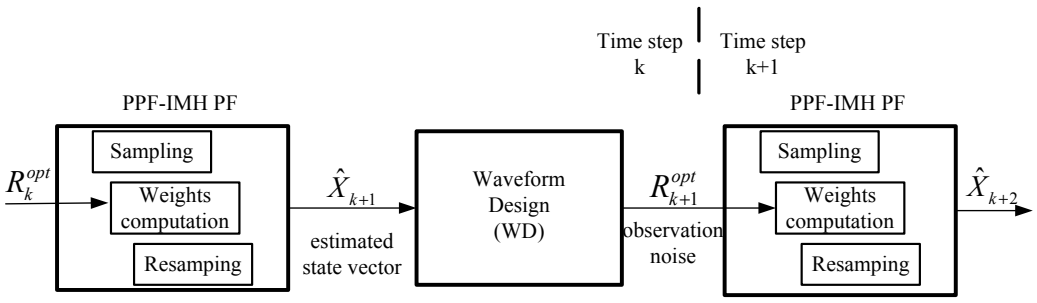


Figure 3.1: Block diagram of tracking with waveform design.

- *Prediction*: Predict the target state at time k as $\tilde{\mathbf{x}}_k = \mathbf{F}\hat{\mathbf{x}}_{k-1}$, where $\hat{\mathbf{x}}_{k-1}$ is the estimated state at time $(k-1)$ using the PPF-IMH algorithm.

- *Optimization*: Use $\tilde{\mathbf{x}}_k$ to calculate $E[\tilde{\mathbf{H}}_k^T \mathbf{R}^{-1}(\theta_k)\tilde{\mathbf{H}}_k] \approx \mathbf{H}_k^T(\tilde{\mathbf{x}}_k)\mathbf{R}^{-1}(\theta_k)\mathbf{H}_k(\tilde{\mathbf{x}}_k)$, where $\mathbf{H}_k = \nabla_{\mathbf{x}_k} h_k(\mathbf{x}_k)$ .. Calculate PMSE(θ_k) for every possible waveform parameter and choose θ_k^{opt}, which minimizes PMSE(θ_k).

- *Updating*: Update observation noise covariance $\mathbf{R}(\theta_k^{\text{opt}})$.

We can see that the computational complexity of the waveform design method is fairly high as 2 matrix additions, 5 matrix multiplications and 5 matrix inversions (including a 4×4 matrix inversion) are included for each waveform parameter set. In the next section, we will modify the algorithm and make it amenable for FPGA hardware implementation.

3.3 FPGA implementation of waveform design

The overall block diagram of the hardware architecture for waveform design is shown in Figure 3.1. It consists of a PPF-IMH PF unit (described in Section 2.3.2) and a waveform design unit. At each time step k, we use the PPF-IMH to obtain $\hat{\mathbf{x}}_k$, which is the estimation of $\mathbf{x}_k$ given measurements $\mathbf{z}_0$ to $\mathbf{z}_k$. Waveform design steps such as prediction, optimization and updating, are operated in the waveform design unit.

The waveform design block diagram is shown in Figure 3.2. The most computationally intensive step is the optimization. In the original optimization method, we find $\mathbf{R}(\theta_k^{\text{opt}})$ by

$$\theta_k^{\text{opt}} = \min_{\theta_k} \text{Tr}\left\{ \left((\mathbf{Q}^{-1} + \mathbf{F}^T \mathbf{I}_{k-1}^{-1}(\theta_{k-1})\mathbf{F})^{-1} + \tilde{\mathbf{H}}_k^T \mathbf{R}^{-1}(\theta_k)\tilde{\mathbf{H}}_k \right)^{-1} \right\}.$$

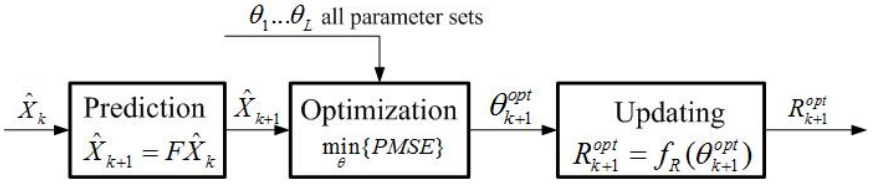


Figure 3.2: Architecture of waveform design unit.

This involves a 4×4 matrix inversion, which is difficult to implement in hardware. Using Woodbury's matrix identity [71], we modify the algorithm in order to reduce the computational complexity as

$$\begin{aligned}\theta_k^{\text{opt}} &= \min_{\theta_k} \text{Tr}\left\{\left(\left(\mathbf{Q}^{-1} + \mathbf{F}^T \mathbf{I}_{k-1}^{-1}(\theta_{k-1})\mathbf{F}\right)^{-1} + \tilde{\mathbf{H}}_k^T \mathbf{R}^{-1}(\theta_k)\tilde{\mathbf{H}}_k\right)^{-1}\right\} \\ &= \min_{\theta_k} \text{Tr}\left\{\mathbf{C} + \mathbf{C}\tilde{\mathbf{H}}^T\left(-\mathbf{R}(\theta_k) - \tilde{\mathbf{H}}_k \mathbf{C}\tilde{\mathbf{H}}_k^T\right)^{-1}\tilde{\mathbf{H}}_k\mathbf{K}\right\} = \min_{\theta_k} \text{Tr}\left\{\mathbf{C}\tilde{\mathbf{H}}_k^T \mathbf{D}^{-1}\tilde{\mathbf{H}}_k\mathbf{C}\right\}.\end{aligned}$$

Here, $\mathbf{C}$ is a symmetric matrix that is given by

$$\mathbf{C} = \mathbf{Q}^{-1} + \mathbf{F}^T \mathbf{I}_{k-1}^{-1}(\theta_{k-1})\mathbf{F} = \begin{bmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \\ b & 0 & d & 0 \\ 0 & b & 0 & d \end{bmatrix},$$

where a, b and d do not depend on θ_k and $\mathbf{D} = -\mathbf{R}(\theta_k) - \tilde{\mathbf{H}}_k \mathbf{C}\tilde{\mathbf{H}}_k^T$ is a 3×3 matrix. As a result, we simplify the 4×4 matrix inverse problem into a 3×3 matrix inverse problem. Furthermore, using

$$\tilde{\mathbf{H}}_{k+1}\mathbf{C}\tilde{\mathbf{H}}_{k+1}^T = \begin{bmatrix} 4a/c^2 & 4bf_c/c^2 & 0 \\ 4bf_c/c^2 & 4df_c{}^2/c^2 & 0 \\ 0 & 0 & a/r_{k+1}^2 \end{bmatrix},$$

and by substituting $\mathbf{R}(\theta_k)$, and simplifying the matrix computation, we obtain

$$\mathbf{D} = -\begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & 0 \\ 0 & 0 & B \end{bmatrix} = -\begin{bmatrix} 2/\alpha_k + 4a/c^2 & 4bf_c/c^2 & 0 \\ 4bf_c/c^2 & 2\alpha_k + 4df_c^2/c^2 & 0 \\ 0 & 0 & \psi + a/y_k^2 \end{bmatrix}.$$

The inverse matrix can be represented as

$$\mathbf{D}^{-1} = -\begin{bmatrix} A_{22}/\zeta & -A_{12}/\zeta & 0 \\ -A_{12}/\zeta & A_{11}/\zeta & 0 \\ 0 & 0 & 1/B \end{bmatrix}$$

where $\zeta = |A_{11}A_{22} - A_{12}^2|$. Thus, the 3×3 matrix inversion requires only 13 multipliers, 5 adders and 2 dividers.

3.4 Waveform target tracking algorithm simulation

The simulation setup consists of a single target moving in a 2-D plane. The initial position and velocity of the target are $\mathbf{x}_0 = [5000 \quad 5000 \quad 100 \quad 100]^T$. We set the waveform parameters to $10^6 < \alpha_k < 10^{14}$ and $\beta_k = 0$. For the case without waveform design, we choose mid range value $\alpha_k = 10^9$. We use N=1,000 particles to track the target. The tracking results with and without waveform design for the x and y positions are shown in Figures 3.3a and 3.3b, and Table 3.1 compares the tracking RMSE. We can see that the tracking performance with waveform design is much better and the RMSE is improved by about 10 times for the x and y position estimations.

Table 3.1: Comparison of RMSE performances.

State Parameter	Numerical simulation without Waveform-agility	Numerical simulation with Waveform-agility	FPGA implementation with Waveform-agility
x-position	141.82	15.12	37.57
y-position	161.52	13.91	33.27
$\dot{x}$-velocity	30.53	17.46	22.56
$\dot{y}$-velocity	37.00	16.18	20.23

3.5 Target tracking hardware synthesis results

The waveform radar tracking hardware architecture described in Section 3.2 is implemented using Verilog HDL and synthesized on a Xilinx Virtex-5 device (XC5VSX240T). The design was also verified using Modelsim. Here, we use a P=4 PEs PPF-IMH parallel architecture for a N=1,000 particle system. The particle weights are represented using 18-bit fixed-point. The target tracking result of the FPGA implementation is shown in Figure 3.4 to match well with the simulation results.

The RMSE results from hardware experiments are shown in Table 3.1. Use of fixed-point data format degrades the performance since extremely small values are determined to be zero.

3.5.1 Resource utilization

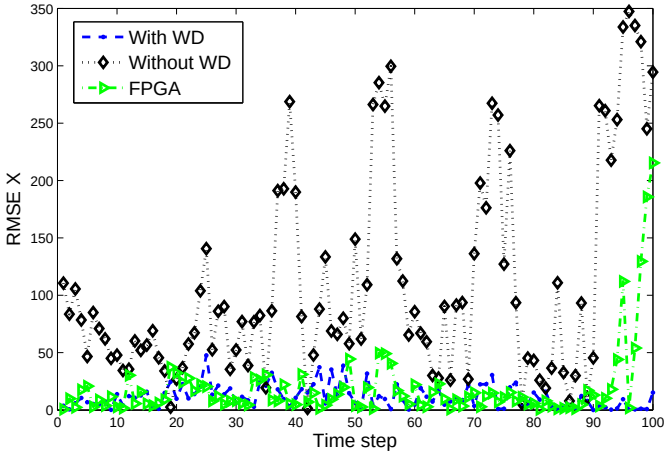
Table 3.2 summarizes the resource utilization for the waveform design unit and the $P = 4$ PEs parallel architecture. The sinusoidal and exponential functions are implemented using CORDIC units, other calculations are implemented using DSP cores. We can see that the hardware resource utilization rate is fairly low; only about 10% of the total hardware resource is used. Thus, 10 such architectural units should be able to fit onto a single Xilinx Virtex-5 platform.

Table 3.2: Resource utilization on Xilinx XC5VSX240T.

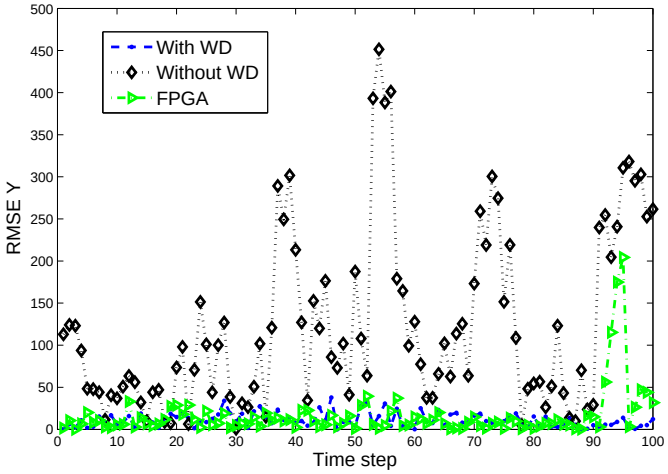
Unit	Occupied Slices	Slice Registers	Slice LUTs	Block Ram	DSP48Es
Waveform design part	673 (1%)	735 (1%)	2229 (1%)	3 (1%)	45 (4%)
P=4 processing elements	3,261 (8%)	7,590 (5%)	10,710 (7%)	48 (9%)	96 (9%)

3.5.2 Execution Time

Figure 3.4 shows the timing chart for one iteration of the proposed radar target tracking system. We can see that additional L_{WA} cycles are needed to obtain the optimal waveform parameter. In our design, $L_{\mathrm{WA}} = 59$. In addition, $L_s = 4$ is the latency of the sampling step, $L_w = 56$ is the weighting latency determined by the computing cycles of the exponential functions, $L_r = 2$ is the latency of the global range calculation, $L_m = 29$ is the time to compute the average value and $L_\rho = 34$ is the latency for calculating the replication factor. Thus, one iteration takes $L_s + L_w + N + L_r + N + L_m + L_\rho + L_{\mathrm{WA}}$=684 cycles. For a system clock rate of 100 MHz, the total processing period for one iteration is $T_{\mathrm{total}} = 6.84\ \mu$s.



(a)



(b)

Figure 3.3: RMSE of the (a) x-position and (b) y-position at each time step, demonstrating the improvement in performance when the waveform is adaptively selected at each time step.

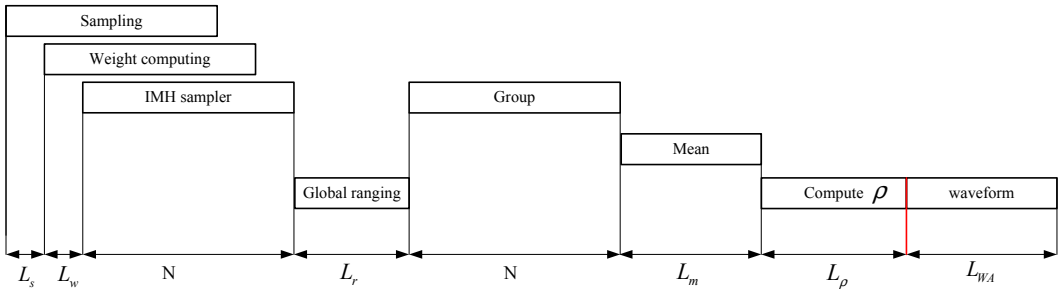


Figure 3.4: Execution time of waveform radar tracking problem.

Chapter 4

Tracking Neural Activity Using Particle Filtering

In Chapter 2, we proposed a new PPF-IMH algorithm to estimate the state of dynamic system in real-time. In this chapter, we investigate the application of PPF-IMH in biomedical signal processing, especially in tracking the positions and moments of neural sources.

EEG and MEG are two techniques used to measure the neural activities in the brain. Compared to fMRI, EEG and MEG offer superior temporal resolution which allows study of the dynamics of neural activities that occur on the order of tens of milliseconds [10]. Localizing and tracking localized current dipoles using EEG/MEG measurements can provide very useful information during brain surgery for patients with medically uncontrolled partial seizures [2]. Although EEG/MEG yield high temporal resolution, high spatial resolution can be achieved by finding an accurate solution to the EEG/MEG inverse problem which is the estimation of the localized current dipole model from EEG/MEG measurements. In this chapter, we present a real-time solution of the EEG/MEG inverse problem based on the PPF-IMH algorithm. The proposed algorithm has good RMSE tracking performance and also increased processing speed. We also implement the resulting algorithm on a Xilinx Virtex-5 FPGA platform to demonstrate its applicability to real-time EEG/MEG tracking in systems.

4.1 EEG and MEG inverse problems and dipole source model

EEG and MEG have had success as clinical tools in localizing neural electrical activities by solving the so-called "electromagnetic inverse problem". Given a set of EEG or MEG signals from an array of external sensors, the inverse problem is to estimate the properties of the current sources in the brain that produced these signals. In this section, the dipole source model for EEG and MEG inverse problem is presented.

Analytic solutions for the MEG/EEG source localization problem, based on current dipole source models, can be obtained when the head is assumed to consist of nested concentric spheres of constant conductivity [1, 15, 19, 20, 21]. Following this model, the primary current $I_k(\mathbf{r})$ at time

k can be represented in terms of N_d current dipoles as [72]

$$I_k(\mathbf{r}) = \sum_{j=1}^{N_d} \mathbf{m}_{k,j}\ \delta(\mathbf{r} - \mathbf{r}_{k,j}),$$

where $\mathbf{r}_{k,j} = [r_{k,j}^{(x)}\ \ r_{k,j}^{(y)}\ \ r_{k,j}^{(z)}]^{\mathrm{T}}$ and $\mathbf{m}_{k,j} = [m_{k,j}^{(x)}\ \ m_{k,j}^{(y)}\ \ m_{k,j}^{(z)}]^{\mathrm{T}}$ are the three-dimensional (3-D) location and moment vectors, respectively, in Cartesian coordinates, of the jth current dipole, $j = 1,\ldots,N_d$, at time k, $k = 1,\ldots,K$. The moment of the jth dipole is given by

$$\mathbf{m}_{k,j} \ = \ \mathbf{q}_{k,j}\, s_{k,j},$$

where $\mathbf{q}_{k,j} = [q_{k,j}^{(x)}\ \ q_{k,j}^{(y)}\ \ q_{k,j}^{(z)}]^{\mathrm{T}}$ and $s_{k,j}$ are the orientation vector and amplitude of the dipole . The MEG/EEG signals acquired by M sensors can be represented as [1]

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{n}_k = \mathbf{A}_k\, \mathbf{s}_k + \mathbf{n}_k, \tag{4.1}$$

where $\mathbf{z}_k = [z_{k,1}\ z_{k,2}\ \ldots\ z_{k,M}]^{\mathrm{T}}$ and $\mathbf{n}_k = [n_{k,1}\ n_{k,2}\ \ldots\ n_{k,M}]^{\mathrm{T}}$ are the $M \times 1$ signal and measurement noise vectors, respectively, from the M sensors, $\mathbf{x}_k = [\mathbf{x}_{k,1}^{\mathrm{T}}\ \mathbf{x}_{k,2}^{\mathrm{T}}\ \ldots\ \mathbf{x}_{k,N_d}^{\mathrm{T}}]^{\mathrm{T}}$ is the $N_d \times 1$ dipole source parameters vector, and $\mathbf{x}_{k,j} = [\mathbf{r}_{k,j}^{\mathrm{T}}\ \mathbf{q}_{k,j}^{\mathrm{T}}\ s_{k,j}]^{\mathrm{T}}$ is the 7-D parameter vector consisting of the 3-D location $\mathbf{r}_{k,j}$, 3-D orientation $\mathbf{q}_{k,j}$ and 1-D amplitude $s_{k,j}$ of the jth dipole source. We assume that the dipole sources are mutually independent of each other and statistically independent of the noise $\mathbf{n}_k$, and that the noise components $n_{k,m}$, $m = 1,\ldots,M$, are mutually independent as in [1]. Also in Equation (4.1), $\mathbf{s}_k = [s_{k,1}\ s_{k,2}\ \ldots\ s_{k,N_d}]^{\mathrm{T}}$ is the amplitude vector corresponding to the N_d dipoles and $\mathbf{A}_k$ is the $M \times N_d$ lead-field matrix that depends on the jth dipole location $\mathbf{r}_{k,j}$ and orientation $\mathbf{q}_{k,j}$. Note that MEG/EEG systems have different lead-field matrices. Specifically, the (m,j)th element $a_{k,m,j}$, $j = 1,\ldots,N_d$, $m = 1,\ldots,M$, of the lead-field matrix $\mathbf{A}_k$ at time step k of an EEG model is given by [8]

$$\begin{aligned} a_{k,m,j} = {} & \frac{1}{4\pi\sigma}\cos(\alpha_{k,j})\left[\frac{2}{d_{k,m,j}^3}\Big(|\mathbf{r}_{k,j}|\cos(\gamma_{k,m,j}) - r\Big) + \big(d_{k,m,j}\,|\mathbf{r}_{k,j}|\big)^{-1} - \big(r\,|\mathbf{r}_{k,j}|\big)^{-1}\right] \\ & + \frac{1}{4\pi\sigma}\sin(\alpha_{k,j})\cos(\beta_{k,j})\sin(\gamma_{k,m,j})\left[\frac{2r}{d_{k,m,j}^3} + \frac{d_{k,m,j} + r}{r\,d_{k,m,j}\,(r - |\mathbf{r}_{k,j}| + d_{k,m,j})}\right]. \end{aligned} \tag{4.2}$$

The corresponding dipole source model is given in Figure 4.1. Here r is the radius of the head model, $d_{k,m,j}$ is the distance between the jth dipole source and the mth sensor, $\gamma_{k,m,j}$ is the angle between the vector pointing to the mth sensor and the vector pointing to the jth dipole location, $\alpha_{k,j}$

is the angle between the jth dipole orientation and the vector pointing to the jth dipole location, $\beta_{k,j}$ is the angle between the plane formed by the jth dipole and the origin, and σ is the head tissue conductivity constant. Also, $|\mathbf{r}_{k,j}| = \left[(r_{k,j}^{(x)})^2 + (r_{k,j}^{(y)})^2 + (r_{k,j}^{(z)})^2\right]^{1/2}$ in Equation (4.2).

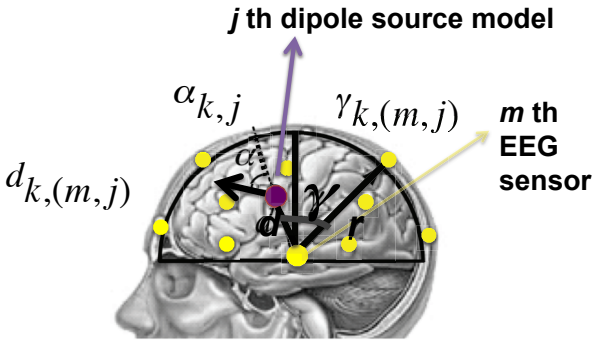


Figure 4.1: Equivalent current dipole model for EEG/MEG localization for the jth dipole source and the mth EEG/MEG sensor. Here, $d = d_{k,m,j}$, $\gamma = \gamma_{k,m,j}$, and $\alpha = \alpha_{k,j}$, as defined in Equation (4.2).

For an MEG system, the (m, j)th lead-field matrix element for the mth sensor, with position $\mathbf{r}_m$ and orientation $\mathbf{q}_m$, is given by [8]

$$a_{k,m,j} = \left[\frac{\mu_0}{4\pi g^2(\mathbf{r}_{k,j}, \mathbf{r}_m)}\, \mathbf{r}_{k,j} \times \Big(g(\mathbf{r}_{k,j}, \mathbf{r}_m)\, \mathbf{q}_m - \mathbf{f}^{\mathrm{T}}(\mathbf{r}_{k,j}, \mathbf{r}_m)\mathbf{q}_m\, \mathbf{r}_m\Big)\right]^{\mathrm{T}} \mathbf{q}_{k,j}, \tag{4.3}$$

where $(\mathbf{a} \times \mathbf{b})$ denotes the cross product between vectors $\mathbf{a}$ and $\mathbf{b}$. The scalar $g(\mathbf{r}_{k,j}, \mathbf{r}_m)$ and the vector $\mathbf{f}(\mathbf{r}_{k,j}, \mathbf{r}_m)$ are obtained as

$$g(\mathbf{r}_{k,j}, \mathbf{r}_m) = d_{k,m,j}\left(d_{k,m,j}\,|\mathbf{r}_m| + |\mathbf{r}_m|^2 - \mathbf{r}_{k,j}^{\mathrm{T}}\mathbf{r}_m\right)$$

$$\mathbf{f}(\mathbf{r}_{k,j}, \mathbf{r}_m) = \left(\frac{d_{k,m,j}^2}{|\mathbf{r}_m|} + \eta_{k,m,j} + 2\,d_{k,m,j} + 2\,|\mathbf{r}_m|\right)\mathbf{r}_m - \left(d_{k,m,j} + 2|\mathbf{r}_m| + \eta_{k,m,j}\right)\mathbf{r}_{k,j}$$

where $\eta_{k,m,j} = (\mathbf{r}_{k,j} - \mathbf{r}_m)^{\mathrm{T}}\mathbf{r}_m / d_{k,m,j}$, μ_0 is the permitivity of free space, and $d_{k,m,j}$ is defined in Equation (4.2).

The MEG/EEG inverse problem is to dynamically estimate the dynamic parameters, $s_{k,j}$, $\mathbf{r}_{k,j}$ and $\mathbf{q}_{k,j}$, of the jth dipole source at time step k from the MEG/EEG signal $\mathbf{z}_k$ defined in Equation (4.1).

