

Turbo Bayesian Compressed Sensing

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Depeng Yang

August 2011

© by Depeng Yang, 2011
All Rights Reserved.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisors, Dr. Gregory Peterson and Dr. Husheng Li, for years of invaluable guidance, encouragement, and financial support. Their brilliant insights have always inspired me throughout my research. During my worst time when I was going to quit my PhD study, their help made me finally finish my PhD study. Without them the dissertation could never has been completed.

I would also like to thank Dr. Hairong Qi for her help and fruitful discussions. I would also say thanks to Dr. Seddik Djouadi, whose help and advice helped my research a lot. I would thank Dr. Yulong Xing for his patient guidance on the mathematics. His deep knowledge and suggestions made me further study and research the compressed sensing theory. I would like to express my gratitude to Prof. Fathy for his valuable advice and guidance on the UWB system.

Friendship has also made a huge part of my doctoral study. I extend my heartiest thanks to you all, Zhenghao Zhang, JunKyu Lee, Getao Liang, Kun Zheng, Rukun Mao, Shuang Gao, for all the joy we have shared together.

The same gratitude goes to my parents and my wife, Liang Xu, as she keeps on encouraging me to fulfill my ambition, despite all the difficulties we faced together. Her love and forgiveness have been giving me strength and courage to pursue my dreams.

Abstract

Compressed sensing (CS) theory specifies a new signal acquisition approach, potentially allowing the acquisition of signals at a much lower data rate than the Nyquist sampling rate. In CS, the signal is not directly acquired but reconstructed from a few measurements. One of the key problems in CS is how to recover the original signal from measurements in the presence of noise. This dissertation addresses signal reconstruction problems in CS. First, a feedback structure and signal recovery algorithm, *orthogonal pruning pursuit* (OPP), is proposed to exploit the prior knowledge to reconstruct the signal in the noise-free situation. To handle the noise, a noise-aware signal reconstruction algorithm based on Bayesian Compressed Sensing (BCS) is developed. Moreover, a novel Turbo Bayesian Compressed Sensing (TBCS) algorithm is developed for joint signal reconstruction by exploiting both spatial and temporal redundancy. Then, the TBCS algorithm is applied to a UWB positioning system for achieving mm-accuracy with low sampling rate ADCs. Finally, hardware implementation of BCS signal reconstruction on FPGAs and GPUs is investigated. Implementation on GPUs and FPGAs of parallel Cholesky decomposition, which is a key component of BCS, is explored. Simulation results on software and hardware have demonstrated that OPP and TBCS outperform previous approaches, with UWB positioning accuracy improved by 12.8x. The accelerated computation helps enable real-time application of this work.

Contents

List