4.2 State-space model of the dipole source tracking problem

First, we build the state-space model (Equation (2.1) and (2.2)) for single dipole source tracking problem. The dynamic evolution model of the jth EEG/MEG dipole sources is given by

$$\mathbf{x}_{k,j} = f(\mathbf{x}_{k-1,j}) + \nu_{k-1} = \mathbf{x}_{k-1,j} + \nu_{k-1}\,, \tag{4.4}$$

where $\mathbf{x}_{k,j} = [\mathbf{r}_{k,j}^{\mathrm{T}}\ \mathbf{q}_{k,j}^{\mathrm{T}}\ s_{k,j}]^{\mathrm{T}}$ is the 7-D state vector to be estimated and ν_k is the modeling error. Based on Equation (4.1), the measurement model of the jth dipole source can be represented as

$$\mathbf{z}_{k,j} = h(\mathbf{x}_{k,j}) + \mathbf{n}_k = \mathbf{A}_{k,j}(\mathbf{r}_{k,j}, \mathbf{q}_{k,j})\ s_{k,j} + \mathbf{n}_k\,, \tag{4.5}$$

where $\mathbf{A}_{k,j}$ is lead field of the jth dipole source which is a function of $\mathbf{r}_{k,j}, \mathbf{q}_{k,j}$. $\mathbf{n}_k$ is the measurement noise that is independent of the modeling noise ν_k in Equation (4.4).

Using the state-space model in Equations (4.4) and (4.5), the PF algorithm described in Chapter 2 provides an iterative approach to sequentially compute the posterior probability density function of the state $\mathbf{x}_{k,j}$ at every time step k, conditioned on the MEG/EEG measurements $\mathbf{z}_{1:k,j}$, $p(\mathbf{x}_{k,j}|\mathbf{z}_{1:k,j})$. The parameters of dipole source can be estimated by $\hat{\mathbf{x}}_{k,j} \approx \sum_{\ell=1}^{N} w_k^{(\ell)}\, \mathbf{x}_{k,j}^{(\ell)}$. Since there are only 7 unknown parameters for the jth dipole, the PF can achieve good tracking performance with a fairly small number of particles. However, when tracking multiple dipole sources, the dimension of state vector increases. Consequently, the number of particles required for accurate tracking also increases, resulting in a significant increase in the PF computational complexity. Next, we propose tracking multiple dipole sources using multiple particle filtering (MPF) to reduce the number of particles required.

4.3 Tracking multiple neural dipole sources using MPF

In the state-space model (Equation (2.1)), if the dimension $\mathfrak{D}_x$ of the state vector $\mathbf{x}_k$ is large, a large number of particles would be required to accurately estimate the state.In this case, multiple particle filtering (MPF) [49, 50] can be used to reduce the number of particles required.

4.3.1 Multiple particle filtering

The main idea of MPF is for $\mathfrak{D}_x > 1$, the state vector can be represented by $\mathbf{x}_k^T = [\mathbf{x}_{k,1}^T\, \mathbf{x}_{k,2}^T \ldots \mathbf{x}_{k,J}^T]$, where each subvector $\mathbf{x}_{k,j}$ is of dimension $L = \mathfrak{D}_x/J$ (with L assumed to be an integer). Therefor

the state update equation can be rewritten as

$$\begin{cases} \mathbf{x}_{k,1} = f_1(\mathbf{x}_{k-1,1}) + \boldsymbol{\nu}_{k-1,1} \\ \mathbf{x}_{k,2} = f_2(\mathbf{x}_{k-1,2}) + \boldsymbol{\nu}_{k-1,2} \\ \quad \vdots \\ \mathbf{x}_{k,j} = f_j(\mathbf{x}_{k-1,j}) + \boldsymbol{\nu}_{k-1,j} \\ \quad \vdots \\ \mathbf{x}_{k,J} = f_J(\mathbf{x}_{k-1,J}) + \boldsymbol{\nu}_{k-1,J} \end{cases}$$

where $\boldsymbol{\nu}_{k,j}$ is the jth modeling error process of dimension L.

A different PF needs to be used for each sub-vector $\mathbf{x}_{k,j}$, using N particles $\mathbf{x}_{k,j}^{(\ell)}$ with corresponding weights $w_{k,j}^{(\ell)}$, $\ell = 1,\ldots,N$ and $j = 1,\ldots,J$. When implementing the MPF, the particle propagation and resampling steps are the same as with the SIR PF. However, the weight updating step is different and given by [49]

$$w_{k,j}^{(\ell)} \propto w_{k-1,j}^{(\ell)} \frac{p(\mathbf{z}_k|\mathbf{x}_{k,j}^{(\ell)}, \tilde{\mathbf{x}}_{k,-j})\; p(\mathbf{x}_{k,j}^{(\ell)}|\mathbf{x}_{k-1,j}^{(\ell)}, \hat{\mathbf{x}}_{k-1,-j})}{q_j(\mathbf{x}_{k,j}^{(\ell)}|\mathbf{x}_{k-1,j}^{(\ell)}, \hat{\mathbf{x}}_{k-1,-j}, \mathbf{z}_k)}$$

where $\tilde{\mathbf{x}}_{k,-j}$ and $\hat{\mathbf{x}}_{k-1,-j}$ are predicted and estimated values of all the states at time step k except of the jth state vector $\mathbf{x}_{k,j}$, respectively. If the same importance density is used by all sub-vectors, that is, $q_j(\mathbf{x}_{k,j}|\mathbf{x}_{k-1,j}, \mathbf{z}_k) = p(\mathbf{x}_{k,j}|\mathbf{x}_{k-1,j})$, then the weight update function is reduced to

$$w_{k,j}^{(\ell)} \propto w_{k-1,j}^{(\ell)} p(\mathbf{z}_k|\mathbf{x}_{k,j}^{(\ell)}, \tilde{\mathbf{x}}_{k,-j})\,, \tag{4.6}$$

where the predicted values are obtained from

$$\tilde{\mathbf{x}}_{k,j} = \sum_{\ell=1}^{N_j} w_{k,j}^{(\ell)} \mathbf{x}_{k,j}^{(\ell)}\,. \tag{4.7}$$

The sub-vector state-space formulation suggests that, at each time step k, the jth PF obtains a prediction of its state and provides the information to the remaining $J-1$ PFs. The PFs use the exchanged information for computing the weights of their particles and eventually for generating new particles.

4.3.2 Tracking a known number of dipole sources using MPF

To efficiently track multiple dipole sources, we divide the high-dimensional multiple-dipole state space model into multiple, low-dimensional single-dipole state space models that can be solved using MPF with smaller number of particles. We assume that the number of dipoles is constant and known to be N_d. Then, for the MPF formulation, we divide the $7N_d \times 1$ vector $\mathbf{x}_k$ into N_d single-dipole 7×1 vectors $\mathbf{x}_{k,j}$, $j = 1, \ldots, N_d$. As before, $\mathbf{x}_{k,j}$ includes the location, orientation and amplitude of the jth dipole source. The state-space model can be written as

$$\begin{bmatrix} \mathbf{x}_{k,1} \\ \mathbf{x}_{k,2} \\ \vdots \\ \mathbf{x}_{k,N_d} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{k-1,1} + \boldsymbol{\nu}_{k-1} \\ \mathbf{x}_{k-1,2} + \boldsymbol{\nu}_{k-1} \\ \vdots \\ \mathbf{x}_{k-1,N_d} + \boldsymbol{\nu}_{k-1} \end{bmatrix} \tag{4.8}$$

$$\mathbf{z}_k = \sum_{j=1}^{N_d} \mathbf{z}_{k,j} = \mathbf{A}_k \mathbf{s}_k + \mathbf{n}_k \tag{4.9}$$

Then, we use N_d sub-PFs to track the N_d different dipole sources. The particle generation and resampling step of each sub-PF are implemented using the sampling importance resampling (SIR) PF [30]. However, the weight update step for the jth sub-PF is modified to [49]

$$w_{k,j}^{(\ell)} \propto w_{k-1,j}^{(\ell)} \; p\left(\mathbf{z}_k | \mathbf{x}_{k,j}^{(\ell)}, \tilde{\mathbf{x}}_{k,-j}\right), \quad \ell = 1, \ldots, N$$

where $\mathbf{x}_{k,j}^{(\ell)}$ and $w_{k,j}^{(\ell)}$ are the ℓth particle and ℓth weight of the jth sub-PF.

$$\tilde{\mathbf{x}}_{k,-j} = [\tilde{\mathbf{x}}_{k,1}^{\mathsf{T}} \; \ldots \; \tilde{\mathbf{x}}_{k,j-1}^{\mathsf{T}} \; \tilde{\mathbf{x}}_{k,j+1}^{\mathsf{T}} \; \ldots \; \tilde{\mathbf{x}}_{k,N_d}^{\mathsf{T}}]^{\mathsf{T}}$$

are the predicted dipole parameters, excluding $\tilde{\mathbf{x}}_{k,j}$, and

$$\hat{\mathbf{x}}_{k,-j} = [\hat{\mathbf{x}}_{k,1}^{\mathsf{T}} \; \ldots \; \hat{\mathbf{x}}_{k,j-1}^{\mathsf{T}} \; \hat{\mathbf{x}}_{k,j+1}^{\mathsf{T}} \; \ldots \; \hat{\mathbf{x}}_{k,N_d}^{\mathsf{T}}]^{\mathsf{T}}$$

are the estimated dipole parameters, excluding $\hat{\mathbf{x}}_{k,j}$. Thus, the predicted values of the jth dipole parameters are obtained as $\tilde{\mathbf{x}}_{k,j} = \sum_{\ell=1}^{N} w_{k-1,j}^{(\ell)} \mathbf{x}_{k,j}^{(\ell)}$. Note that the overall number of required particles

can be significantly reduced since we use the predicted value $\tilde{\mathbf{x}}_{k,-j}$ instead of a set of random particles $\mathbf{x}_{k,-j}^{(\ell)}$ to calculate the likelihood function

$$p(\mathbf{z}_k|\mathbf{x}_k^{(\ell)}) = p(\mathbf{z}_k|\mathbf{x}_{k,j}^{(\ell)}, \tilde{\mathbf{x}}_{k,-j}).$$

By using the multiple PF, the $7N_d$ dimensional estimation problem is reduced to multiple 7 dimensional estimation problems. Compared to traditional PF, this method can achieve better tracking performance in terms of mean square error (MSE) using the same number of particles or achieve same MSE with much less particles as shown in Section 4.3.3.

When applying the MPF to track neural activity, we make the critical assumption that the number of dipole sources is fixed and known at any given time step. However, in real scenarios, the number of dipoles is unknown and changing over time. As a result, the number of dipoles has to be estimated together with their location, orientation, and amplitude at each time step.

4.3.3 Neural activity tracking results of MPF

Synthetic data results: The RMSE performance results are first demonstrated for both the single-dipole and the multiple-dipole neural activity tracking problem using synthetic data. The data was created by inserting current dipoles into the sphere head model and calculating the resulting magnetic field using Equation (4.3) with Gaussian noise. Figures 4.2a and 4.2b compares the true location and moment of the x Cartesian coordinate of the dipole with the estimated ones obtained using both the SIR PF (SPF) tracker and the MPF tracker. Note that the SPF used 5,000 particles, and each of the two sub-PFs of the MPF used 1,000 particles each. For the N_d=2 multi-dipole case, the tracking results are shown in Figures 4.2c-4.2f. The MPF used two SPFs with 5,000 particles each, and the PF used 20,000 particles.

The estimation performance in terms of RMSE for both dipole cases is summarized in Tables 4.1 and 4.2. 100 Monte Carlo runs were performed to get the RMSE. Here, we also include RMSE of the Beamforming algorithm [73, 74] for comparison. We can see that although PF and MPF both provide reasonable estimates of the dipole sources, the MPF tracker needs significantly fewer number of particles. When the MPF tracker uses the same number of particles as the PF, it results in improved RMSE performance. For example, the RMSE performance improves from 0.398 mm to 0.253 mm for the single-dipole model and from 0.576 mm to 0.377 mm for the two-

dipole model. Furthermore, the RMSE of MPF is also smaller than the Beamforming algorithm when using 5,000 particles for single dipole and 20,000 for two dipoles as shown in Table 4.1 and 4.2.

Table 4.1: Comparison of RMSE for single-dipole model

Approach	Beamforming	PF	Multiple PF	
Number of particles	NaN	5,000	2,000	5,000
3-D Location RMSE	3.12 mm	3.98 mm	3.63 mm	2.53 mm
3-D Moment RMSE	2.41 nA	3.32 nA	2.95 nA	2.06 nA

Table 4.2: Comparison of RMSE for two-dipole model

Approach	Beamforming	PF	Multiple PF	
Number of particles	NaN	20,000	10,000	20,000
Dipole1 Location RMSE	4.16 mm	5.76 mm	4.91 mm	3.77 mm
Dipole2 Location RMSE	4.22 mm	5.47 mm	4.86 mm	3.59 mm

Real data results: The performance the new MPF approach is also compared with the SPF using real data from a language study experiment that was done in [74]. In this experiment, 87 incongruent sentences were given to the subjects sequentially. After a 300 ms warning tone, followed by a 1,200 ms pause, a sentence was presented. The time lapse between two sentences was 4,100 ms. MEG signals were recorded with a 151 sensor CTF Omega System, lowpass filtered at 100 Hz, and digitized at 300 Hz. The MEG data was averaged from 87 trials to increase the signal-to-noise ratio (SNR). The tracking trajectory results using single PF and MPF are shown in Figure 4.3. Figure 4.4 compares the tracking performance of the PF and the MPF with the Beamforming algorithm [73, 74]. Here, we used 10,000 particles for the PF and 5,000 particles for each of the sub-PFs of the MPF. Note that for real data, we do not have the true location of the dipole sources, however, we can see that the tracking results of MPF and Beamforming are close to each other.

4.4 Hardware implementation of proposed neural dipole tracking system

The overall block diagram of the proposed hardware architecture is shown in Fig. 4.6(a). It consists of J=N_d processing elements (PE) connected by a global bus, where N_d is the number of current dipoles and also the number of sub-PFs. Each PE implements the computation steps for one sub-PF concurrently as shown in Fig. 4.6(a). The inter-PE communication is quite small: PEj transmits

the predicted value $\tilde{\mathbf{x}}_{j,k}$ to other PEs and receives $\tilde{\mathbf{x}}_{-j,k}$ from other PEs to update the weights, where $\tilde{\mathbf{x}}_{j,k}$ can be calculated using (4.7).

Here, we employ the parallel PF independent Metropolis-Hastings (PPF-IMH) architecture [75] which can significantly reduce processing time and communication overhead. The block diagram of each PE is given in Figure 4.6(b). Each PE consists of four computing engines (CE) which operate in parallel and one central unit (CU). The M particles are equally distributed among four CEs, and the local PF processing steps, such as particle generation, weight evaluation and IMH resampling, are executed in each CE.

In order to reduce the communication overhead, the particles in each CE are distributed into groups based on the global range XL and XH computed in the CU. The average value $\mu_{\tilde{x}}$ and average weight μ_w of the particles in the group are used to form the new particles. This method may affect the estimation accuracy since the average value in each group is replicated. To improve estimation performance while keeping the communication overhead low, we use IMH resampling in each CE and transmit the range of the resampled particles to the CU. The resampled particles represent the posterior probability density function more accurately, thereby improving the estimation performance. In addition, since the IMH sampler can be easily pipelined, the processing period is not increased.

The MPF PPF-IMH hardware architecture is implemented using Verilog HDL and synthesized on the Xilinx Virtex-5 device (XC5VSX240T). The design was verified using Modelsim.

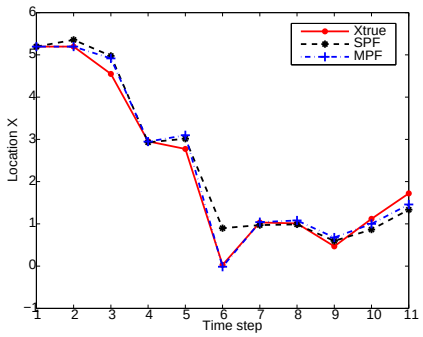
Resource utilization: Table 4.3 summarizes the architecture resource utilization for a two-dipole, 8,000 particle system. Each dipole is processed by a sub-PF, each sub-PF consists of 4 CEs, and each CE processes 1,000 particles. The exponential functions are implemented using CORDIC units, and the rest of the units are implemented using DSP cores.

Table 4.3: Resource utilization on Xilinx XC5VSX240T for MPF system

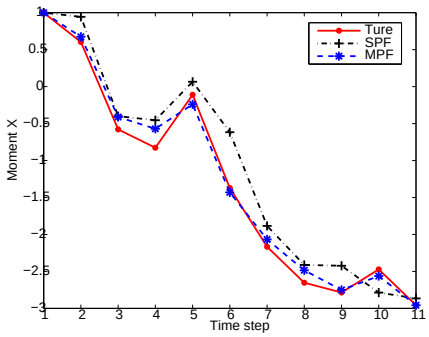
Unit	Occupied slices	Slice Reg.	Slice LUTs	Block Ram	DSP48Es
4-CE	25,820 (68%)	81,837 (54%)	84,164 (56%)	247 (47%)	606 (57%)

Execution Time: Figure 4.5 shows the timing for one iteration of the proposed method. For our

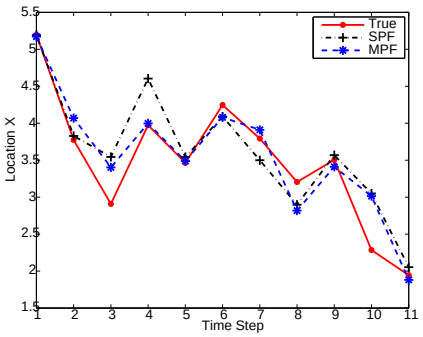
implementation, $M/4$=1000, L_s=2 is the sampling step delay, L_w=78 is the weighting latency determined by the calculation period of the exponential functions, L_r=12 is the latency of calculating the global ranges, and L_ρ=23 is the computing time for the replication factor. Thus, one iteration takes $T_{\text{total}}=(L_s+M/4+L_w+M/4+L_r+M/4+L_\rho)\times T_{\text{clk}}$=3,115 cycles. We choose the system clock rate as 100 MHz. The total processing period for one iteration is only T_{total}=31.15 μs.



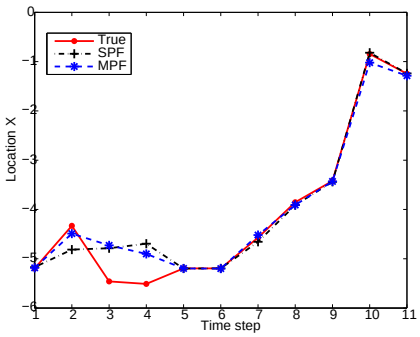
(a) Single dipole, location



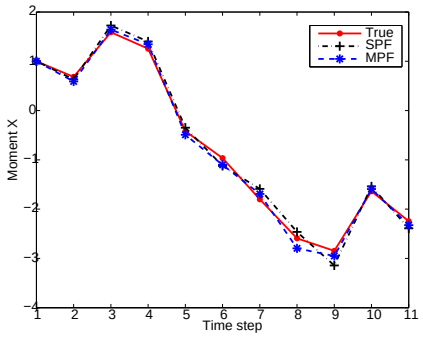
(b) Single dipole, moment



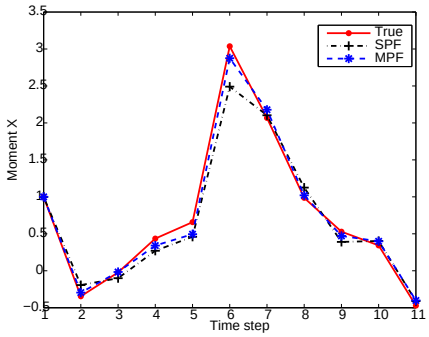
(c) Dipole 1, location



(d) Dipole 2, location



(e) Dipole 1, moment



(f) Dipole 2, moment

Figure 4.2: Comparison between the true (red) and estimated, using SIR PF or SPF (black) and MPF (blue), x Cartesian coordinate of (a) single dipole location; (b) single dipole moment; (c) Dipole 1 location; (d) Dipole 2 location; (e) Dipole 1 moment; and (f) Dipole 2 moment.

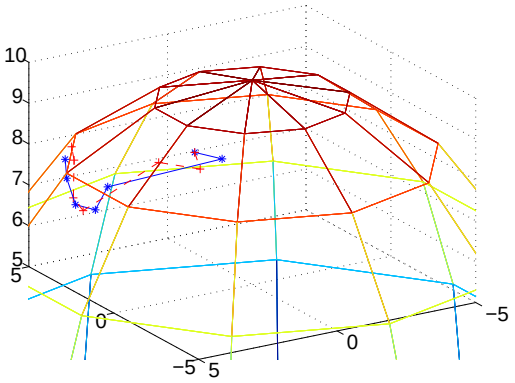
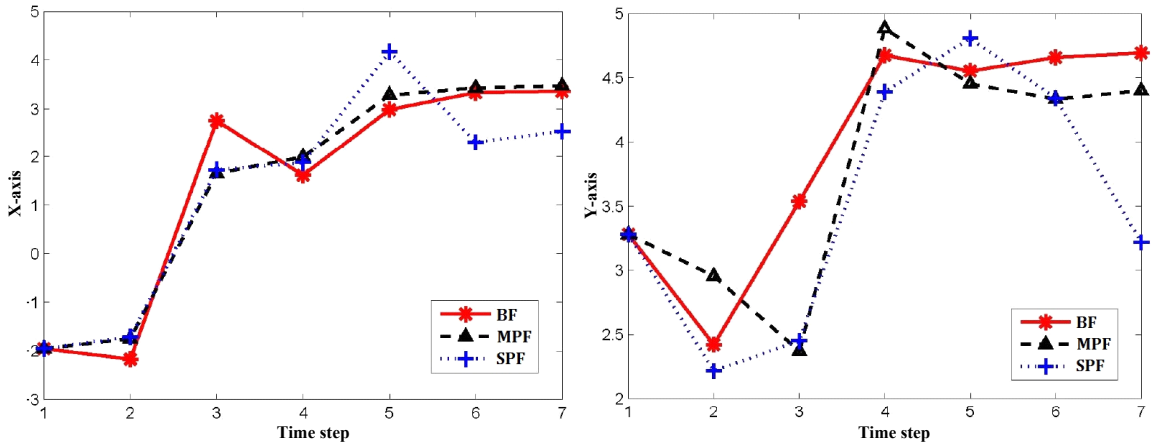
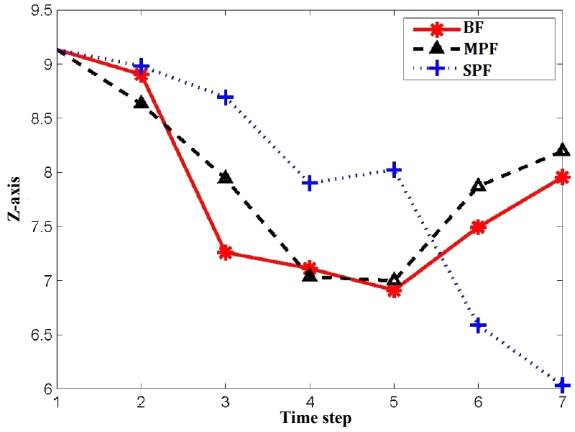


Figure 4.3: 3-D location coordinate tracking result using real data. Red dashed line is the tracking result for single PF and blue solid line is for MPF.



(a) x-coordinate (b) y-coordinate



(c) z-coordinate

Figure 4.4: Tracking performance for estimating the dipole location using real data using beamformer method (BF), SIR PF (SPF), and MPF.

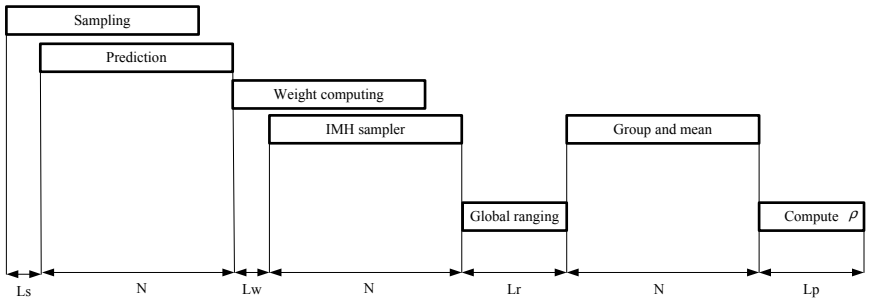
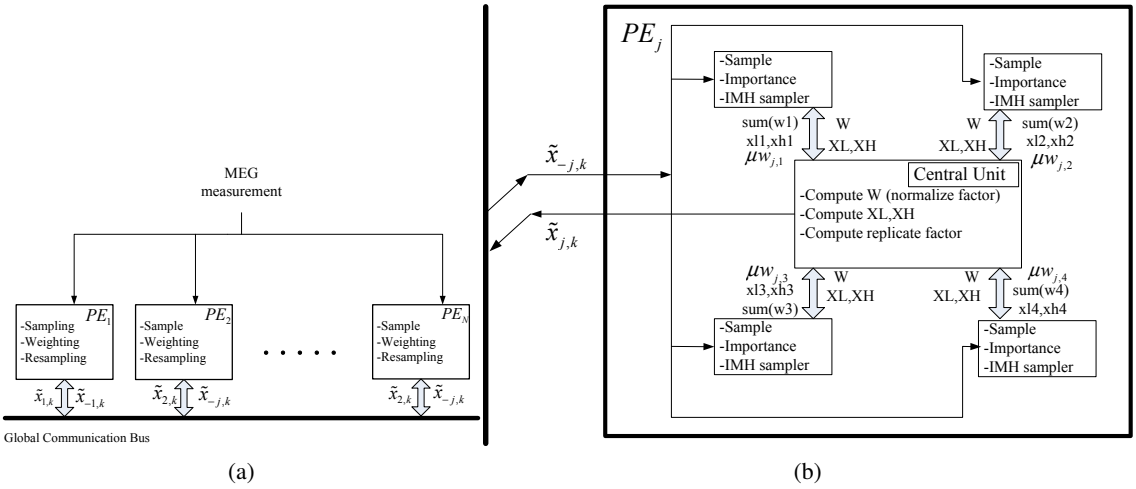


Figure 4.5: Execution time of proposed method.



(a) (b)

Figure 4.6: Block diagram: (a) overall architecture; (b) PPF-IMH architecture.

Chapter 5

Tracking Unknown Number of Dipole Sources Using PHD Filtering

The approaches mentioned in Chapter 4 assume that neural activity can be represented by a fixed and known number of current dipole sources. However, this is not a realistic assumption: neural activity varies with time, so current dipole source models and their parameters should also vary with time. In this chapter, we propose a new algorithm based on the probability hypothesis density filter (PHDF) for estimating both the unknown number of neural dipole sources and their parameters for real EEG/MEG data, with much fewer particles.

5.1 Problem formulation

In order to realistically model neural activity, the number of dipole sources N_k is unknown and changes with time. Under this scenario, the multiple dipole source state and measurements are modeled as random finite sets (RFS) [76, 51]. Thus, the problem under consideration is to estimate N_k as well as the parameters of the N_k dipole sources at each time step k.

Using the RFS formulation, the multiple dipole source state RFS is given by

$$\mathbf{X}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,N_k}\},$$

where $\mathbf{x}_{k,j} = [\mathbf{r}_{k,j}^{\mathrm{T}} \; \mathbf{q}_{k,j}^{\mathrm{T}} \; s_{k,j}]^{\mathrm{T}}$ is the state vector of the jth dipole source in Equation (4.4). Specifically, the multiple dipole source state RFS $\mathbf{X}_k$ is a finite-set-valued random vector whose elements, $\mathbf{x}_{k,j}$, $j = 1, \ldots, N_k$, are jointly characterized by a *discrete* probability distribution. Similarly, the multiple sensor measurement RFS is given by $\mathbf{Z}_k = \{z_{k,1}, \ldots, z_{k,M_k}\}$ where $z_{k,m}$ is the mth measurement in Equation (4.1) and M_k is the number of measurements at time step k; the measurements could also include false alarm measurements due to the presence of clutter. Since both the number of sources N_k and the number of measurements M_k can vary randomly in time, it is difficult to estimate both N_k and $\mathbf{X}_k$ at each time step k, given all measurements, up to time k, $\mathbf{Z}_{1:k} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_k\}$. It is also not known *a priori* which source has generated a given measurement. As dipole sources generate neural activity independently of one another, the PHDF has been shown to provide an efficient approach towards solving this problem [52, 51].

5.2 Probability hypothesis density Filtering

Assuming a state RFS $\mathbf{X}_k$, with $\mathbf{x}_k \in \mathbf{X}_k$, and a measurement RFS $\mathbf{Z}_k$, integrating the probability hypothesis density or complexity function $\zeta(\mathbf{x}_k|\mathbf{Z}_k)$ on a given region $\mathscr{R}$, provides the expected number of sources present in region $\mathscr{R}$. Also, the locations of the peaks of the density provide estimates of the parameters of the sources in the region $\mathscr{R}$ [51]. To formulate the PHDF for the dipole source estimation problem, we use the method in [51] to first describe an RFS model for time evolution of the multiple dipole source state and an RFS model for the sensor measurements. Specifically, given the dipole source state $\mathbf{X}_{k-1}$ at time step $k-1$, the dipole source state $\mathbf{X}_k$ is formed by combining: (a) dipole sources still present from the previous time step, $\mathbf{X}_{k|k-1}^{\text{prev}}$; (b) dipole sources that are new at the present time step, $\mathbf{X}_k^{\text{new}}$; and (c) dipole sources spawning from sources from the previous time step, $\mathbf{X}_{k|k-1}^{\text{spn}}$. For the RFS measurement model, we consider the likelihood $p(\mathbf{Z}_k|\mathbf{x}_k)$ for each possible dipole source state $\mathbf{x}_k \in \mathbf{X}_k$.

The PHDF assumes that the predicted multiple dipole source posterior density $p(\mathbf{x}_k|\mathbf{Z}_{1:k-1})$ can be completely characterized by its first moment that can be represented by the multiple dipole source RFS probability hypothesis density or intensity function $\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1})$ [51]. Thus, given the posterior intensity $\zeta(\mathbf{x}_{k-1}|\mathbf{Z}_{1:k-1})$ at time step $(k-1)$, the predicted intensity $\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1})$ can be obtained as [51]:

$$\begin{aligned}\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1}) &= \zeta(\mathbf{x}_k^{\text{new}}|\mathbf{Z}_{1:k}) + \int \Big[\Pr_{k|k-1}(\tilde{\mathbf{x}}_{k-1})\, p(\mathbf{x}_k|\tilde{\mathbf{x}}_{k-1}) + \\ &\qquad \zeta(\mathbf{x}_k^{\text{spn}}|\mathbf{Z}_{1:k-1})\Big]\, \zeta(\tilde{\mathbf{x}}_{k-1}|\mathbf{Z}_{1:k-1})\, d\tilde{\mathbf{x}}_{k-1}\end{aligned} \tag{5.1}$$

where $\mathbf{x}_k^{\text{new}} \in \mathbf{X}_k^{\text{new}}$, $\mathbf{x}_k^{\text{spn}} \in \mathbf{X}_{k|k-1}^{\text{spn}}$, and $\Pr_{k|k-1}(\mathbf{x}_{k-1})$ is the probability that a dipole source that was present at time step $(k-1)$ will still be present at time step k. The posterior intensity is given by

$$\begin{aligned}\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k}) &= (1 - \Pr_k^{\text{det}}(\mathbf{x}_k))\, \zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1}) + \\ &\qquad \sum_{\mathbf{Z}_k \in \mathbf{Z}_{1:k}} \frac{\Pr_k^{\text{det}}(\mathbf{x}_k)\; p(\mathbf{Z}_k|\mathbf{x}_k)\; \zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1})}{\zeta(\mathbf{Z}_k^{\text{clt}}) + \int \Pr_k^{\text{det}}(\tilde{\mathbf{x}}_k)\, p(\mathbf{Z}_k|\tilde{\mathbf{x}}_k)\, \zeta(\tilde{\mathbf{x}}_k|\mathbf{Z}_{1:k-1})\, d\tilde{\mathbf{x}}_k}\end{aligned} \tag{5.2}$$

where $\Pr_k^{\text{det}}(\mathbf{x}_k)$ is the probability of detecting a dipole source at time step k. As the received multiple measurement RFS $\mathbf{Z}_k$ can also include clutter, $\mathbf{Z}_k^{\text{clt}}$, due to possible false alarms, $\zeta(\mathbf{Z}_k^{\text{clt}})$ is used to denote the clutter intensity; it is assumed that the clutter RFS is independent of the dipole source

measurement RFS and that the dipole source measurement RFS elements are mutually independent.

5.3 PHDF implementation using particle filtering

The PHDF prediction and update equations in Equation (5.1) and (5.2) involve multiple integrals that do not have computationally tractable closed form expressions, even for the simple linear Gaussian dynamic case. One possible implementation of the PHDF is using particle filtering [54]. Specifically, assuming that the intensity function $\zeta(\mathbf{x}_{k-1}|\mathbf{Z}_{1:k-1})$ at time step $(k-1)$ can be approximated by a set of N_p particles $\mathbf{x}_{k-1}^{(\ell)}$ and their corresponding weights $w_{k-1}^{(\ell)}$, $\ell = 1,\ldots,N_p$,

$$\zeta(\mathbf{x}_{k-1}|\mathbf{Z}_{1:k-1}) = \sum_{\ell=1}^{N_p} w_{k-1}^{(\ell)} \delta(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^{(\ell)}). \tag{5.3}$$

where $\delta(\cdot)$ is the Dirac delta function, then substituting (5.3) into (5.1), we obtain

$$\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1}) = \zeta(\mathbf{x}_k^{\text{new}}|\mathbf{Z}_{1:k}) + \sum_{\ell=1}^{N_p} w_{k-1}^{(\ell)} \left[\Pr_{k|k-1}(\mathbf{x}_{k-1}^{(\ell)})\, p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)}) + \zeta(\mathbf{x}_k^{\text{spn}}|\mathbf{Z}_{1:k-1}) \right]. \tag{5.4}$$

A particle approximation of $\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1})$ can be obtained by applying importance sampling to each term in (5.4). Specifically, the samples $\mathbf{x}_k^{(\ell)}$, $\ell = 1,\ldots,N_p$, are drawn from the importance density function $q_k(\mathbf{x}_k^{(\ell)}|\mathbf{x}_{k-1}^{(\ell)}, \mathbf{Z}_k) = p(\mathbf{x}_k^{(\ell)}|\mathbf{x}_{k-1}^{(\ell)})$, and the samples $\mathbf{x}_k^{(\ell)}$, $\ell = N_p+1,\ldots,(N_p+N_q)$ are drawn from the importance intensity function $\xi_k(\mathbf{x}_k^{(\ell)}|\mathbf{Z}_k) = \zeta(\mathbf{x}_k^{\text{new}}|\mathbf{Z}_{1:k})$. Here N_q is the additional number of particles that are needed to represent the new dipole sources. Then, the prior intensity function $\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1})$ can be approximated by particles $\mathbf{x}_k^{(\ell)}$ and their weights $w_{k|k-1}^{(\ell)}$, $\ell = 1,2,\ldots,(N_p+N_q)$ as

$$\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k-1}) = \sum_{\ell=1}^{N_p+N_q} w_{k|k-1}^{(\ell)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(\ell)}) \tag{5.5}$$

where

$$w_{k|k-1}^{(\ell)} = \begin{cases} w_{k-1}^{(\ell)} \left(\Pr_{k|k-1}(\mathbf{x}_{k-1}^{(\ell)}) + \dfrac{\zeta(\mathbf{x}_k^{\text{spn}}|\mathbf{Z}_{1:k-1})}{p(\mathbf{x}_k^{(\ell)}|\mathbf{x}_{k-1}^{(\ell)})} \right), & \ell = 1,\ldots,N_p \\ 1/N_q, & \ell = N_p+1,\ldots,(N_p+N_q) \end{cases} \tag{5.6}$$

Substituting Equation (5.5) into Equation (5.2), we obtain the particle approximation of the posterior intensity function as

$$\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k}) = \sum_{\ell=1}^{N_p+N_q} w_k^{(\ell)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(\ell)}), \tag{5.7}$$

where

$$w_k^{(\ell)} = w_{k|k-1}^{(\ell)} \left(1 - \Pr_k^{\text{det}}(\mathbf{x}_k^{(\ell)}) + \sum_{\mathbf{Z}_k \in \mathbf{Z}_{1:k}} \frac{\Pr_k^{\text{det}}(\mathbf{x}_k^{(\ell)})\, p(\mathbf{Z}_k|\mathbf{x}_k^{(\ell)})}{\zeta(\mathbf{Z}_k^{\text{clt}}) + C_k(\mathbf{Z}_k)} \right) \tag{5.8}$$

and

$$C_k(\mathbf{Z}_k) = \sum_{\ell=1}^{N_p+N_q} w_{k|k-1}^{(\ell)} \Pr_k^{\text{det}}(\mathbf{x}_k^{(\ell)})\, p(\mathbf{Z}_k|\mathbf{x}_k^{(\ell)}) .$$

Based on Equations (5.1) and (5.2), a particle approximation of the posterior intensity $\zeta(\mathbf{x}_k|\mathbf{Z}_{1:k})$ can be obtained at time step k from its particle approximation at the previous time step $k-1$. Using this particle approximation, both the number of targets in a given region and the targets' parameters can be estimated. The particle implementation of PHD filtering (PF-PHDF) is robust and computationally inexpensive compared to existing multiple target tracking techniques, and it has been successfully used in radar and sonar tracking [77, 78]. However, there are significant challenges in applying the PF-PHDF to solve the MEG/EEG inverse problem. Whereas each measurement in radar is generated from a single target, MEG/EEG sensor measurements are due to contributions from all dipole sources; thus, the observed data must be decomposed before it can be used by the PF-PHDF. We next introduce the independent component analysis (ICA) algorithm which can decompose the measurements into independent components such that each component corresponds to one dipole source.

5.4 Independent Component Analysis

Independent Component Analysis (ICA) [79, 80] is a signal processing technique used to express a set of random variables as a linear combination of statistically independent components.

Assume that the measurement vector $\mathbf{z} = [z_1, z_2, ..., z_m]^T$ is a linear mixture of the source signal $\mathbf{s} = [s_1, s_2, \ldots s_n]^T$. Then the ICA model is written as

$$\mathbf{z} = \mathbf{A}\mathbf{s} \tag{5.9}$$

where both $\mathbf{z}$ and $\mathbf{s}$ are random vectors and $\mathbf{A}$ is called the mixing matrix. Here we assume that $s_1, s_2, \ldots, s_n$ are independent components and cannot be directly observed. Without loss of generality, we can assume that both the mixture variables and the independent components have zero mean.

The objective of ICA is to estimate both $\mathbf{A}$ and $\mathbf{s}$ using the measurement $\mathbf{z}$. This must be done under two assumptions [81]

1. The components s_i are statistically independent;

2. The independent components must have nonGaussian distributions or just one of the independent components is Gaussian.

5.4.1 Principles of ICA

Let us now assume that the data vector $\mathbf{z}$ is a mixture of independent components according to the ICA data model in Equation (5.9). For simplicity, we also assume that the dimension of the measurement vector m is equal to the number of independent components n. Under this assumption the mixing matrix $\mathbf{A}$ is square (this assumption can be relaxed, as explained in Section 5.4.3). To estimate one of the independent components, we consider a linear combination of z_i:

$$y = \mathbf{w}^T \mathbf{z} = \sum_{i=1}^{n} w_i z_i, \tag{5.10}$$

where $\mathbf{w}$ is the vector to be determined. If $\mathbf{w}$ was one of the rows of $\mathbf{A}^{-1}$, y would equal one of the independent components. In practice, we cannot determine such a $\mathbf{w}$ exactly, because we have no knowledge of matrix $\mathbf{A}$, but we can find an estimator that gives a good approximation. This leads to the basic principle of ICA estimation.

Equation (5.10) can be rewritten as

$$y = \mathbf{w}^T \mathbf{z} = \mathbf{w}^T \mathbf{A} \mathbf{s} = \mathbf{u}^T \mathbf{s}$$

where $\mathbf{u} = \mathbf{A}^T \mathbf{w}$. Scalar y is thus a linear combination of s_i, with weights given by u_i. Based on the Central Limit Theorem, the distribution of a sum of independent random variables tends toward a Gaussian distribution. Thus, a sum of two independent random variables usually has a distribution that is closer to Gaussian than any of the two original random variables. As a result, y is more Gaussian than any of the s_i. y becomes least Gaussian when it equals one of the s_i, in which case, only one of the elements u_i of $\mathbf{u}$ is nonzero.

Therefore, we could take $\mathbf{w}$ as a vector that maximizes the nonGaussianity of $y = \mathbf{w}^T \mathbf{z}$. Such a vector would necessarily correspond a $\mathbf{u}$ which has only one nonzero component. This means that

$\mathbf{w}^T\mathbf{z} = \mathbf{u}^T\mathbf{s}$ equals one of the independent components. Maximizing the nonGaussianity of $\mathbf{w}^T\mathbf{z}$ thus gives us one of the independent components.

5.4.2 Solution of ICA

In order to maximize the nonGaussianity, first we must have a quantitative measure of nonGaussianity of a random variable. To simplify the problem, we assume that the random variable y is zero-mean and has variance equal to one. This can be achieved by preprocessing.

The classical measure of nonGaussianity is kurtosis or the fourth-order cumulant. The kurtosis of y is classically defined by

$$\begin{aligned} kurt(y) &= E\{y^4\} - 3(E\{y^2\})^2 \\ &= E\{y^4\} - 3 \end{aligned} \tag{5.11}$$

This shows that kurtosis is simply a normalized version of the fourth moment $E\{y^4\}$. For a Gaussian random variable, the kurtosis is zero. While for a nonGaussian random variable, the kurtosis is nonzero. Kurtosis can be both positive or negative. Random variables that have a negative kurtosis are called subGaussian, and those with positive kurtosis are called superGaussian. Typically nonGaussianity is measured by the absolute value of kurtosis.

The independent components can be found by kurtosis minimization or maximization. An algorithm based on gradient descent or ascent has been used in [82, 83]. However, for this algorithm, a bad choice of the learning rate can destroy the convergence. Therefore, a fixed-point iteration algorithm (FastICA) is proposed to make the learning radically faster and more reliable [53].

5.4.3 FastICA algorithm

FastICA is a fixed-point iteration used to find the maxima or minima of the kurtosis $kurt(y) = kurt(\mathbf{w}^T\mathbf{z})$. The steps of FastICA can be shown as follows [53]

1. Take a random initial vector $\mathbf{w}(0)$ of norm 1. Let $k = 1$.

2. Let $\mathbf{w}(k) = E\{\mathbf{z}(\mathbf{w}(k-1)^T\mathbf{z})^3\} - 3\mathbf{w}(k-1)$. The expectation can be estimated using a large sample of $\mathbf{z}$ vectors (say, 1,000 points).

3. Divide $\mathbf{w}(k)$ by its norm.

4. If $|\mathbf{w}(k)^T\mathbf{w}(k-1)|$is not close enough to 1, let $k = k+1$ and go back to step 2. Otherwise, output the vector $\mathbf{w}(k)$.

The final $\mathbf{w}(k)$ given by the algorithm is a vector that maximizes the kurtosis of $y = \mathbf{w}^T\mathbf{z}$. Such a vector equals one of the rows of the inverse of $\mathbf{A}$. This means that $\mathbf{w}(k)$ separates one of the non Gaussian source signals in the sense that $\mathbf{w}(k)^T\mathbf{x}$ equals one of the source signals. In addition, a very small number of iterations, usually $5-10$ seems to be enough to obtain the maximal accuracy allowed by the sample data [53].

To estimate n independent components, we run this algorithm n times. To ensure that each time we estimate a different independent component, we only need to add a orthogonalizing projection [53] inside the loop. After ICA, the mixed measurement $\mathbf{z}$ is separated into n independent components and these components can be used as the inputs of the PF-PHDF to track the unknown number of targets. Bsed on the ICA and PF-PHDF, next, we will proposed the neural activity tracking system which can estimate both the unknown number of neural dipole sources and their parameters for EEG/MEG data.

5.5 Proposed algorithm for tracking an unknown number of dipole sources

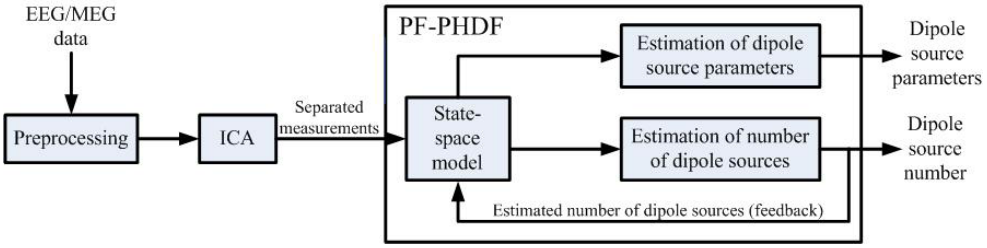


Figure 5.1: Block diagram of proposed unknown number of dipole sources tracking system.

The overall block diagram of the proposed system is shown in Figure 5.1. The steps of the proposed system are

1. Prewhiten the EEG/MEG measurement. This reduces the size of the measurement matrix and simplifies computation.

2. Decompose the pre-whitened measurement into independent components; each component corresponds to only one dipole source or clutter.

3. Apply the PF-PHDF to estimate the number of dipole sources and their parameters.

Next we will describe each of the stages in more detail.

5.5.1 Efficient pre-whitening of measurements

The PF-PHDF algorithm requires each measurement vector to be generated by a single object. However, for the EEG/MEG dipole model, the EEG/MEG measurement $\mathbf{z}_k = \mathbf{A}_k \mathbf{s}_k$ in Equation (4.1) is generated by multiple neural sources. Thus, we first need to decompose the EEG/MEG measurements into individual components, with each component corresponding to an individual neural source. The decomposition can be simplified by first performing a preliminary pre-whitening of the EEG/MEG data $\mathbf{z}_k = [z_{k,1}\ z_{k,2}\ \ldots\ z_{k,M}]^{\mathrm{T}}$. Specifically, the data is linearly transformed to $\mathbf{v}_k = \mathbf{U}\mathbf{z}_k = \mathbf{U}\mathbf{A}_k\mathbf{s}_k$ whose elements are mutually uncorrelated with unit variance, $E[\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}] = \mathbf{I}$ [53], where $\mathbf{I}$ is the identity matrix and $\mathbf{U}$ is a linear transformation matrix. An example of such a linear transformation can be obtained using eigenvalue decomposition $\mathbf{U} = \Lambda^{-1/2}\Psi^{\mathrm{T}}$ of the measurement covariance matrix $\Sigma_{\mathrm{cov}} = E[\mathbf{z}_k\mathbf{z}_k^{\mathrm{T}}]$, where Λ is a diagonal matrix whose elements are the eigenvalues of Σ_{cov} and Ψ is a matrix whose columns consist of the corresponding eigenvectors. In particular, assuming M EEG/MEG measurement sensors, the eigenvalues of the $M \times M$ covariance matrix Σ_{cov} need to be obtained. Although there are many algorithms to calculate eigenvectors from a covariance matrix, when the size of $M > 5$, most of them can hardly be mapped into an efficient VLSI architecture. In most EEG/MEG systems, the number of sensors is between $M = 30$ and $M = 150$, and so the challenge is to find an efficient approach to calculate the eigenvalues of a large matrix. In our previous work [52], we used channel decomposition to solve this problem. The channels were divided into several groups and ICA was applied to each group. However, that approach resulted in estimation performance loss, and, despite the fact that the computational intensity was reduced, this step was still the bottleneck of our implementation.

Here we consider a different approach, where we assume that only a small set of patches of the human brain are activated at a time [1, 17, 24, 84]. Under this assumption, the number of dipoles N_d is much smaller than the number of sensors M, $N_d << M$. Thus instead of calculating all the

Algorithm 4 Threshold-based leading-only eigenvalue-eigenvector distilling algorithm [85]

Choose eigenvalue threshold V_{thr} and number of iterations r; compute covariance matrix Σ_{cov}
Calculate the leading eigenvector matrix Ψ and eigenvalues λ
Set $i = 1$ and $\Psi_i^{(0)} = 0$
Initialize eigenvector $\Psi_i^{(0)}$ randomly with norm 1
for $j = 1 : r$ **do**

$$\Psi_i^{(j)} = \frac{1}{||\Phi_i^{(j)}||}\Phi_i^{(j)} \text{ where } \Phi_i^{(j)} = \Sigma_{\text{cov}}\Psi_i^{(j-1)} - \sum_{l=0}^{i-1}\left(\left[\Sigma_{\text{cov}}\Psi_i^{(j-1)}\right]^{\mathrm{T}}\Psi_l\right)\Sigma_{\text{cov}}\Psi_i^{(j-1)}$$

end for
Set $\Psi_i = \Psi_i^{(r)}$ and $\lambda_i = \frac{1}{\Psi_i^{\mathrm{T}}\Psi_i}\left(\Sigma_{\text{cov}}\Psi_i\right)^{\mathrm{T}}\Psi_i$

if $\lambda_i > V_{\text{thr}}$ **then**
Store λ_i and Ψ_i and set $i = i + 1$
else
Stop
end if

eigenvalues and eigenvectors, we only need to find the leading eigenvalues and eigenvectors corresponding to the N_d active dipoles. We propose an efficient leading eigenvector and eigenvalue calculation approach based on the eigenvector distilling algorithm [85]. The steps of the proposed method are described in Algorithm 4. In order to obtain the ith largest eigenvector Ψ_i, we first initialize $\Psi_i^{(0)}$ randomly with norm 1 and then update it using r iterations to obtain Ψ_i. Its corresponding eigenvalue λ_i is obtained and compared to the threshold V_{thr}. If $\lambda_i > V_{\text{thr}}$, this eigenvector is retained and the same procedure is used to calculate the next one. Note that the leading eigenvectors are calculated one by one, in reducing order of dominance. As the number of dipoles is unknown, the number of eigenvalues and eigenvectors to be distilled is unknown as well. So we use a threshold to pick the eigenvalues to be computed. We will analyze the influence of this threshold on algorithm performance in Section 5.6.2.

5.5.2 Component analysis of pre-whitened measurements

After pre-processing, the new data vector $\mathbf{v}_k$ has reduced dimensionality and reduced noise power. Since distinct neural sources are mutually independent [1], the ICA algorithm introduced in Section 5.4 can be used to decompose the pre-processed MEG/EEG data. Here, we assume that in a short time window the location and orientation of dipole sources are fixed. As a result, the lead-field matrix $\mathbf{A}$ doesn't change with time in this period. The window length is a critical parameter which

will be analyzed in Section 5.7.2. For the EEG/MEG data in one window, we use the FastICA, a free Matlab software package that implements a fast fixed-point ICA algorithm [53], to obtain a de-mixing matrix, $\mathbf{W} = \mathbf{UA}$. Using this matrix, we can obtain J estimated independent sources as $\hat{\mathbf{s}}_k = \mathbf{W}^{\mathrm{T}}\mathbf{v}_k = [\hat{s}_{k,1} \ \ldots \ \hat{s}_{k,J}]^{\mathrm{T}}$. The estimated mixing matrix $\hat{\mathbf{A}} = [\hat{\mathbf{a}}_1 \ \ldots \ \hat{\mathbf{a}}_J]$, where $\hat{\mathbf{a}}_j$ is the jth column of $\hat{\mathbf{A}}$, can be obtained as $\hat{\mathbf{A}} = \mathbf{U}^{-1}\mathbf{W}$. Using $\hat{\mathbf{a}}_j$, the MEG/EEG signal that contributed from the jth individual source is given by

$$\hat{\mathbf{z}}_{k,j} = \hat{\mathbf{a}}_j \hat{s}_{k,j}, \quad j = 1, \ldots, J. \tag{5.12}$$

After this stage, the decomposed unmixed MEG/EEG measurement vectors are given by

$$\hat{\mathbf{z}}_k = [\hat{z}_{k,1} \ \ldots \ \hat{z}_{k,J}].$$

Thus, each new measurement $\hat{z}_{k,j}$ is assumed to have originated either from a single dipole source or an artifact (non-brain) activity (false alarm).

5.5.3 Multi-dipole estimation using PF-PHDF

The decomposed pre-whitened MEG/EEG measurement RFS $\hat{\mathbf{Z}}_k = \{\hat{\mathbf{z}}_{k,1}, \ \hat{\mathbf{z}}_{k,2}, \ \ldots \ \hat{\mathbf{z}}_{k,J}\}$ can now be directly used as the input to the PF-PHDF algorithm to estimate the number of dipole sources and their unknown parameters. The corresponding state-space RFS model for the MEG/EEG source estimation problem is given by

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \nu_{k-1} \tag{5.13}$$

$$\hat{\mathbf{Z}}_k = h(\mathbf{X}_k) + \mathbf{n}_k. \tag{5.14}$$

The specific PF-PHDF algorithm steps are described next; Figure 5.2 demonstrates the PF-PHDF framework for the MEG/EEG dipole source estimation problem.

*S*tep 1-Initialization: At time step $k = 0$, the particles $\tilde{\mathbf{x}}_0^{(\ell)}$, $\ell = 1, \ldots, N_0$, are drawn from the initial intensity function $\zeta(\mathbf{x}_0)$, where $N_0 = N N_{d_0}$, N_{d_0} is the initial number of dipoles and N is the number of particles used for each dipole, and the corresponding weights are obtained as $w_0^{(\ell)} = N_{d_0}/N_0$, $\ell = 1, \ldots, N_0$.

*S*tep 2-Prediction: At time step k, particles $\tilde{\mathbf{x}}_k^{(\ell)}$, $\ell = 1, \ldots, N_{k-1}, \ldots, R_k$, where $R_k = N_{k-1} + N_q$, are sampled and, assuming no dipole source spawning, the corresponding weights are evaluated using

Equation (5.6)

$$\tilde{w}_{k|k-1}^{(\ell)} = \begin{cases} \tilde{w}_{k-1}^{(\ell)} \Pr_{k|k-1}(\tilde{\mathbf{x}}_{k-1}^{(\ell)}), & \ell = 1,\ldots,N_{k-1} \\ 1/N_q, & \ell = N_{k-1}+1,\ldots,R_k \end{cases} \tag{5.15}$$

Step 3-Updating: Using (5.8), the weights are updated to

$$\tilde{w}_k^{(\ell)} = \tilde{w}_{k|k-1}^{(\ell)} \left(1 - \Pr_k^{\text{det}}(\tilde{\mathbf{x}}_k^{(\ell)}) + \sum_{j=1}^{J} \frac{\Pr_k^{\text{det}}(\tilde{\mathbf{x}}_k^{(\ell)})\, p(\hat{\mathbf{z}}_{k,j}|\tilde{\mathbf{x}}_k^{(\ell)})}{\zeta(\mathbf{Z}_k^{\text{clt}}) + C_k(\hat{\mathbf{z}}_{k,j})}\right),$$

where

$$C_k(\hat{\mathbf{z}}_{k,j}) = \sum_{\ell=1}^{R_k} \tilde{w}_{k|k-1}^{(\ell)} \Pr_k^{\text{det}}(\tilde{\mathbf{x}}_k^{(\ell)})\, p(\hat{\mathbf{z}}_{k,j}|\tilde{\mathbf{x}}_k^{(\ell)}).$$

*S*tep 4-Resampling: The number of dipole sources is estimated as $\hat{N}_{d_k} = \sum_{\ell=1}^{R_k} \tilde{w}_k^{(\ell)}$; the particles $\tilde{x}_k^{(\ell)}$ are resampled and their corresponding weights $\tilde{w}_k^{(\ell)}$, $\ell = 1,\ldots,R_k$ are computed to obtain $\tilde{\mathbf{x}}_k^{(\ell)}$ and $\tilde{w}_k^{(\ell)} = \hat{N}_{d_k}/N_k$, $\ell = 1,\ldots,N_k$. Here, $N_k = N\,\text{round}(\hat{N}_{d_k})$, where round$(N)$ denotes the nearest integer to N.

*S*tep 5-Estimating dipole state: The resampled particles are clustered and the state parameters are estimated. The clustering is performed in 3-D using the k-means clustering algorithm.

Using the PF-PHDF, the number of particles changes over time and is proportional to the number of dipoles, i.e., at time k, $N_k \propto \hat{N}_{d_k}$. Unlike standard PF, there is a summation among sub-measurements $\hat{\mathbf{z}}_{k,j}$, $j = 1,\ldots,J$, corresponding to individual sources, when updating the weights in Step 3. After Step 3, the posterior intensity $\zeta(\mathbf{x}_k|\hat{\mathbf{Z}}_k)$ at time k is approximated using particles $\tilde{\mathbf{x}}_k^{(\ell)}$ and weights $\tilde{w}_k^{(\ell)}$, $\ell = 1,\ldots,R_k$, that contain all available dipole source information. For example, the number of dipoles can be obtained by integrating the posterior intensity, which is equal to the summation of the weights; and the dipole source parameters can be estimated from the peaks of the intensity. In the resampling step, the new weights are not normalized to 1, but sum to $\hat{N}_{d_k}$, the estimated number of dipole sources.

5.6 Algorithm performance results for tracking unknown number of dipoles

In order to demonstrate the tracking performance, we apply the proposed PF-PHDF algorithm to track the unknown number of neural sources using both synthetic and real EEG data.

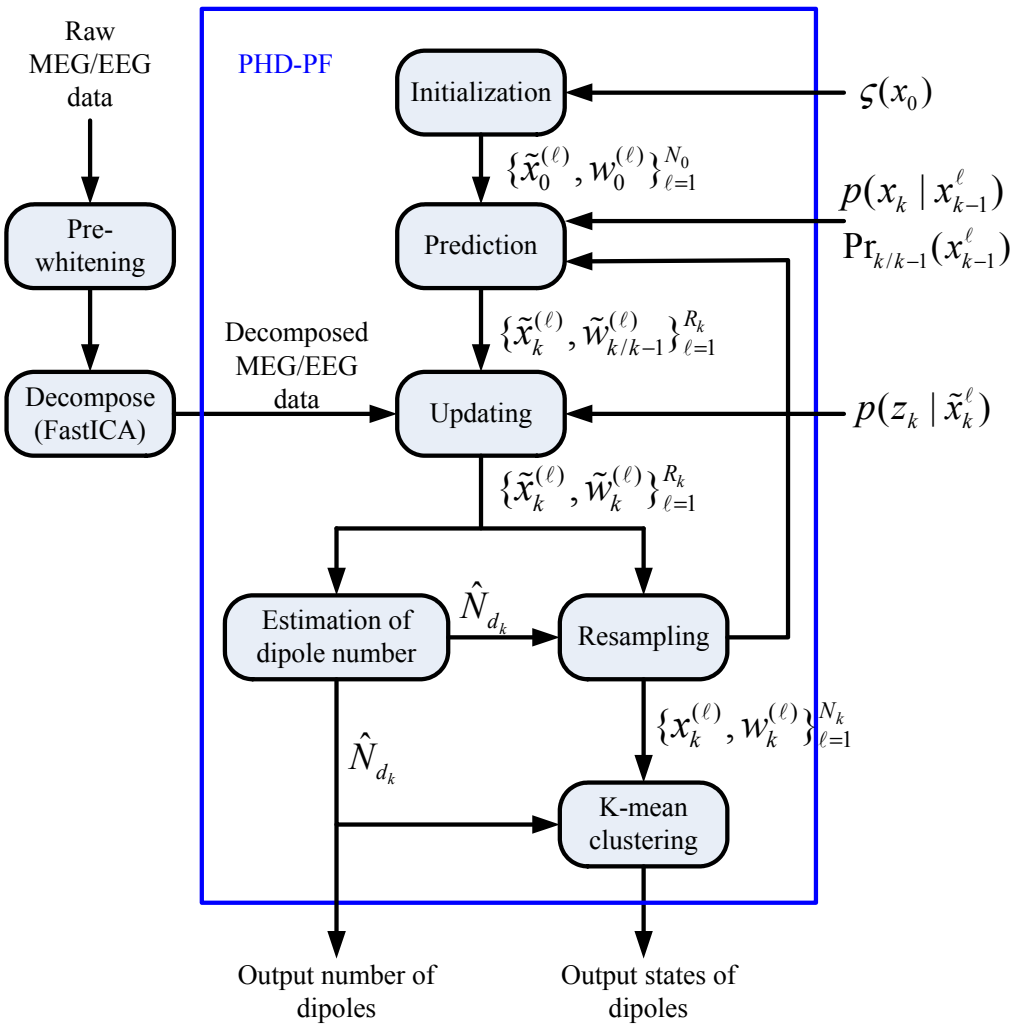


Figure 5.2: Framework of the proposed neural activity tracking system based on PF-PHDF.

5.6.1 Simulation set up

First we use synthetic data from a previous study in [25] with three dipoles localized at $V1$ (1.11, 5.34, 4.98), $V5_R$ (4.36, 3.68, 4.44) and $V5_L$ (3.37, 4.85, 4.81) in an example where the brain volume hemisphere has a radius of 10 cm. The measurement noise is Gaussian with 0 mean and variance $\sigma = 10^{-5}$. For this simulation, we used uniformly distributed particles with $N = 1,000$ particles for each dipole. For a three dipole system, the maximum number of particles is $3N + N_q = 3,200$, which is much less than the 100,000 particles used in [25]. With $\mathbf{x}_k = [\mathbf{r}_k^{\mathrm{T}}\ \mathbf{q}_k^{\mathrm{T}}\ s_k]^{\mathrm{T}}$, the dipole state transition model in Equation (5.13) is a random walk model with Gaussian transition kernel $p(\mathbf{r}_k|\mathbf{r}_{k-1}) = \mathcal{N}(\mathbf{r}_{k-1}, \sigma_r)$ and $p(\mathbf{q}_k|\mathbf{q}_{k-1}) = \mathcal{N}(\mathbf{q}_{k-1}, \sigma_q)$, with $\sigma_r = 1$ cm and $\sigma_q = 2$ nA.

Each existing dipole has a probability of survival $\Pr_{k|k-1}(\mathbf{x}_{k-1}) = 0.8$ and a probability of detection $\Pr_k^{det}(\mathbf{x}_k) = 0.95$.

5.6.2 *Eigenvalue threshold selection*

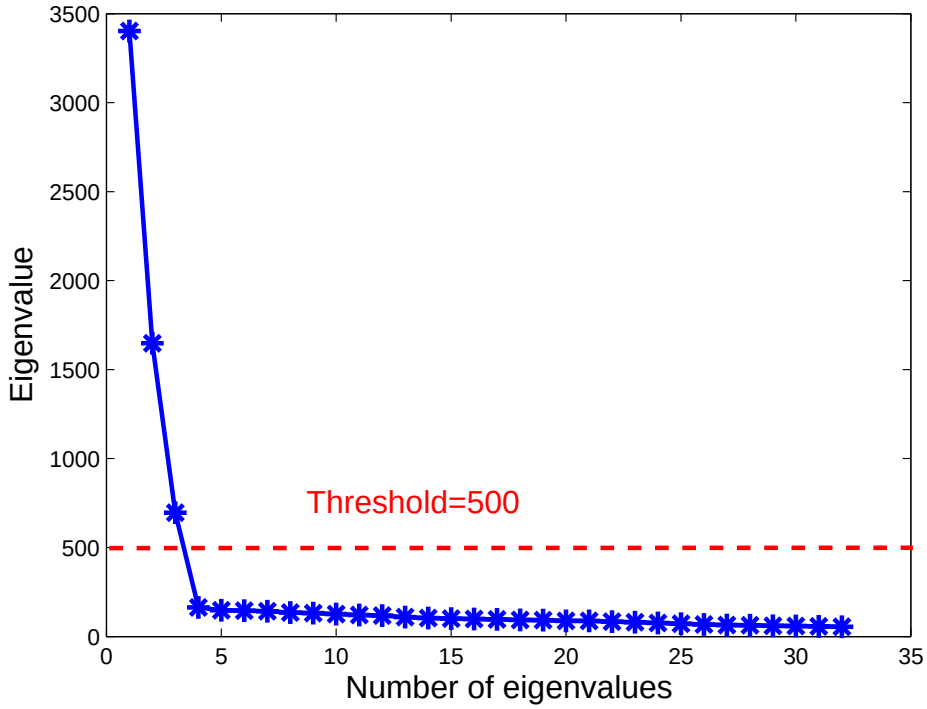


Figure 5.3: Eigenvalues of EEG covariance matrix for simulated data.

In Section 5.5, we discussed that eigenvalue decomposition can be used to reduce the dimension of EEG data by choosing several leading eigenvalues instead of all the eigenvalues. Eigenvalue selection method is crucial since it determines the number of independent components and the reconstruction error of ICA. A threshold based method is used to select the leading eigenvalues. Figure 5.3 shows the amplitudes of all the eigenvalues of the covariance matrix.

In order to find the optimal threshold for leading eigenvalue selection, we use the root mean square error (RMSE) of the reconstructed EEG data as the cost function. The RMSE is defined as the difference of estimated individual EEG data $\hat{\mathbf{z}}_{k,j}$ obtained using Equation (5.12) and the true

Table 5.1: RMSE of reconstructed EEG data for different threshold values.

Threshold	500	150	130	100	50
Dipole1	2.68 μV	2.91 μV	4.97 μV	7.45 μV	NaN
Dipole2	2.67 μV	2.87 μV	5.08 μV	8.35 μV	NaN
Dipole3	2.53 μV	2.79 μV	4.16 μV	7.95 μV	NaN

EEG data $\mathbf{z}_{k,j}$

$$\mathrm{RMSE}_j^{\mathrm{rec}} = \left[\frac{1}{K} \sum_{k=1}^{K} \frac{1}{M} (\mathbf{z}_{k,j} - \hat{\mathbf{z}}_{k,j})^{\mathrm{T}} (\mathbf{z}_{k,j} - \hat{\mathbf{z}}_{k,j}) \right]^{1/2}$$

where K is the number of time steps and M is the number of sensors. Table 5.1 shows the RMSE of the reconstructed EEG data for different threshold values. We can see that as the threshold value decreases, the RMSE increases and that there is a significant RMSE degradation when the threshold is smaller than 150. In addition, with thresholds smaller than 100, we can hardly distinguish dipole signals with noise which causes the FastICA algorithm to fail. Based on these results, we choose a threshold value of 500 for the rest of the simulations.

5.6.3 Window length selection

The choice of window length L_w is crucial as it greatly impacts the estimation results. Table 5.2 shows the reconstruction RMSE with respect to different L_w. We can see from the table that if the window length is too long (much longer than the duration of dipole), the independent component analysis cannot capture the changing information of the dipole. As a result, the reconstruction error is large. If the window length is too small, the samples in the window cannot statistically represent the whole data, which also leads to larger reconstruction error. Based on these results, we choose a window length $L_w = 100$ samples with sampling rate 1 kHz.

Table 5.2: RMSE of reconstructed EEG data for different window lengths.

Window length	50	100	250	500
Dipole 1	2.78 μV	2.16 μV	2.33 μV	4.22 μV
Dipole 2	2.68 μV	2.21 μV	2.42 μV	4.51 μV
Dipole 3	2.60 μV	2.13 μV	2.44 μV	4.13 μV

5.6.4 Estimation results

The eigenvalue selection threshold was chosen as 500 and the window length as $L_w = 100$. We used 1,000 particles for each existing dipole and 200 particles for the new dipoles. The estimation

results for the amplitudes of three dipoles are shown in Figure 6.3. The RMSE for the dipole current amplitude is 2.07 nA. Figure 6.4 shows the estimation results for the dipoles 3-D location; the position RMSE is 6.93 mm. The orientation estimation results are demonstrated in Figure 5.6. We compared the performance of the proposed tracking algorithm with those in [20, 22, 25], as shown in Table 6.2. Compared to [20] which has the best tracking performance, the proposed PF-PHDF based system has a tracking error of 1.5 mm. However, the PF-PHDF only uses 3,200 particles (compared to 100,000) and also does not require knowledge of the number of dipoles. Compared to [25] which also tracks an unknown number of dipoles, the tracking error is only 0.7 mm but the number of particles used is only 3,200 compared to 100,000. In summary, the proposed system obtains comparable estimation results with significantly reduced computational complexity for tracking unknown number of dipoles.

Table 5.3: Comparison of neural activity tracking for synthetic data.

Approach	Number of particles	Number of dipoles	Knowledge of dipole number	RMSE of location
PF [20]	100,000	4	Known	5.4 mm
RB-PF [22]	50,000	2	Known	6.3 mm
D-PF [25]	100,000	3	Unknown	6.2 mm
PF-PHDF	16,000	3	Unknown	6.2 mm
PF-PHDF	3,200	3	Unknown	6.9 mm

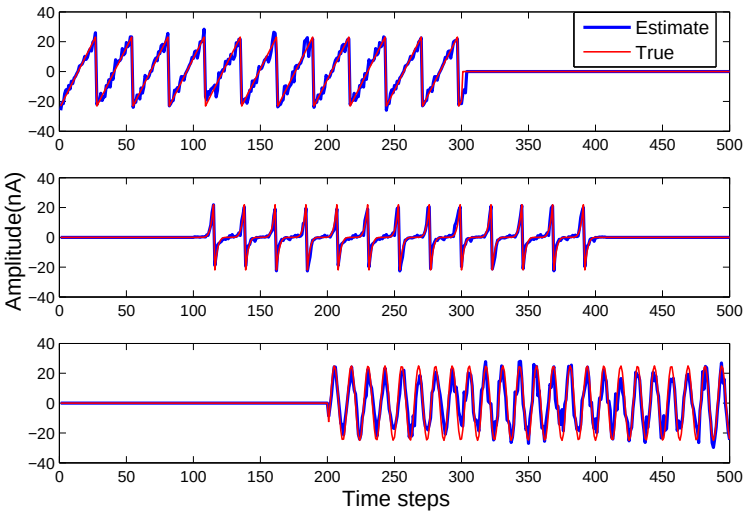


Figure 5.4: Amplitude tracking result of three dipoles for synthetic EEG data.

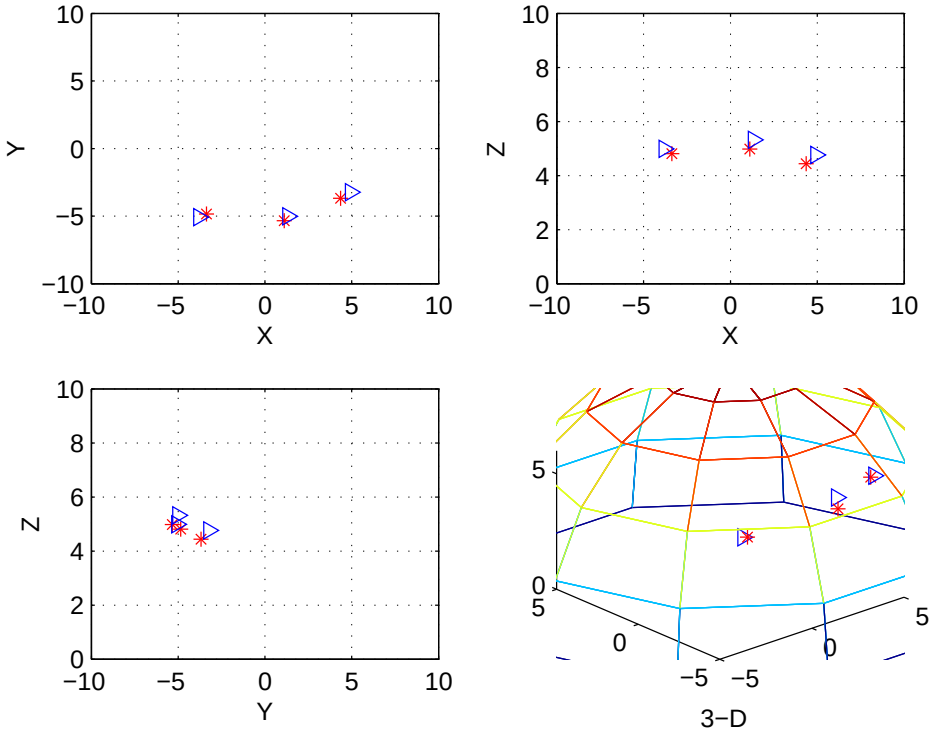


Figure 5.5: Estimated 3-D locations of dipoles for simulated EEG data (red star–true location; blue triangle–estimated location).

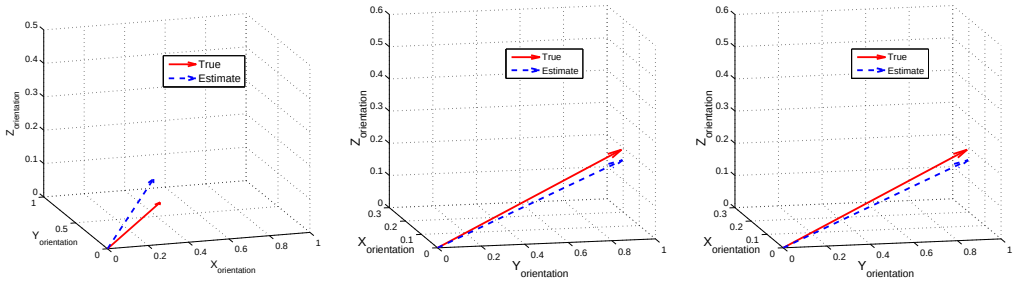


Figure 5.6: Orientation estimation results of three dipoles for simulated EEG data.

5.6.5 Real EEG data tracking results

We applied the proposed PF-PHDF algorithm to a real EEG data set from a visual experiment described in [86]. This experiment tracks brain activity when the subject reacts to green squares appearing on the screen. We first used the EEG data from one trial as training data and found the threshold for eigenvalue selection. The result in Figure 5.7 is similar to the one in Figure 5.3.

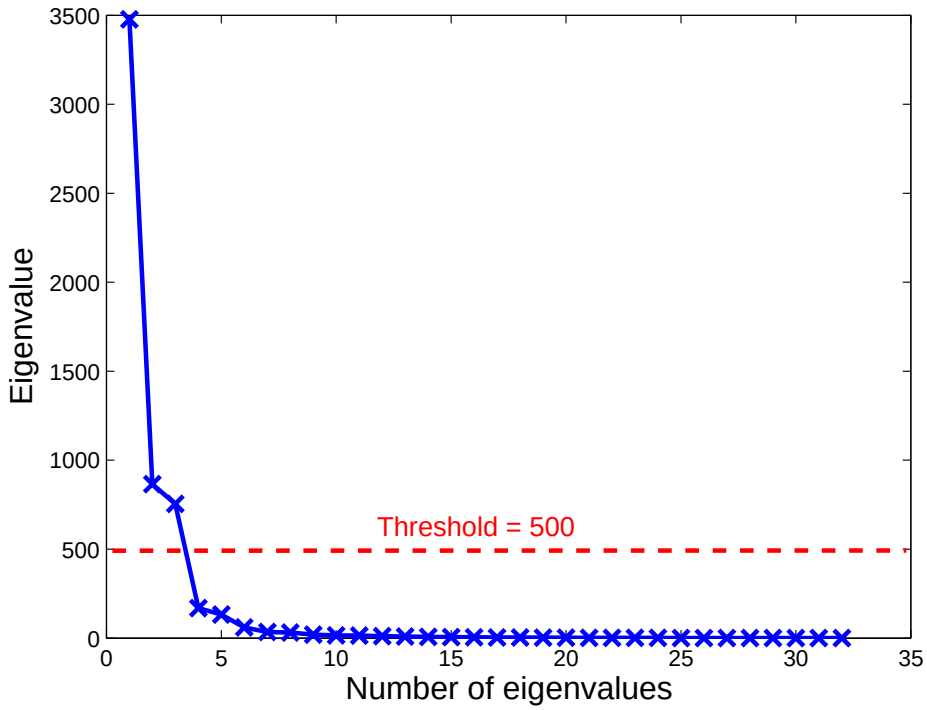


Figure 5.7: Eigenvalues of EEG covariance matrix for real data.

As there is a significant drop when the eigenvalues are smaller than 500, we choose the threshold as 500; this is the same value used for the synthetic data simulation. The sampling rate of this experiment is 128 Hz. Since the frequency of the stimuli is once every 3 s and the subject pushes the button about 1 s after the stimuli, we assume that the states and number of dipoles will change every 1 s. As a result, we choose the window length to be $L_w = 128$ samples.

We show the estimation results for one trial spanning 3 s. The EEG data for 5 (out of 32) channels is shown in Figure 5.8. Figure 5.9 shows the estimation result of the 3-D location for the dipoles. Since the true location of the dipoles is unknown, we use the estimation result of a standard dipole fitting procedure [87] as the ground truth. The RMSE of the proposed PF-PHDF for the 3-D dipole location calculated with respect to [87] is 2.1 mm. From Figure 5.9, we can see that during the first second, there is only one dipole which means that in this period most of the brain is in the inactive state. During the next two seconds, the number of dipoles increased to three. At 0.78 s, the subject received a stimulus from the screen and made a response by pushing the button at 1.13 s. As a result, certain neurons located in the posterior of the brain were activated

which led to the increase in number of dipoles.

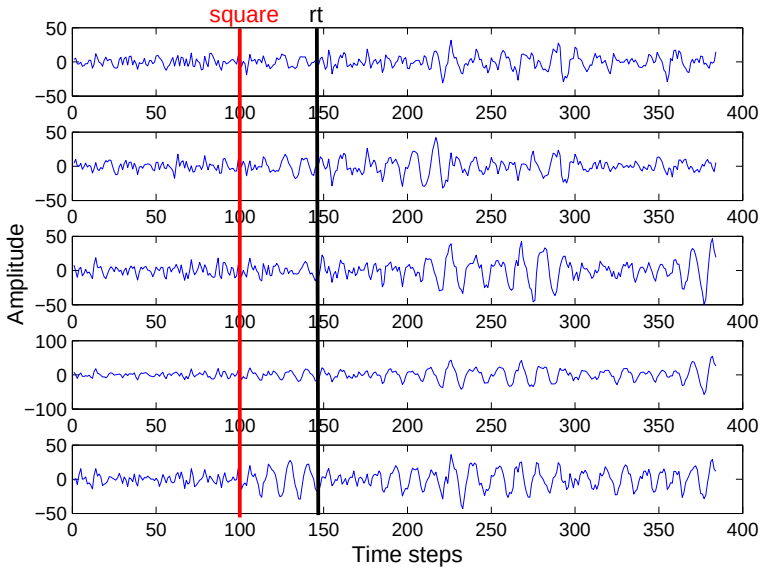


Figure 5.8: One typical cycle of the EEG data (5 channels and 3 seconds).

5.7 Hardware implementation of proposed PF-PHDF

In this section, we describe an efficient multi-processor architecture for the PF-PHDF. Pre-whitening and FastICA can be implemented using the architecture in [85, 88] and are not described here.

5.7.1 Architecture overview

The high level block diagram of the proposed PF-PHDF architecture is shown in Figure 5.10. It consists of P processing elements (PE) and one central unit (CU) connected by a global bus. Local processing steps, such as initialization (Step 1), prediction (Step 2) and part of updating (Step 3) are conducted in each PE. Global processing steps, such as computing normalization factors C_k, estimating the number of dipoles (Step 3), resampling (Step 4) and clustering (Step 5), are executed in the CU. Each PE communicates with the CU, but there is no communication among PEs.

The operation flow for the PF-PHDF is shown in Figure 5.11. Each PE processes R_k/P particles, where R_k is the number of particles at time step k. First, the particles $\mathbf{x}_{k-1}^{(i)}$ are processed in the prediction unit to generate the new particles $\tilde{\mathbf{x}}_k^{(i)}$ by sampling the transition density $p_{k|k-1}(\cdot|\mathbf{x}_{k-1}^{(i)})$. The predicted weights $\tilde{w}_{k|k-1}^{(i)}$ are calculated in the prediction unit. Next, the likelihoods $\psi_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$ are calculated for each individual measurement $\mathbf{z}_{k,j}$, for each particle $\tilde{\mathbf{x}}_k^{(i)}$. Since the calculations of

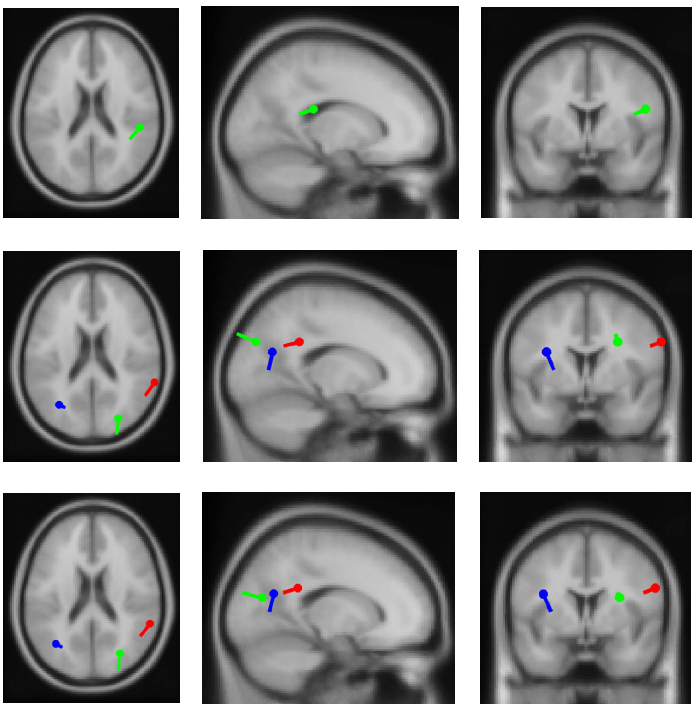


Figure 5.9: 3-D dipole location estimation result for the EEG data. (a) dipole location for the 1st second (top three); (b) dipole location for the 2nd second (middle three); (c) dipole location for the 3rd second (bottom three).

$\psi_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$, $j = 1,\ldots,J$ are independent, they are implemented in parallel. After obtaining the likelihoods for all particles, $\psi_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$, $i = 1,\ldots,R_k$, the sum of the likelihoods $\sum_i \psi_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$ is sent to the CU by each PE. Then, the central unit calculates the normalization factor, $C_k(\mathbf{z}_j) = \sum_p \sum_i \psi^p_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$ and sends it back to each PE; each PE then computes the final weights $\tilde{w}_k^{(i)}$ based on Equation (5.15) in the final weight unit.

Since the resampling step is operated in the CU, the weights of all the particles have to be transferred from the PEs to the CU, which results in a large communication overhead. In order to reduce this overhead, we employ the grouping method in [50] and add the group-and-mean unit in each PE. The main idea is as follows. The particles $\tilde{\mathbf{x}}_k^{(i)}$ and their corresponding weights $\tilde{w}_k^{(i)}$ are divided into G groups based on the range of the particles; the averages of each group, $x^{(g)}_{k,\text{mean}}$, $g = 1,\ldots,G$, are used as the new particles, and only the average weights $w^{(g)}_{k,\text{mean}}$, $g = 1,\ldots,G$, are transmitted to the CU to be used as input to the resampling step. These particles are stored in the mean particle memory (MPMEM) for future use. Before the resampling step, we estimate the

Table 5.4: Hardware operators for each block in Figure 5.11.

Unit	Block	Additions	Multiplications	Divisions	Square Roots	Exponentials
PE	Sampling	6	0	0	0	0
	Likelihood	88	93	1	2	1
	Group mean	52	0	1	0	0
	Final weight	6	1	1	0	0
CU	Global	6	0	0	0	0
	Normal	9	0	0	0	0
	Resampling	2	3	1	0	0
	Clustering	6	3	1	1	0

number of dipole sources by summing the final weights, $\hat{N}_{d_k} = \sum_i \hat{w}_k^{(i)}$. During the resampling step, the replication index ρ is calculated at the CU based on the average weights. After the resampling step, the group averaged particles are read from MPMEM and sent to the prediction unit for the computations in the next iteration. These particles are also sent to the clustering unit at the CU, and are used to estimate the dipole parameters.

This procedure significantly reduces the communication between the PEs and the CU. Figure 5.11 depicts eight data transactions (DT): DT 1 corresponds to the local extremum, $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$, transmitted from the PEs to the CU; DT 2 is the summation of the likelihoods, $\sum_i \psi_{k,\mathbf{z}_j}(\tilde{\mathbf{x}}_k^{(i)})$ that is sent by the PEs to the CU; DT 3 corresponds to the normalization factors, $C_k(\mathbf{z}_j)$, sent by the CU to the PEs; DT 4 corresponds to the updated weights, $\tilde{w}_k^{(i)}$, sent by the PEs to the CU; DT 5 corresponds to the global extremum, $\mathbf{x}_{\text{Min}}$ and $\mathbf{x}_{\text{Max}}$, from the CU to the PEs; DT 6 corresponds to the weights with the average values, $w_{k,\text{mean}}^{(g)}$, sent by the PEs to the CU; DT 7 corresponds to the replication index, ρ, sent by the CU to the PEs; and DT 8 corresponds to the updated particles, $\mathbf{x}_k^{(i)}$, sent by the PEs to the CU. The hardware resource for each block in Figure 5.11 is shown in Table 5.4. Note that the likelihood computation unit is the most demanding part in terms of resources.

5.7.2 Data windowing for on-line processing

Most methods process MEG/EEG data off-line, after all of the data has been collected [89]. Our proposed algorithm can be used to process MEG/EEG data on-line by windowing the data and then applying the algorithm on each windowed data segment in a pipelined fashion. Figure 5.12 shows how we pre-whiten and run FastICA on the data from Window 1, while obtaining data in Window 2. The length of the window, L_w, is a critical parameter because the processing time and tracking

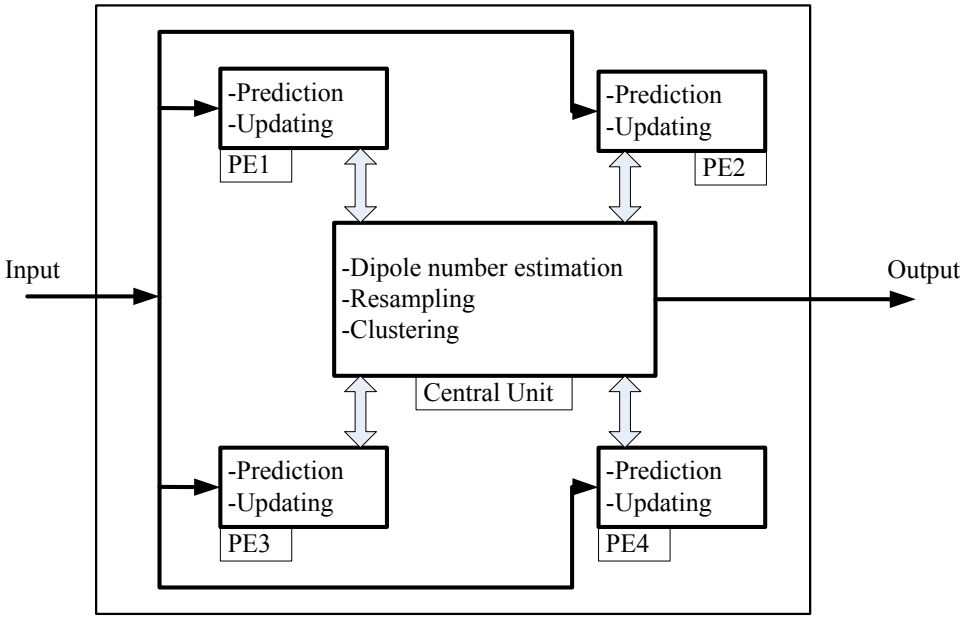


Figure 5.10: Parallel PF-PHDF architecture with four processing elements: PE1, PE2, PE3, and PE4.

accuracy both depend on it. If L_w is small, the computations take less time. Unfortunately, the data in a short window do not provide an accurate statistical representation of neural activity and result in a large estimation error (see Table 5.2). We also require the window length to be larger than the execution time of pre-whitening and FastICA, as shown in Figure 5.12.

5.7.3 Computational complexity analysis

Even with the computational complexity reduction due to the use of eigenvalue distilling, the proposed estimation system is still computationally intensive. Table 5.5 lists the number of computational operations for the case when $M = 32$ sensors, $N_d = 3$ dipole sources, and we choose a window length $L_w = 100$ samples and 3,200 particles for processing. The pre-whitening step includes the computation of of the measurement covariance matrix, which is computationally intensive. The PF-PHDF computational complexity is the largest, creating a system bottleneck. In the next section, we describe a hardware implementation of the PF-PHDF which makes use of pipelining and parallel processing to reduce the computation time.

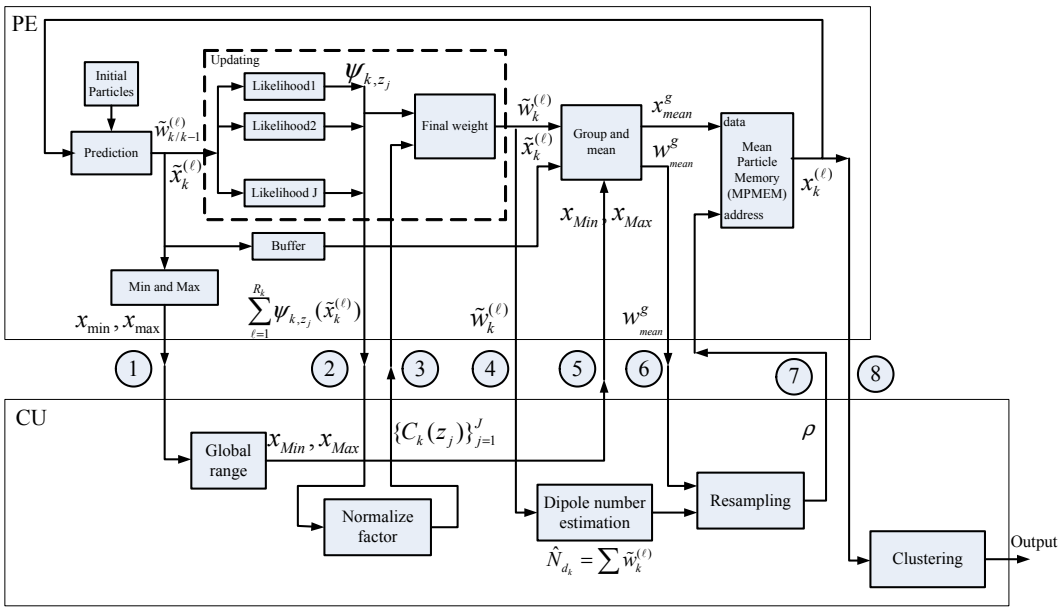


Figure 5.11: Operation flow and data transactions between PE p and the central unit (CU) at each time step.

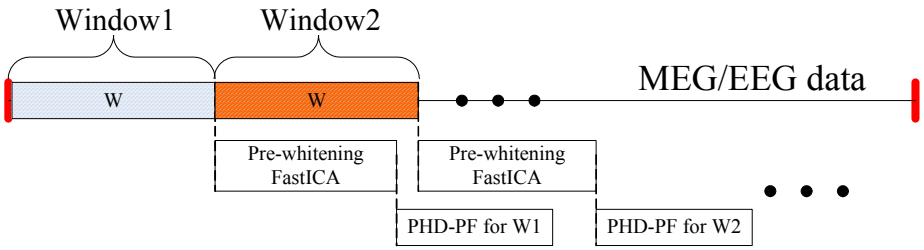


Figure 5.12: Pipelined window processing of MEG/EEG data.

Table 5.5: Number of computational operations used in the proposed dipole source estimation system in Figure 5.2

Block	Additions	Multiplications	Divisions	Square roots	Exponentials
Pre-whitening	107,680	107,680	11	10	0
FastICA	3,070	8,070	10	10	0
PF-PHDF	560,000	320,000	16,000	9,600	3,200

5.7.4 Hardware implementation results

The proposed PF-PHDF hardware architecture is implemented using Verilog HDL and synthesized on the Xilinx Virtex-5 device. The design is verified using Modelsim.

*R*esource Utilization: Table 5.6 summarizes the architecture resource utilization for the 4 PE par-

allel architecture for the PF-PHDF. Since the total number of particles is 3,200, each PE processes 800 particles. For the likelihood calculation in the updating step, the exponential functions are implemented using CORDIC units, and the rest of the units are implemented using DSP cores.

Table 5.6: Resource utilization for PF-PHDF on Xilinx XC5VSX240k

Unit	Occupied slices	Slice Reg.	Slice LUTs	Block Ram	DSP48Es
PF-PHDF	14,291 (39%)	43,637 (30%)	42,383 (29%)	134 (26%)	283 (27%)

*E*xecution Time: Figure 5.13 shows the timing performance for one iteration of the proposed PF-PHDF; the actual number of cycles is given in Table 5.7. One iteration takes $N_{\text{total}}=(N_s+N_g+N_{gm}+N_c)$=4,852 cycles. We choose a system clock rate of 100 MHz and so the processing time for one iteration is only $T_{\text{total}}=N_{\text{total}}\,T_{\text{clk}}=48.52\mu$s. Based on the pre-whitening and FastICA implementation in [85, 88], the execution time for preprocessing is about 265 μs. Thus, for a window with 100 samples spanning 1 s, the total processing time is $265+(48.52\times 100)=5,117$ μs or 5.1 ms.

Table 5.7: Execution cycles for each block

Unit	N_s	N_l	N_g	N_n	N_w	N_{gm}	N_k	N_r	N_c
Cycles	804	827	5	4	818	832	4	53	3211

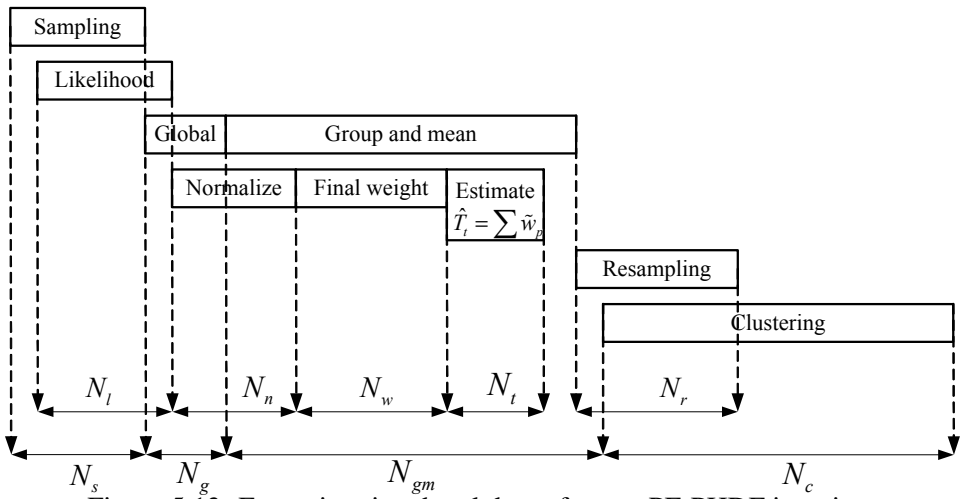


Figure 5.13: Execution time breakdown for one PF-PHDF iteration.

*S*calability: For the proposed tracking system, the number of particles used for each dipole N is a

critical parameter as it impacts the estimation accuracy and processing time. Figure 5.14 shows the tracking performance in terms of RMSE for dipole location and the processing time for PF-PHDF with respect to N. Here we choose the number of PEs $P = 4$. From Figure 5.14, we can see that as N increases, the RMSE decreases and the processing time increases. However, when N is greater than 1000, there is no significant improvement in the RMSE but the processing time increases rapidly. A good tradeoff between RMSE and processing time is obtained at $N = 1000$.

For the real data case corresponding to the visual experiment of a person tracking green squares, the maximum number of dipoles was small (less than 5). From the FPGA timing results, we project that if the maximum number of dipoles is 3, the proposed system can perform real-time processing at sampling rates of up to 10 kHz for window length $L_w = 100$ samples. However, for epilepsy patients, the number of dipoles during seizures can be greater than 10 [84]. Figure 5.15 shows the timing performance of the proposed system with respect to the maximum number of dipoles for 1,000 particles per dipole and $P = 4$. From Figure 5.15, we can see that as the maximum number of dipoles increases, the processing time for a window grows, as expected. Even when the number of dipoles is as large as 15, our system can still support real-time tracking with sampling rate of about 1.5 kHz.

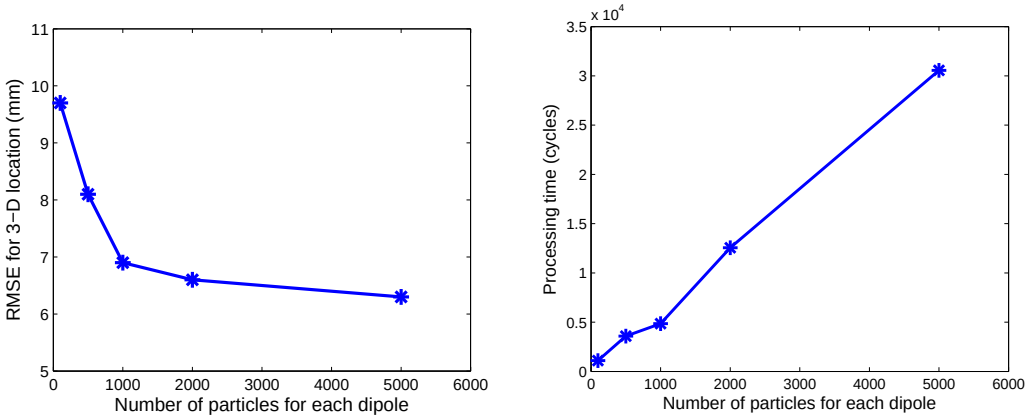


Figure 5.14: RSME for location and processing time for PF-PHDF with respect to N.

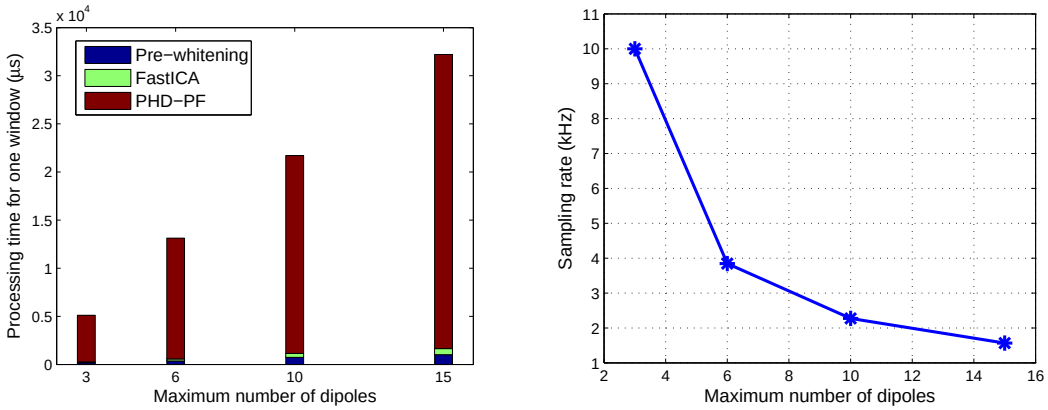


Figure 5.15: Scalability of the proposed system with respect to maximum number of dipoles: Effect on (a) processing time and (b) maximum sampling rate for real-time processing.

Chapter 6

Efficient EEG/MEG Tracking System Design

In Chapters 4 and 5, we proposed advanced signal processing algorithms to estimate the parameters of neural dipole sources in EEG/MEG systems. In this chapter, we focus on increasing the hardware efficiency of an EEG system though similar techniques can be used for our MEG system.

A typical EEG system contains tens and hundreds of sensors. Even low power EEG sensors that have been designed to make wearable EEG systems feasible, have a power consumption of 10 mW per sensor [55, 56]. In this chapter, we focus on reducing the number of sensors required in an EEG system without affecting the tracking performance. We propose two methods to achieve this objective: sensor scheduling [57, 58] and compressive sensing (CS) [59]. Sensor scheduling is implemented by adaptively configuring the EEG sensors at each time step using the minimum predicted mean squared error (PMSE) or maximum signal-to-noise ratio (SNR) as the performance metric. Compressive sensing uses the sparsity of EEG signal to compress the EEG data and then tracks the parameters of neural dipole sources using the compressed EEG data.

6.1 Sensor scheduling of EEG/MEG system

In this section, we propose a sensor scheduling algorithm that enables adaptive use of sensing resources for neural activity tracking. Our goal is to develop an algorithm that has good tracking performance with fewer number of active sensors. Since the power consumption of each EEG sensor is about 10 mW [55, 56], such an algorithm can reduce the total power consumption, making wearable EEG devices common place.

Sensor scheduling is a closed-loop feedback optimization procedure that allows adaptive selection of the sensors to be used for obtaining measurements at the next time-step in order to optimize the cost function of interest. Here we consider two cost functions: (a) minimization of the PMSE in estimation, and (b) maximization of the SNR of the measurements, and optimize them with respect to the sensor configurations.

6.1.1 Sensor scheduling based on minimization of PMSE

Minimum predicted mean squared error (PMSE) has been used as a cost function for sensor scheduling [57, 58]. The PMSE in dipole state estimation at time k can be written as

$$\begin{aligned} \mathscr{J}^P(\mathbf{y}_k) &= E_{\mathbf{x}_k,\mathbf{z}_k}[(\mathbf{x}_k-\hat{\mathbf{x}}_k)(\mathbf{x}_k-\hat{\mathbf{x}}_k)^T] \\ &= \int\int (\mathbf{x}_k-\hat{\mathbf{x}}_k)(\mathbf{x}_k-\hat{\mathbf{x}}_k)^T p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{y}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1},\mathbf{y}_{k-1})d\mathbf{x}_k d\mathbf{z}_k, \end{aligned} \tag{6.1}$$

where $\mathbf{x}_k$ denotes the dipole state at time k and $\mathbf{y}_k$ is the sensor configuration at time k. Here, we consider the sensor configuration problem with M sensors. The sensor configuration vector $\mathbf{y}_k = [y_{k,1}\, y_{k,2} \ldots y_{k,M}]^T$ is comprised of binary values $y_{k,m} \in \{0,1\}$, with $y_{k,m} = 1$ indicating that the mth sensor is selected and $y_{k,m} = 0$ indicating that the mth sensor is not selected. The measurement $\mathbf{z}_k$ is related to $\mathbf{y}_k$ and $\mathbf{z}_k(\mathbf{y}_k) = \mathbf{z}_{k,\mathscr{M}}$, where $\mathscr{M} = \{m : y_{k,m} = 1\}$. $\hat{\mathbf{x}}_k$ is the estimate of $\mathbf{x}_k$ at time k computed using the measurement $\mathbf{z}_k$. Using Monte Carlo integration, $\mathscr{J}^P(\mathbf{y}_k)$ can be approximated as

$$\mathscr{J}^P(\mathbf{y}_k) \approx \sum_{\ell=1}^{N_k} w_k^{(\ell)} \frac{1}{J} \sum_{j=1}^{J} (\mathbf{x}_k^{(\ell)} - \hat{\mathbf{x}}_k(\mathbf{z}_k^{(j,\ell)}))(\mathbf{x}_k^{(\ell)} - \hat{\mathbf{x}}_k(\mathbf{z}_k^{(j,\ell)}))^T, \tag{6.2}$$

where $\mathbf{x}_k^{(\ell)}$ and $w_k^{(\ell)}, \ell = 1,\ldots,N_k$, are the particles and weights in the PF representation of the predicted state distribution $p(\mathbf{x}_k|\mathbf{z}_{k-1},\mathbf{y}_{k-1})$, $\mathbf{z}_k^{(j,\ell)} \sim p(\mathbf{z}_k|\mathbf{x}_k^{(\ell)},\mathbf{y}_k)$, $j = 1,\ldots,J$, are i.i.d. samples drawn from the measurement model, and $\hat{\mathbf{x}}_k(\mathbf{z}_k^{(j,\ell)})$ is the estimate of $\mathbf{x}_k$ computed using the predicted measurement $\mathbf{z}_k^{(j,\ell)}$ with a secondary PF. The approximation error is small for large N_k and J. The PMSE $\mathscr{J}^P(\mathbf{y}_k)$ can now be optimized by searching through all sensor configurations. For M sensors, the total number of possible sensor configurations is $R = 2^M$, which can be quite large. We impose a constraint to limit the total power concumption to $\mathscr{P}$ as shown below

$$\sum_{m=1}^{M} y_{k,m} C_m \leq \mathscr{P}, \tag{6.3}$$

where C_m is the power consumption of the mth sensor. A block diagram of the dipole state estimation and sensor scheduling method is shown in Figure 6.1. The key steps of the sensor scheduling algorithm, performed at each time k, are as follows:

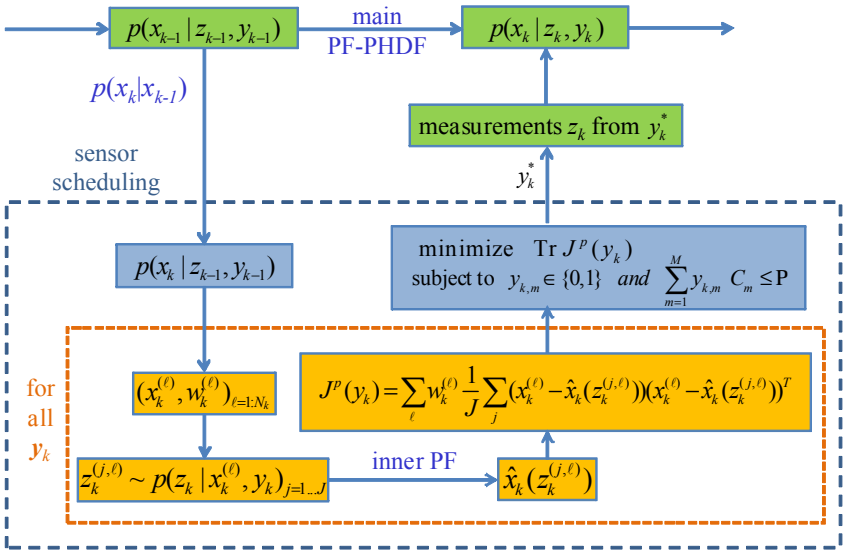


Figure 6.1: Block diagram of the state estimation and sensor optimization method.

1. **State prediction:** Compute the distribution $p(\mathbf{x}_k|\mathbf{z}_{k-1},\mathbf{y}_{k-1})$ of the predicted state at time k by propagating the PF-PHDF posterior distribution $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1},\mathbf{y}_{k-1})$ at time $k-1$ using the state evolution model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$.

2. **Measurement sampling:** For each particle $\mathbf{x}_k^{(\ell)}$ and sensor configuration $\mathbf{y}_k$, draw predicted measurements $\mathbf{z}_k^{(j,\ell)}$ from the measurement distribution $p(\mathbf{z}_k|\mathbf{x}_k^{(\ell)},\mathbf{y}_k)$.

3. **Inner estimation:** For each measurement $\mathbf{z}_k^{(j,\ell)}$, estimate the state $\hat{\mathbf{x}}_k(\mathbf{z}_k^{(j,\ell)})$ using a secondary PF.

4. **Error calculation:** Calculate the PMSE $\mathscr{J}^P(\mathbf{y}_k)$ using the approximation in Equation (6.2).

5. **Optimization:** Find the optimum sensor configuration $\mathbf{y}_k^*$ which minimizes the PMSE based on the power constraint in Equation (6.3).

The scheduled sensor configuration $\mathbf{y}_k^*$ is then used to obtain the measurement $\mathbf{z}_k(\mathbf{y}_k^*)$ at time k and estimate the state $\hat{\mathbf{x}}_k$ using the PF-PHDF algorithm.

For M sensors, there are a total of $R = 2^M$ possible sensor configurations, making the optimization problem very difficult to solve directly unless M is small (e.g. $M = 10$). For large M, the current best methods for solving such problems include branch and bound methods [90] (which provide global optima at the expense of computational cost) or other convex relaxation based techniques [91] (which are fast but only provide approximate solutions). Even so, sensor scheduling based on the direct computation and optimization of the PMSE remains computationally demanding and not suitable for real-time implementation. As a result, in this study, we did not implement this sensor scheduling method. In the next section, we propose another sensor scheduling method based on maximization of SNR which has lower computational complexity and is much easier to implement.

6.1.2 Sensor scheduling with maximum SNR

We now describe a sensor scheduling algorithm that relies on maximization of the signal-to-noise ratio (SNR) of the measured sensor data. This approach leads to significant reduction in the number of required sensors for accurate tracking performance, with a computational complexity that is much lower than the minimum PMSE based sensor scheduling method discussed earlier.

The proposed method is as follows. Suppose that at some time step k the parameters of the dipole sources $(\mathbf{r}_k, \mathbf{q}_k, \mathbf{s}_k)$ are given and fixed. From Equations (4.1) and (4.2), the SNR of the measurement from the mth sensor can be represented by

$$SNR_{k,m} = \frac{(\mathbf{A}_{k,m}\,\mathbf{s}_k)^2}{\sigma_n^2} = \frac{(\sum_{i=1}^{N_d} a_{k,(m,i)}\,s_{k,i})^2}{\sigma_n^2} \propto (\sum_{i=1}^{N_d} a_{k,(m,i)}\,s_{k,i})^2, \tag{6.4}$$

where $\mathbf{A}_{k,m}$ is the mth row of the gain matrix, σ_n^2 is the variance of measurement noise and N_d is the number of dipole sources. After ICA, the measurement is decomposed to independent components where each component corresponds to an individual dipole source. The SNR of the mth sensor for the ith dipole source can be represented as

$$SNR_{k,(m,i)} \propto (a_{k,(m,i)}\,s_{k,i})^2, \tag{6.5}$$

where $a_{k,(m,i)}$ is a function of the distance $d_{k,(m,i)}$ between the mth sensor and ith dipole source. Figure 6.2 shows the relationship between $|a_{k,(m,i)}|$ and $d_{k,(m,i)}$. From Figure 6.2 we can see that as $d_{k,(m,i)}$ increases, $|a_{k,(m,i)}|$ decreases and so does the corresponding SNR. Since the MSE is expected

to be lower with higher SNR measurements, using sensors with smaller $d_{k,(m,i)}$ can provide better neural tracking performance. The sensors are therefore scheduled by adaptively selecting the ones with smallest $d_{k,(m,i)}$ to form the final sensor configuration used for obtaining the measurements. The steps of the sensor scheduling method, performed at each time k, are as follows:

1. **State prediction:** Calculate the predicted state of the dipole source $\tilde{\mathbf{x}}_k$ at time k using the particles $x_{k-1}^{(\ell)}$ and weights $w_{k-1}^{(\ell)}$ at time $k-1$ based on the state model

$$\tilde{\mathbf{x}}_k = \sum_{\ell=1}^{N_k} p_{x_{k-1}|x_k}(x_{k-1}^{(\ell)})\, w_{k-1}^{(\ell)},$$

   where N_k is the number of particles and $p_{x_{k-1}|x_k}(\cdot)$ is the state updating equation. Extract the predicted dipole location $\tilde{\mathbf{r}}_k$ from $\tilde{\mathbf{x}}_k$.

2. **Distance calculation:** For each sensor, calculate the distance between the sensor location $\mathbf{r}_m$ and the predicted dipole source location $\tilde{\mathbf{r}}_k$ as

$$\tilde{d}_{k,m} = ||\mathbf{r}_m - \tilde{\mathbf{r}}_k||, \quad m = 1, \ldots, M,$$

   where $||\cdot||$ denotes Euclidean distance.

3. **Optimization:** Sort the sensors in increasing order of $\tilde{d}_{k,m}$. Choose the first $\mathfrak{N}_s$ sensors to estimate the dipole states, where $\mathfrak{N}_s$ is the number of sensors to be used depending on power constraint in Equation 6.3.

As before, the scheduled sensor configuration $\mathbf{y}_k^*$ is then used to obtain the measurement $\mathbf{z}_k(\mathbf{y}_k^*)$ at time k and estimate the state $\hat{\mathbf{x}}_k$ using the PF-PHDF algorithm.

6.1.3 Algorithm performance results for sensor scheduling

The number of sensors $\mathfrak{N}_s$ after sensor scheduling impacts the power consumption of the EEG system and the tracking performance of the proposed algorithm. Table 6.1 shows the position tracking result in terms of RMSE for different number of sensors. As expected, use of fewer sensors result in higher RMSE but lower sensor power consumption. Thus, $\mathfrak{N}_s$ should be chosen based on the power constraint or the RMSE constraint depending on the application. Since the most important bottleneck of wearable EEG instruments is the battery size [11, 4], here $\mathfrak{N}_s$ is

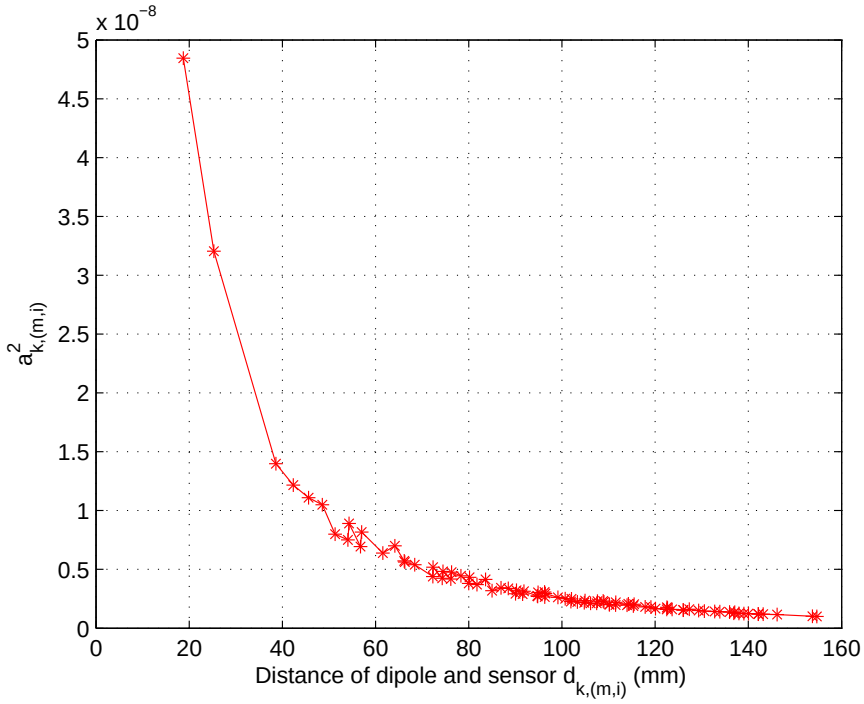


Figure 6.2: Amplitude of the EEG sensor signal as a function of the distance between the sensor and dipole source.

chosen based on the power constraint. In this paper, the power consumption of each wireless EEG sensor is set as $C_m = 10mV$ [55, 56] and the total power constraint is set as $\mathscr{P} = 150\ mW$. Based on Equation (6.3), $\mathfrak{N}_s$ is chosen to be 15.

Table 6.1: Position RMSE for different number of sensors $\mathfrak{N}_s$.

Number of sensors	6	9	15	24	32
RMSE	9.98 mm	6.95 mm	6.41 mm	6.33 mm	6.28 mm

6.1.4 Estimation results with sensor scheduling

In this section, we show the estimation results for synthetic EEG data. Here we choose the eigenvalue selection threshold as 500 and the window length $L_w = 100$. We use 2,000 particles for each existing dipole and 400 particles for the newborn dipole. The tracking result for the amplitudes of three dipoles are shown in Figure 6.3. The RMSE for the dipole current amplitude is 1.83 nA. Figure 6.4 shows the estimation results for the 3-D location of the three dipoles; the position RMSE is 6.41 mm.

We compare the performance of the proposed tracking algorithm with the results in [20, 22, 25] and the comparison is shown in Table 6.2. From Table 6.2, we can see that the proposed PF-PHDF algorithm has comparable tracking performance, with significantly reduced number of particles (only 6,400 compared to 100,000 in [51]). Furthermore, by using the proposed sensor scheduling technique, the number of sensors is reduced from 32 to 15, which means about 50% reduction in sensor power consumption.

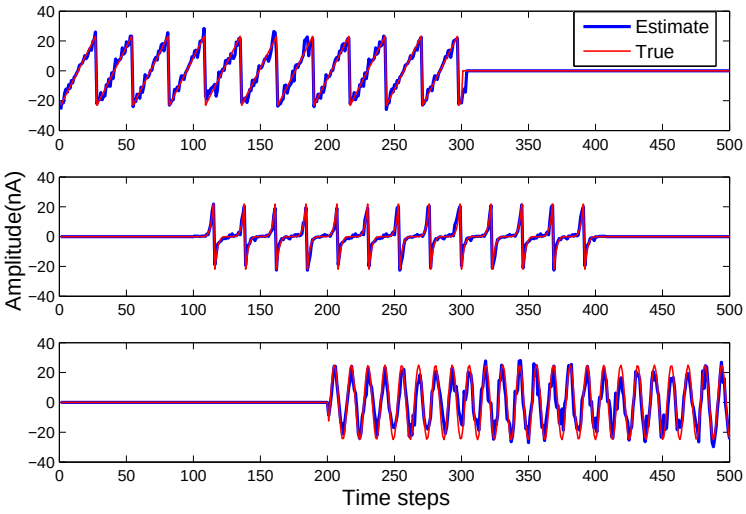


Figure 6.3: Amplitude tracking result of three dipoles for synthetic EEG data.

Table 6.2: Comparison of neural activity tracking for synthetic data.

Approach	Number of particles	Number of dipoles	Knowledge of dipole number	RMSE of location
PF [20]	100,000	4	Known	5.4 mm
RB-PF [22]	50,000	2	Known	6.3 mm
D-PF [25]	100,000	3	Unknown	6.2 mm
PF-PHDF (with SS)	6,400	3	Unknown	6.4 mm

6.1.5 Hardware architecture for sensor scheduling

The overall block diagram of the hardware architecture for sensor scheduling is shown in Figure 6.5. At each time step k, we use the PF-PHDF to obtain the predicted dipole position $\tilde{\mathbf{r}}_k$, and feed $\tilde{\mathbf{r}}_k$ as input to the sensor scheduling unit. First, the square of the predicted distance $\tilde{d}^2_{k,m}$ is calculated using three subtractors, three multipliers and two adders. Next, the sensors are sorted

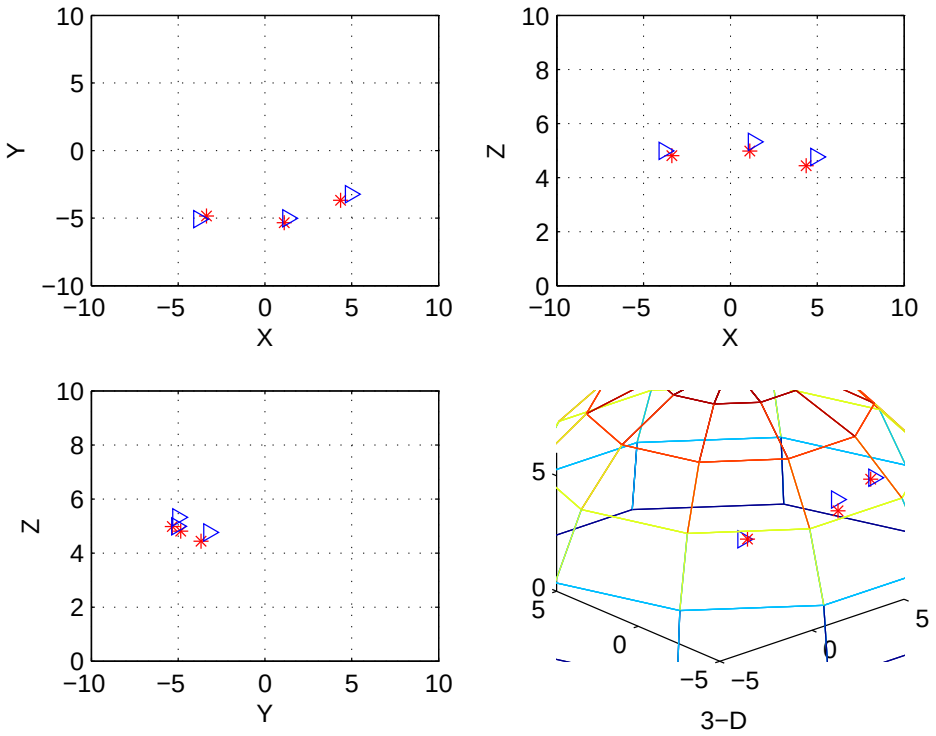


Figure 6.4: Estimated 3-D locations of dipoles for synthetic EEG data.

based on the predicted distance $\tilde{d}^2_{k,m}$ in the SORT unit. After sorting, the top n sensors are picked as the optimized sensor configuration for the next time step. Since we only need to find n sensors with the smallest $\tilde{d}^2_{k,m}$, we maintain a sorted list of n elements and use the insertion-deletion sort algorithm [92] to update the list.

The architecture of the SORT unit is shown in Figure 6.6. It consists of n processors, one adder tree that adds n 1-bit numbers and one rank register. All processors consist of one data register to store the distance value, one comparator to compare with the new distance and one rank register to store the parameter used to calculate the rank of new distance. The rank of the new distance is calculated by the adder tree and stored in the new rank register.

6.1.6 Hardware implementation evaluation of proposed sensor scheduling

The hardware implementations of PF-PHDF and ICA have been described in Section 5.7.4. Here, we show the hardware resource utilization and processing time for the sensor scheduling module. The sensor scheduling algorithm was implemented using Verilog HDL and synthesized on Xilinx

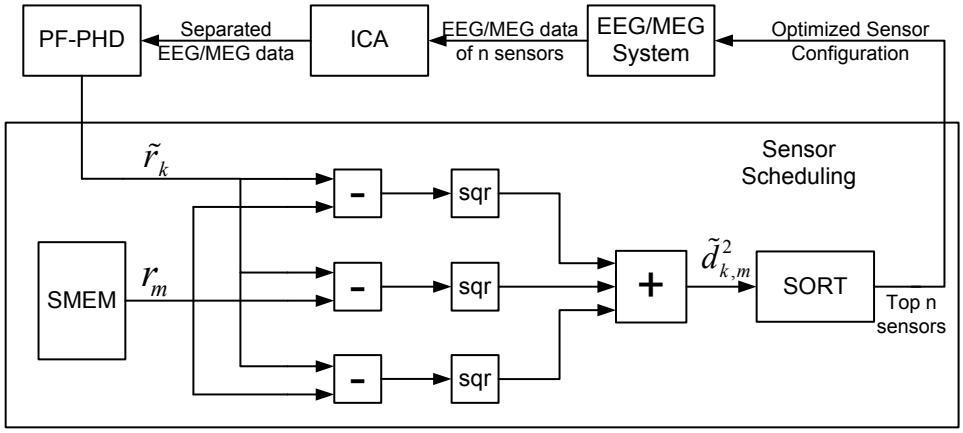


Figure 6.5: Overall architecture of sensor scheduling.

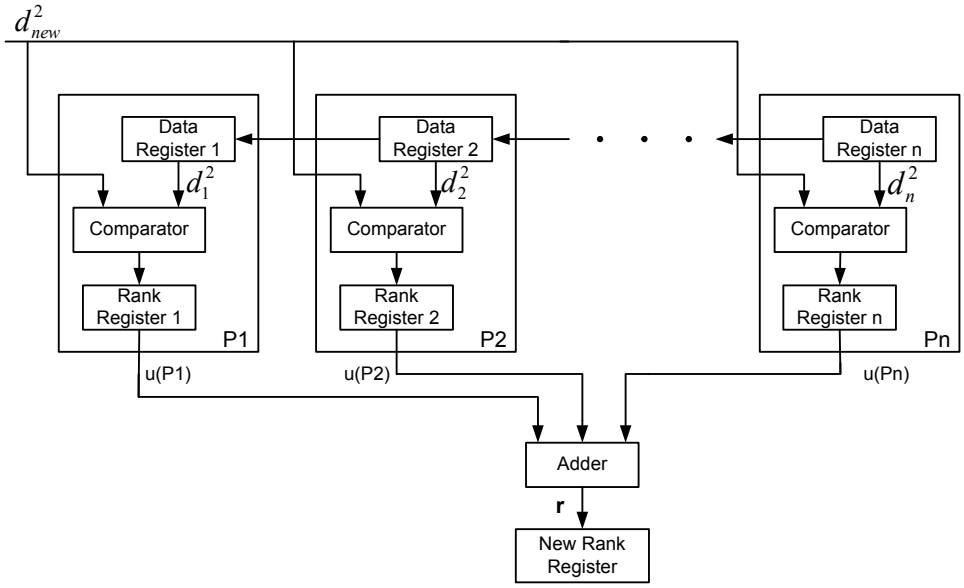


Figure 6.6: Architecture of the SORT unit.

Virtex-5 device (XC5VSX240T). The design was verified using Modelsim.

Resource Utilization: Table 6.3 summarizes the resource utilization for the sensor scheduling architecture shown in Figure 6.5 and 6.6. The square root functions are implemented using CORDIC units. Our resource usage is fairly low, for instance, only about 1% of the regular slices and 2% of the DSP slices in a Xilinx Virtex-5 FPGA (XC5VSX240T).

Execution time: Figure 6.7 shows the timing performance for one iteration of the proposed system; the actual number of cycles is given in Table 6.4. Since some of the computations can be over-

Table 6.3: Resource utilization on Xilinx XC5VSX240T for sensor scheduling

Unit	Occupied slices	Slice Reg.	Slice LUTs	Block Ram	DSP48Es
SS	623 (1%)	1,714 (1%)	1,792 (1%)	3 (1%)	20 (2%)

lapped, the PF-PHDF takes $N_{\text{PF-PHDF}}=(N_s+N_g+N_{gm}+N_c)$=9,150 cycles. We choose a system clock rate of 100 MHz and so the processing time for PF-PHDF is $T_{\text{PF-PHDF}} = N_{\text{PF-PHDF}} \times T_{\text{clk}} =$ 91.5 μs. Based on the pre-whitening and FastICA implementation in [85, 88], the execution time of the preprocessing step for 15 sensors is about 66 μs. The processing time for sensor scheduling is $T_{\text{SS}} = (N_d + N_{sort}) \times T_{\text{clk}} = 4.5$ μs. Thus, for a window with 100 samples, the total processing time with sensor scheduling is $96 \times 100 + 66 = 9,666$ μs.

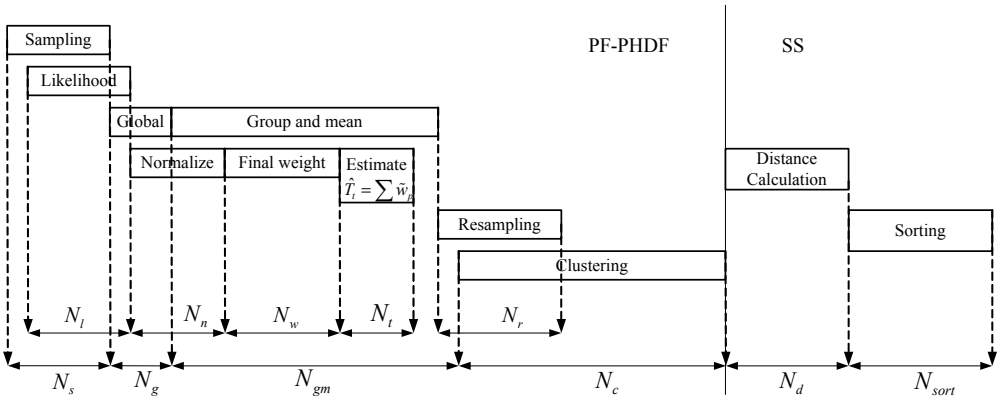


Figure 6.7: Execution time breakdown for one iteration.

Table 6.4: Execution cycles for each block in Figure 5.13

Unit	N_s	N_l	N_g	N_n	N_w	N_{gm}	N_k	N_r	N_c	N_d	N_{sort}
Cycles	1608	1654	5	4	1636	1664	4	53	5873	111	338

6.2 Data compression of EEG/MEG system

In this section, we propose an efficient spatial domain EEG CS technique, which results in a significant reduction in the amount of EEG data that needs to be stored and processed. Sequential Bayesian estimation techniques, such as the particle filtering algorithm [30], have been used to track neural activity dipole sources [93, 94]. However, as the number of dipole sources increases, the computation complexity of the PF tracking algorithm grows proportionally. In order to avoid

this high computational complexity, we first analyze the EEG data sparsity in the spatial domain using equivalent current dipole source representations. We then compressively sense the multiple channel EEG signals using independent and identically distributed Gaussian basis functions. Finally, we apply the PF on the spatial compressed EEG data to localize the neural activities with reduced computational complexity.

6.2.1 Compressive sensing

Compressive sensing (CS) can reduce the number of measurements required to reconstruct a signal since a small set of linear projections of a sparse signal contains enough information for reconstruction. Let $\mathbf{z} \in \Re^{d_z}$ denote a vector with d_z elements; then $\mathbf{z}$ is said to be K-sparse if $\mathbf{z}$ can be represented by $\mathbf{z} = \Omega\theta$, where the columns of $\Omega \in \Re^{d_z \times d_\theta}$ constitute a basis function and θ is a vector with at most K non-zero elements and $K << d_\theta$ [60]. CS theory states that it is highly probable that θ and $\mathbf{z}$ can be exactly recovered from the measurements $\mathbf{y} = \Phi\mathbf{z}$, where $\Phi \in \Re^{d_y \times d_z}$ is a projection matrix and $d_y < d_z$ [59]. It has been shown that the signal can be recovered from its measurements when the projection matrix Φ is incoherent with the basis Ω over which the signal is sparse. A typical choice for the projection matrix, Φ, is a random matrix with independent and identically distributed Gaussian or Bernoulli entries. The vector $\mathbf{z}$ can be recovered from the measurement $\mathbf{y}$ by solving the following optimization problem [59]

$$\arg\min_{\theta} \; ||\theta||_1 \quad \text{subject} \quad \text{to} \quad \Phi\Omega\theta = \mathbf{y}, \tag{6.6}$$

where $||\cdot||_i$ denotes the ℓ_i norm. When $d_y << d_z$, CS theory can be used to reduce the dimensionality of vector $\mathbf{z}$.

6.2.2 Spatial sparsity of EEG signal

At time k, we assume that a small patch of activated cortex can be represented by an equivalent current dipole with three-dimensional (3-D) location $\mathbf{r}_k$ and 3-D moment $\mathbf{m}_k = s_k\,\mathbf{q}_k$, where $\mathbf{q}_k$ is a 3-D orientation vector and s_k is the current amplitude of the dipole [95]. Using the dipole source model, EEG signals acquired by M sensors are represented as

$$\mathbf{z}_k = \mathbf{A}_k\,\mathbf{s}_k + \mathbf{n}_k. \tag{6.7}$$

where $\mathbf{z}_k = [z_k^1\; z_k^2 \ldots z_k^M]^{\mathrm{T}}$ is the M channel EEG signal at time k, $\mathbf{s}_k = [s_k^1\; s_k^2\; \ldots s_k^{N_d}]^{\mathrm{T}}$ is the amplitude of N_d dipoles at time k, $\mathbf{n}_k$ is the measurement noise, and $\mathbf{A}_k$ is the $M \times N_d$ gain matrix. The jth

column of this matrix is given by $\mathbf{a}_k^j = \mathbf{F}(\mathbf{r}_k^j)\mathbf{q}_k^j$, where the lead field $\mathbf{F}(\mathbf{r})$ is represented by an $M \times 3$ matrix and is a nonlinear function of the dipole location $\mathbf{r}$ [15].

In order to show the spatial sparsity of the EEG signal, we constrained the dipoles to G grids in a 3-D Cartesian coordinate enclosing the brain volume. For the fixed Cartesian coordinate y, the grid is shown in Figure 6.8. Given this constraint, the M-channel EEG signal can be approximated as,

$$\mathbf{z}_k = \Omega\,\theta_k + \mathbf{n}_k \tag{6.8}$$

where $\Omega = [\mathbf{F}(\mathbf{r}_1)\ \mathbf{F}(\mathbf{r}_2)\ \ldots\ \mathbf{F}(\mathbf{r}_G)]$ is an $M \times 3G$ matrix, $\mathbf{F}(\mathbf{r}_i)$ is the $M \times 3$ lead field of the ith grid located at $\mathbf{r}_i$, θ_k is a $3G \times 1$ vector with elements $[\mathbf{m}_k^1\ \ldots\ \mathbf{m}_k^G]^{\mathrm{T}}$, and $\mathbf{m}_k^i = [m_k^{i,x}\ m_k^{i,y}\ m_k^{i,z}]$ represents the moment of a dipole concentrated on the ith grid at time k. Since there are N_d dipoles, only N_d grids have non-zero moments. As a result, at most $3 \times N_d$ elements in θ_k are non-zero, where $N_d \ll G$. Thus, according to CS theory, the EEG signal $\mathbf{z}_k$ is $3 \times N_d$-sparse in the spatial domain.

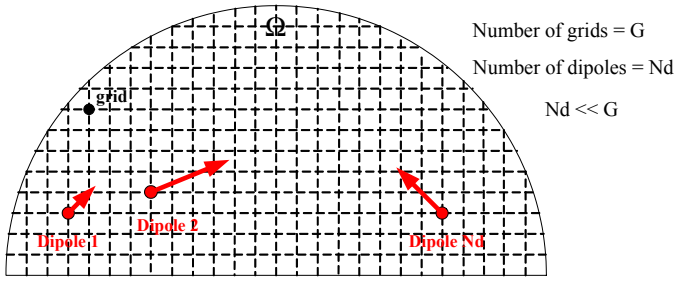


Figure 6.8: Sparse current dipole signals.

We define the L-dimensional compressed measurement as

$$\mathbf{y}_k = \Phi\mathbf{z}_k + \varepsilon_k = \Phi\Omega\theta_k + \varepsilon_k \tag{6.9}$$

where $L \ll M$, Φ is the $L \times M$ projection matrix uncorrelated with Ω, and ε is the projected noise. CS theory ensures that the compressed measurement $\mathbf{y}_k$ has all the information of $\mathbf{z}_k$ and θ_k. As a result, we can use $\mathbf{y}_k$ as our new measurement in tracking neural dipole sources.

6.2.3 Spatial compressed particle filtering

Using the new compressed measurement vector, we can re-formulate the state-space model for the neural dipole source tracking problem as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \nu_k, \tag{6.10}$$

$$\mathbf{y}_k = \Phi h(\mathbf{x}_k) + \mathbf{n}_k, \tag{6.11}$$

where $\mathbf{x}_k$ is the state of a dipole at time k including its 3-D location $\mathbf{r}_k$ and 3-D moment $\mathbf{m}_k$, $h(\cdot)$ is the nonlinear measurement function obtained from Equation (6.7), ν_k is the modeling error, and $\mathbf{n}_k$ is a combination of measurement noise and projection noise ε_k. Based on this model, we use a particle filtering (PF) algorithm to track the state $\mathbf{x}$ of the neural dipole source; the spatial compressed measurements $\mathbf{y}$ is used as the PF input. The steps of the proposed algorithm are as follows.

Initialization: The samples $\{\mathbf{x}_0^{(\ell)}\}_{\ell=1}^N$ are drawn from the initial density $p(\mathbf{x}_0)$, where N is the number of particles and $p(\mathbf{x}_0)$ is a uniform distribution of samples in the sphere head model. The weights are assigned to be initially equal, $w_0^{(\ell)} = 1/N$.

Prediction: For $k = 1,\ldots,K$, the particles $\mathbf{x}_k^{(\ell)}$ are drawn from the probability density $p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(\ell)})$. Then the prior density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ can be approximated by $p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \sum_{\ell=1}^N w_{k-1}^{(\ell)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(\ell)})$.

Update: The weights $w_k^{(\ell)}$ are updated based on the new compressed measurement y_k using $w_k^{(\ell)} \propto w_{k-1}^{(\ell)} p(\mathbf{y}_k|\mathbf{x}_k^{(\ell)})$. We also normalize the weights $w_k^{(\ell)} = w_k^{(\ell)} / \sum_{\ell=1}^N w_k^{(\ell)}$.

Resampling: The particles are resampled based on their weights to obtain $\{\mathbf{x}_k^{(\ell)}, w_k^{(\ell)}\}_{\ell=1}^N$. The posterior density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ can be approximated by

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \sum_{\ell=1}^{N} w_k^{(\ell)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(\ell)}).$$

Thus, the measurement $\mathbf{y}_k$ is the spatial compressed version of the original measurement $\mathbf{z}_k$, $\mathbf{y}_k = \Phi\mathbf{z}_k + \mathbf{n}_k$. Typically, the elements of the projection matrix are random variables drawn from independent and identically distributed Gaussian distributions or Bernoulli distributions to make sure that Φ is uncorrelated with Ω. After the spatial compressive sensing, the dimension of the measurement reduces from M to L, where $L \ll M$. As a result, the amount of EEG data that is needed to be stored can be reduced from $M \times K$ to $L \times K$ bytes. The likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$ is

now an L-dimensional Gaussian density function instead of an M-dimensional density function. This results in a significant reduction in the computational complexity of the signal processing operations that are a function of the number of EEG channels L.

This algorithm can be extended to multiple dipole source tracking problem by using the multiple particle filtering framework [50]. In our simulation results, we use one PF for each dipole source. The jth sub-PF uses the same prediction and resampling steps as the ones just just described. However, the weight update step is modified to $w_{j,k}^{(\ell)} \propto w_{j,k-1}^{(\ell)} \, p(\mathbf{y}_k | \mathbf{x}_{j,k}^{(\ell)}, \tilde{\mathbf{x}}_{-j,k})$, where $\tilde{\mathbf{x}}_{-j,k} = [\tilde{\mathbf{x}}_{1,k}^{\mathrm{T}} \ \cdots \ \tilde{\mathbf{x}}_{j-1,k}^{\mathrm{T}} \ \tilde{\mathbf{x}}_{j+1,k}^{\mathrm{T}} \ \cdots \ \tilde{\mathbf{x}}_{N_d,k}^{\mathrm{T}}]^{\mathrm{T}}$ are the predicted values of all the states, excluding $\tilde{\mathbf{x}}_{j,k}$ and $\tilde{\mathbf{x}}_{j,k} = \sum_{\ell=1}^{N} w_{j,k-1}^{(\ell)} \mathbf{x}_{j,k}^{(\ell)}$.

6.2.4 Algorithm performance results for compressive sensing

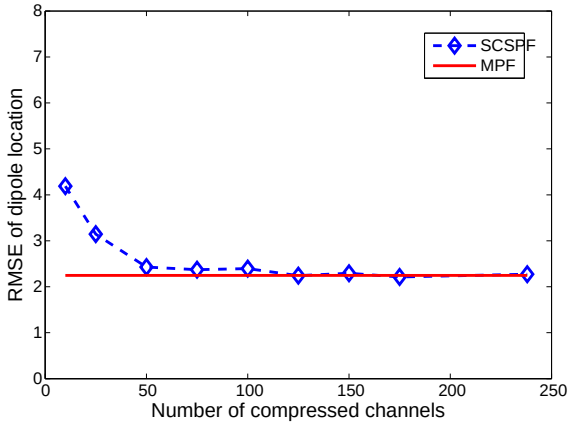
We compared our proposed spatial compressive particle filtering (SCPF) with the multiple particle filtering (MPF) [50] using simulated and real EEG data. For our simulations, we draw each element of the projection matrix Φ from a zero-mean, additive white Gaussian random process with unit variance elements.

Simulated data results. The synthetic data was created by inserting three current dipoles into a sphere head model and calculating the resulting EEG signals using Equation (6.7) with Gaussian noise. The three dipoles are localized at $V1$ (1.11, 5.34, 4.98), $V5_R$ (4.36, 3.68, 4.44) and $V5_L$ (3.37, 4.85, 4.81) from a previous study. For this simulation, we used 1,000 particles for each dipole. The particles are initially uniformly distributed in the hemisphere representing the brain with a radius of 85 mm. The dipole evolution model in Equation (6.11) is a random walk with Gaussian transition kernel $p_{k+1|k}(\mathbf{p}_{k+1}|\mathbf{p}_k) = \mathcal{N}(\mathbf{p}_k, \sigma_p)$ and $p_{k+1|k}(\mathbf{m}_{k+1}|\mathbf{m}_k) = \mathcal{N}(\mathbf{m}_k, \sigma_m)$, with σ_p=1 cm and σ_m=2 nA, where $\mathbf{p}$ and $\mathbf{m}$ are the 3-D dipole location and moment vector, respectively.

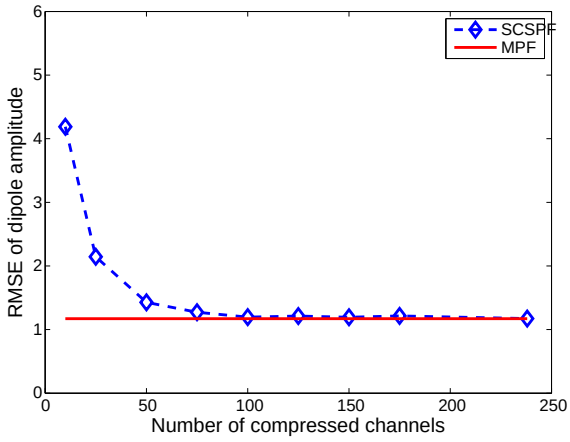
In Figure 6.9, we show the estimation results, averaged over 50 independent runs, in terms of RMSE for the proposed SCPF using 1,000 particles. For comparison, we also show the estimation results for the MPF [50] using all 238 EEG measurements and also 1,000 particles. From Figure 6.9, we can see that as the number of compressed sensors increases, the RMSE decreases as expected. When the number of compressed sensors is smaller then 50, there is a significant performance degradation in term of RMSE. When the number of compressed channels is greater than

50, there is no noticeable improvement in the RMSE. Thus, we choose the number of compressed sensors to be 50 and the corresponding projection matrix Φ to be 50×238.

For this case when the number of sensors is 50, the location and amplitude estimation RMSE for SCPF and MPF are compared in Table 6.5. Assuming that there are 100 time steps in this simulations, the amount of EEG data needed to be stored has been reduced from $238 \times 100 \approx 24$ kbytes to $50 \times 100 = 5$ kbytes. Table 6.6 shows a comparison of the two competing methods in terms of computing operations. Note that SCPF has reduced number of additions and multiplications.



(a) RMSE, dipole location



(b) RMSE, dipole amplitude

Figure 6.9: RMSE estimation performance comparison.

Real EEG data results. Next we apply the proposed SCPF to real EEG data that is publicly available [96]. In this experiment, the subject's screen showed 5 empty boxes arranged horizontally above the screen center. At the screen center, there was a plus sign which was used as the fixation

Table 6.5: RMSE comparison of SCPF and MPF for synthetic data (For SCPF, we used 1,000 particles and 50 sensors; for MPF, we used 1,000 particles and 238 sensors).

Algorithm	Location	Amplitude
MPF	2.25 mm	1.17 nA
SCPF	2.42 mm	1.43 nA

Table 6.6: Operations per particle for SCPF and MPF (For SCPF, we used 1,000 particles and 50 sensors; for MPF, we used 1,000 particles and 238 sensors).

Algorithm	Block	+	$\times$	$\div$	$\sqrt{}$	exp
MPF		323	341	1	2	1
SCPF	CS	73	91	0	0	0
	PF	69	74	1	2	1

point throughout the experiment. When the task began, a white disc would appear in any one of the boxes for 100 ms. The location of the box is called the attended location. The subject was instructed to press the response button whenever the disc appeared at the attended location. Data were collected from 238 scalp, neck, face and eye locations using the Biosemi Active Two system. Data is referenced with respect to the electrode located in the right mastoid.

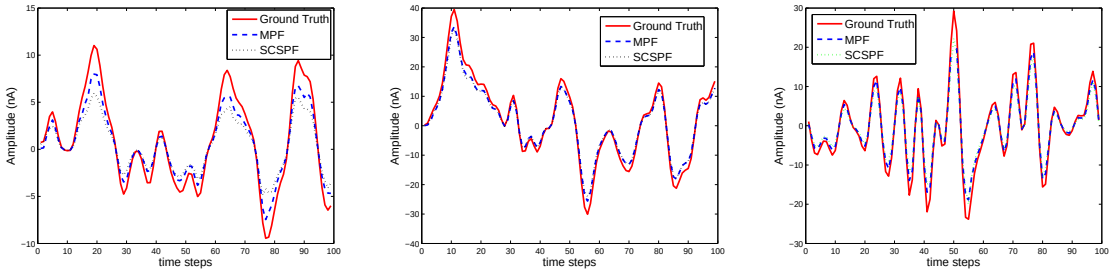


Figure 6.10: Estimation of the amplitudes of three dipoles.

First, we preprocess the real EEG data based on the methods in [87]. The preprocessing steps include bandpass filtering, event extraction and independent component analysis. Next we apply the SCPF and MPF on the preprocessed EEG data. Here, we choose the projection matrix Φ as an 50×238 matrix with elements drawn from standard Gaussian distribution. We estimate the locations and amplitudes of three dipoles and for each dipole we use 1,000 particles for both the SCPF and MPF algorithms. We also compare the estimation results with the dipole fitting method in [87]. Since the true locations and amplitudes of the dipoles are unknown in the real

data case, we set the dipole fitting results in [87] as the *ground truth*. The estimation results are shown in Figure 6.10 and Figure 6.11 and the RMSE is shown in Table 6.7. From Figure 6.10 and Figure 6.11, we can see that the amplitudes and locations of the dipoles estimated by SCPF and MPF match with the results given in [87]. Table 6.7 indicates that the SCPF can give comparable RMSE performance with the MPF. However, since for SCPF we use only 50 compressed channel measurements instead of the 238 original channel measurements, the amount of EEG data needed to be stored and processed is significantly reduced. For one hour of EEG data acquisition with 1 kHz sampling rate, the amount of EEG data needed to be stored has been reduced from 857 MB to 180 MB.

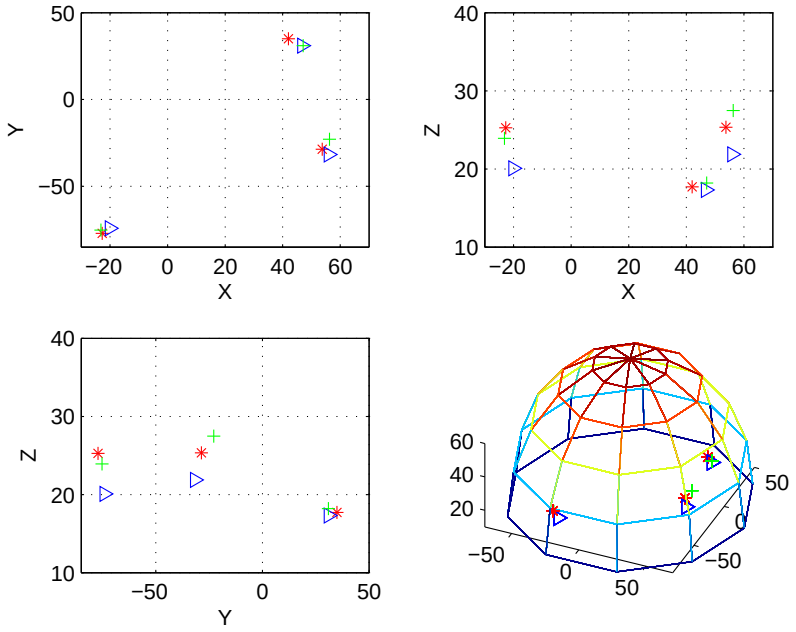


Figure 6.11: Estimation of the locations of three dipoles.

Table 6.7: RMSE comparison of SCPF and MPF for real data (For SCPF, we used 1,000 particles and 50 sensors; for MPF, we used 1,000 particles and 238 sensors).

Algorithm	Location	Amplitude
MPF	5.17 mm	2.63 nA
SCPF	5.62 mm	2.85 nA

Chapter 7

Conclusions and Future Work

In this study, we developed advanced statistical signal processing algorithms to automatically track dynamic neural activities and designed efficient hardware architectures to implement the tracking systems in real-time.

7.1 Summary

First, we proposed an efficient parallel architecture for implementing the particle filtering algorithm. This architecture achieves both high speed and accurate estimation performance by using the independent Metropolis-Hastings sampler along with the parallel PF implementation (PPF-IMH). The proposed method was implemented on a Xilinx Virtex-5 FPGA platform. For a one-dimensional problem with 1,000 particles, the PPF-IMH architecture with four processing elements uses less than 5% of a Virtex-5 FPGA's resources and takes 5.85 μs for one iteration.

Next, we used the PPF-IMH algorithm in waveform agile sensing to adaptively increase dynamic state estimation performance. Simulations demonstrated that the estimation performance is significantly improved and the processing speed is faster due to the PF parallelization.

We also applied the PPF-IMH for neural activity tracking and proposed the use of multiple particle filtering (MPF) for tracking multiple neural dipole sources. First, we considered the case when the number of dipole sources was known. We demonstrated the tracking performance of the proposed MPF system using both synthetic and real data. This method achieves better tracking performance in terms of MSE using the same number of particles as the SIR PF. When using lower number of particles for each sub-PF (compared to SIR PF), it achieves the same MSE but with a highly reduced computational complexity. The proposed method was also implemented on the Xilinx Virtex-5 FPGA platform. The processing time for one iteration using 8,000 particles was shown to be only 31.15 μs.

When the number of neural dipole sources is unknown, we used PF-PHDF to estimate both the number of dipoles and their parameters. We demonstrated its performance using numerical simulations for both synthetic and real EEG data. The proposed method achieves good performance in terms of RMSE using significantly fewer number of particles compared to existing approaches.

We also presented a window based processing method and a threshold based eigenvalue distilling algorithm to enable real-time processing. The proposed method was implemented on the Xilinx Virtex-5 FPGA platform. The processing time for a window with 100 samples using 3,200 particles for a system with 3 dipoles was shown to be only 5.1 ms. Thus, this implementation is also capable of real-time processing of systems with larger number of dipoles and/or larger number of samples per second.

Finally, we considered sensor scheduling and compressive sensing methods to reduce the number of required sensors of an EEG system. By using the proposed sensor scheduling method, the system achieved comparable tracking performance in terms of RMSE (6.4 mm compared to 6.3 mm) with only half the number of sensors (15 out of 32 sensors). Then, we proved the sparsity of EEG signal and integrated compressive sensing technique with PF to track the dipole parameters. The RSME tracking results of the proposed algorithm are comparable with those of conventional methods. While the number of required EEG channels is reduced from 238 to 50. Thus, both methods significantly reduced the number of active sensors and hence the power consumption of the EEG system without increasing the tracking error.

7.2 Future work

There are still a number of issues that need to be addressed.

1. In Chapter 5, we assumed that in a short time window the location and orientation of dipole sources are fixed. Under this assumption, the measurement vector $\mathbf{z}_k$ can be represented as $\mathbf{z}_k = \mathbf{A}\mathbf{s}_k$ and ICA can be used to decompose the mixed measurement. Actually, this assumption can be removed if other blind signal separation (BSS) algorithms, such as Bayesian Nonlinear Independent Component Analysis (BICA) in [97] are used.

2. The state evolution model (Equation (4.4)) for neural dipole sources is a random walk model. It is a general model which includes little information of how the neural dipoles evolve with time. However, in order to further improve the tracking performance, a more accurate state evolution model is needed. This model is likely to be different for each type of disease. So one method could be to use a set of training data to find the trajectory of the moving dipole and build a state evolution model for a specific brain disease based on the trajectory.

3. In Chapter 6, we proposed two sensor scheduling methods with different cost functions: (a) minimization of the predicted mean squared error (PMSE) in estimation, and (b) maximization of the signal-to-noise ratio (SNR) of the measurements. In this study, we only showed the software and hardware evaluation results for the second method. When the number of sensors is large, the computational complexity of minimization of the PMSE is very high. Convex optimization [91] can be used to solve this minimization problem with low computational complexity. Thus sensor scheduling with minimization PMSE can also be implemented to reduce the number of sensors required.

Reference

[1] S. Baillet, J. C. Mosher, and R. M. Leahy, "Electromagnetic brain mapping," *IEEE Signal Processing Magazine*, vol. 18, pp. 14–30, 2001.

[2] J. S. Ebersole, "Functional neuroimaging with EEG source models to localize epileptogenic foci noninvasively," *University of Chicago Hospitals' Clinical Comment.*

[3] A. L. Benabid, "Deep brain stimulation for Parkinsons disease," *Current Opinion in Neurobiology*, vol. 13, pp. 767–791, 2003.

[4] A. J. Casson, S. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable EEG: what is it, why is it needed and what does it entail," in *Proceeding of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2008, pp. 5867–5870.

[5] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, pp. 767–791, 2002.

[6] "Augmented cognition international society," http://www.augmentedcognition.org/.

[7] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuuttila, and O. V. Lounasmaa, "Magnetoencephalography–theory, instrumentation and applications to nonivasive studies of the working human brain," *Reviews of Modern Physics*, vol. 65, pp. 413–497, 1993.

[8] J. C. Mosher, R. M. Leahy, and P. S. Lewis, "EEG and MEG: Forward solutions for inverse methods," *IEEE Transactions in Biomedical Engineering*, vol. 46, pp. 245–259, 1999.

[9] E. Somersalo, "The inverse problem of magnetoencephalography: Source localization and the shape of a ball," *SIAM News*, vol. 40, no. 2, March 2007.

[10] P. Nunez, *Electric Fields of the Brain*. New York: Oxford, 1981.

[11] A. J. Casson and E. Rodriguez-Villegas, "Data reduction techniques to facilitate wireless and long term AEEG epilepsy monitoring," in *Proceeding of the 3rd International IEEE/EMBS Conference on Neural Engineering*, 2007, pp. 298–301.

[12] A. J. Casson, D. C. Yates, S. Patel, and E. Rodriguez-Villegas, "Algorithm for AEEG data selection leading to wireless and long term epilepsy monitoring," in *Proceeding of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 2456–2459.

[13] D. Yates, E. Lopez-Morillo, R. G. Carvajal, J. Ramirez-Angulo, and E. Rodriguez-Villegas, "A low-voltage low-power front-end for wearable EEG systems," in *Proceeding of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 5282–5285.

[14] J. C. Mosher, P. S. Lewis, and R. M. Leahy, "Multiple dipole modeling and localization from spatio-temporal MEG data," *IEEE Transactions on Biomedical Engineering*, vol. 39, pp. 541–557, June 1992.

[15] J. C. Mosher and R. M. Leahy, "Source localization using recursively applied and projected (RAP) MUSIC," *IEEE Transactions on Signal Processing*, vol. 47, pp. 332–340, 1999.

[16] B. D. Van Veen and K. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Magazine*, vol. 5, pp. 4–24, 1988.

[17] E. Somersalo, A. Voutilainen, and J. P. Kaipio, "Non-stationary magnetoencephalography by Bayesian filtering of dipole models," *Inverse Problems*, vol. 19, pp. 1047–1063, 2003.

[18] A. Galka, O. Yamashita, T. Ozaki, R. Biscay, and P. Valde, "A solution to the dynamical inverse problem of EEG generation using spatiotemporal Kalman filtering," *Inverse Problems*, pp. 435–453, 2004.

[19] J. Antelis and J.Minguez, "Dynamic solution to the EEG source localization problem using Kalman filters and particle filters," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, 2009, pp. 77–80.

[20] A. Sorrentino, L. Parkkonen, and M. Piana, "Particle filters: A new method for reconstructing multiple current dipoles from MEG data," in *International Congress Series*, vol. 1300, 2007, pp. 173–176.

[21] H. R. Mohseni, E. L. Wilding, and S. Sanei, "Sequential Monte Carlo techniques for EEG dipole placing and tracking," in *Sensor Array and Multichannel Signal Processing Workshop*, July 2008, pp. 95–98.

[22] C. Campi, A. Pascarella, A. Sorrentino, and M. Piana, "A Rao-Blackwellized particle filter for magnetoencephalography," *Inverse Problems and Imaging*, vol. 24, p. 15, 2008.

[23] H. R. Mohseni, F. Ghaderi, E. L. Wilding, and S. Sanei, "A beamforming particle filter for EEG dipole source localization," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 337–340.

[24] A. Pascarella, A. Sorrentino, C. Campi, and M. Piana, "Particle filtering, beamforming and multiple signal classification for the analysis of MEG time series: A comparison of algorithms," *Inverse Problems and Imaging*, vol. 4, pp. 169–190, February 2010.

[25] A. Sorrentino, L. Parkkonen, A. Pascarella, C. Campi, and M. Piana, "Dynamical MEG source modeling with multi-target Bayesian filtering," *Human Brain Mapping*, vol. 30, pp. 1911–1921, June 2009.

[26] A. Pascarella and A. Sorrentino, "Statistical approaches to the inverse problem," *Magnetoencephalography. InTech*, November 2011.

[27] C. Campi, A. Pascarella, A. Sorrentino, and M. Piana, "Highly Automated Dipole EStimation (HADES)," *Computational Intelligence and Neuroscience*, 2011, article ID 982185.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME*, vol. 82, pp. 35–45, March 1960.

[29] N. J. Gordon, D. J. Salmon, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings in Radar and Signal Processing*, vol. 140, 1992, pp. 107–113.

[30] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[31] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[32] B. Ristic, S. Arulampalam, and N. J. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Norwood, MA: Artech House Publishers, 2004.

[33] A. Athalye, M. Bolić, S. Hong, and P. M. Djurić, "Architectures and memory schemes for sampling and resampling in particle filters," in *Digital Signal Processing Workshop*, vol. 1, August 2004, pp. 92–96.

[34] ——, "Generic hardware architectures for sampling and resampling in particle filters," *EURASIP Journal of Applied Signal Processing*, vol. 17, pp. 2888–2902, 2005.

[35] M. Bolić, "Architectures for efficient implementation of particle filters," Ph.D. dissertation, State University of New York at Stony Brook, 2004.

[36] M. Bolić, P. M. Djurić, and S. Hong, "Resampling algorithms for particle filters: A computational complexity perspective," *EURASIP Journal of Applied Signal Processing*, vol. 15, pp. 2267–2277, 2004.

[37] ——, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Process.*, vol. 7, pp. 2442–2450, July 2005.

[38] S. Hong, Z. Shi, J. Chen, and K. Chen, "Compact resampling algorithm and hardware architecture for particle filters," in *IEEE International Conference on Communications, Circuits and Systems*, vol. 2, 2008, pp. 886–890.

[39] ——, "Novel roughening algorithm and hardware architecture for bearings-only tracking using particle filter," *Journal of Electromagnetic Waves and Applications*, vol. 22, pp. 411–422, 2008.

[40] ——, "A low-power memory-efficient resampling architecture for particle filters," *Circuits, Systems and Signal Processing*, vol. 29, pp. 155–167, 2010.

[41] S. Hong, M. Bolić, and P. M. Djurić, "An efficient fixed-point implementation of residual resampling scheme for high-speed particle filters," *IEEE Signal Processing Letters*, vol. 11, pp. 482–485, May 2004.

[42] C. Berzuini, N. G. Best, W. R. Gilks, and C. Larizza, "Dynamic conditional independence models and Markov chain Monte Carlo methods," *Journal of the American Statistical Association*, vol. 92, pp. 1403–1412, 1997.

[43] A. C. Sankaranarayanan, R. Chellappa, and A. Srivastava, "Algorithmic and architectural design methodology for particle filters in hardware," in *IEEE International Conference on Computer Design*, October 2005, pp. 275–280.

[44] A. C. Sankaranarayanan, A. Srivastava, and R. Chellappa, "Algorithmic and architectural optimizations for computationally efficient particle filtering," *IEEE Transactions on Image Processing*, vol. 17, pp. 737–748, May 2008.

[45] B. B. Manjunath, A. S. Williams, C. Chakrabarti, and A. Papandreou-Suppappola, "Efficient mapping of advanced signal processing algorithms on multi-processor architectures," in *IEEE Workshop on Signal Processing Systems*, October 2008, pp. 269–274.

[46] D. J. Kershaw and R. J. Evans, "Optimal waveform selection for tracking systems," *IEEE Transactions on Information Theory*, vol. 40, pp. 1536–1550, September 1994.

[47] S. P. Sira, A. Papandreou-Suppappola, and D. Morrell, "Dynamic configuration of time-varying waveforms for agile sensing and tracking in clutter," *IEEE Transactions on Signal Processing*, vol. 55, pp. 3207–3217, July 2007.

[48] ——, *Advances in Waveform-Agile Sensing for Tracking*. San Rafael, CA, Morgan & Claypool Publishers, 2009.

[49] P. M. Djuric, T. Lu, and M. F. Bugallo, "Multiple particle filering," in *IEEE International Confernece on Acoustics, Speech, and Signal Processing*, June 2007, pp. 1181–1184.

[50] L. Miao, J. J. Zhang, C. Chakrabarti, and A. Papandreou-Suppappola, "Multiple sensor sequential tracking of neural activity with FPGA implementation," in *Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA*, November 2010, pp. 369–373.

[51] R. Mahler, "Multi-target Bayes filtering via first-order multi-target moments," *IEEE Transactions on Aerospace and Electronic System*, vol. 39, pp. 1152–1178, 2003.

[52] L. Miao, J. J. Zhang, C. Chakrabarti, A. Papandreou-Suppappola, and N. Kovvali, "Real-time closed-loop tracking of an unknown number of neural sources using probability hypothesis density particle filtering," in *IEEE Workshop on Signal Processing Systems*, October 2011, pp. 367–372.

[53] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation*, vol. 9, pp. 1483–1492, 1997.

[54] B. N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking," in *International Conference of Information Fusion*, vol. 2, 2003, pp. 792–799.

[55] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 804–816, 2010.

[56] Y. M. Chi, S. R. Deiss, and G. Cauwenberghs, "Non-contact low power EEG/ECG electrode for high density wearable biopotential sensor networks," in *Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, 2009, pp. 246–250.

[57] W. Zhou, N. Kovvali, A. Papandreou-Suppappola, and A. Chattopadhyay, "Sensor optimization for progressive damage diagnosis in complex structures," in *Proceedings of SPIE*, 2010, pp. 282–285.

[58] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "On the use of binary programming for sensor scheduling," *IEEE Transactions on Signal Processing*, vol. 55, pp. 2826–2839, 2007.

[59] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[60] E. Wang, J. Silva, and L. Carin, "Compressive particle filtering for target tracking," in *Statistical Signal Processing Workshop*, Sept. 2009, pp. 233–236.

[61] V. E. *Beneš*, "Exact nite-dimensional lters with certain diffusion non linear drift," *Stochastics*, vol. 5, p. 6592, 1981.

[62] F. E. Daum, "Beyond kalman filters: practical design of nonlinear filters," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, 1995, pp. 252–262.

[63] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 2004.

[64] S. Hong, Z. Shi, and K. Chen, "Easy-hardware-implementation MMPF for maneuvering target tracking: Algorithm and architecture," *Journal of Signal Processing Systems*, vol. 61, pp. 1–5, November 2009.

[65] R. van der Merwe, A. Doucet, J. F. G. de Freitas, and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems*, vol. 13, Dec. 2000, pp. 584–590.

[66] H. V. Trees, *Detection, Estimation, and Modulation Theory, Optimum Array Processing*. Wiley-Interscience, 2004.

[67] P. Tichavsky, C. H. Muravchik, and A. Nehorai, "Posterior Cramer-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1386–1396, 1998.

[68] J. Zhang, B. Manjunath, G. Maalouli, A. Papandreou-Suppappola, and D. Morrell, "Dynamic waveform design for target tracking using MIMO radar," in *Asilomar Conference on Signals, Systems and Computers*, November 2008, pp. 31–35.

[69] J. Zhang, Q. Ding, S. Kay, A. Papandreou-Suppappola, and M. Rangaswamy, "Agile multimodal tracking with dependent measurements," in *Asilomar Conference on Signals, Systems and Computers*, November 2010, pp. 1653–1657.

[70] S. P. Sira, A. Papandreou-Suppappola, and D. Morrell, "Time-varying waveform selection and configuration for agile sensors in tracking applications," in *IEEE International Conference of Acoustics, Speech and Signal Processing*, vol. 5, March 2005, pp. 881–884.

[71] M. A. Woodbury, "Inverting modified matrices," *Statistical Research Group, Princeton University, Princeton, NJ*, vol. 42, p. 4, 1950.

[72] K. Uutela, M. Hämäläinen, and R. Salmelin, "Global optimization in the localization of neuromagnetic sources," *IEEE Transactions on Biomedical Engineering*, vol. 45, pp. 716–723, 1998.

[73] J. Gross, J. Kujala, M. Hamalainen, L. Timmermann, A. Schnitzler, and R. Salmelin, "Dynamic imaging of coherent sources: Studying neural interactions in the human brain," *National Academy of Sciences*, vol. 98, pp. 694–699, January 2001.

[74] "FieldTrip Matlab software toolbox for MEG and EEG analysis," http://fieldtrip.fcdonders.nl.

[75] L. Miao, J. J. Zhang, C. Chakrabarti, and A. Papandreou-Suppappola, "Algorithm and parallel implementation of particle filtering and its use in waveform-agile sensing," *Journal of Signal Processing Systems*, vol. 65, pp. 211–227, November 2011.

[76] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes, Volume II: General theory and Structure*, 2nd ed. Springer-Verlag, 1997.

[77] M. Tobias and A. D. Lanterman, "Probability hypothesis density-based multi-target tracking with bistatic range and Doppler observations," *IEEE Radar, Sonar and Navigation*, vol. 152, pp. 195–205, 2005.

[78] D. E. Clark and J. Bell, "Bayesian multiple target tracking in forward scan sonar images using the PHD filter." *IEEE Radar, Sonar and Navigation*, vol. 152, pp. 327–334, 2005.

[79] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[80] C. Jutten and J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal processing*, vol. 24, no. 1, pp. 1–10, 1991.

[81] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[82] J. Hurri, A. Hyvärinen, J. Karhunen, and E. Oja, "Image feature extraction using independent component analysis," in *In Proc. NORSIG'96*, 1996.

[83] N. Delfosse and P. Loubaton, "Adaptive blind separation of independent sources: a deflation approach," *Signal processing*, vol. 45, pp. 59–83, 1995.

[84] K. Shindo, A. Ikeda, T. Musha, K. Terada, H. Fukuyama, W. Taki, J. Kimura, and H. Shibasaki, "Clinical usefulness of the dipole tracing method for localizing interictal spikes in partial epilepsy," *Epilepsia*, vol. 39, pp. 371–379, 1998.

[85] T. Chen, W. Liu, and L. Chen, "VLSI architecture of leading eigenvector generation for on-chip principal component analysis spike sorting system," in *International Conference of the IEEE Engineering in Medicine and Biology Society*, 2008, pp. 3192–3195.

[86] S. Makeig, M. Westerfield, T. P. Jung, S. Enghoff, J. Townsend, E. Courchesne, and T. J. Sejnowski, "Dynamic brain sources of visual evoked responses," *Science*, vol. 295, pp. 690–694, 2002.

[87] "EEG lab: Open source Matlab toolbox for EEG analysis," http://sccn.ucsd.edu/eeglab/.

[88] S. Fujio, H. Shiomi, and Y. Okamura, "Acceleration of FPGA-based ICA processor for real-time processing," in *International Symposium on Antennas and Propagation*, July 2010, pp. 1–4.

[89] J. Cao, N. Murata, S. Amari, A. Cichocki, and T. Takeda, "Independent component analysis for unaveraged single-trial MEG data decomposition and single-dipole source localization," *Neurocomputing*, vol. 49, no. 1-4, pp. 255–277, 2002.

[90] W. Welch, "Branch-and-bound search for experimental designs based on D-optimality and other criteria," *Technometrics*, vol. 24, pp. 41–48, 1982.

[91] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transaction on Signal Processing*, vol. 57, pp. 451–462, 2009.

[92] C. Chakrabarti, "High sample rate array architecture for median filters," *IEEE Transaction on Signal Processing*, vol. 57, pp. 707–712, Mar. 1994.

[93] L. Miao, S. Michael, N. Kovvali, C. Chakrabarti, and A. Papandreou-Suppappola, "Multi-source neural activity estimation and sensor scheduling: Algorithms and hardware implementation," *Journal of Signal Processing Systems*, vol. 1, pp. 1–18, 2012.

[94] L. Miao, J. Zhang, C. Chakrabarti, and A. Papandreou-Suppappola, "Efficient Bayesian tracking of multiple sources of neural activity: Algorithms and real-time FPGA implementation," *IEEE Transactions on Signal Processing*, vol. 61, pp. 633–647, 2013.

[95] S. Baillet, J. Mosher, and R. Leahy, "Electromagnetic brain mapping," *IEEE Signal Processing Magazine*, vol. 18, no. 6, pp. 14–30, Nov. 2001.

[96] M. Westerfield, T. P. Jung, S. Enghoff, J. Townsend, E. Courchesne, and T. J. Sejnowski, "Dynamic brain sources of visual evoked responses," *Science*, vol. 295, pp. 690–694, Jan. 2002.

[97] H. Lappalainen and A. Honkela, "Bayesian non-linear independent component analysis by multi-layer perceptrons," in *Advances in independent component analysis*, 2000, pp. 93–121.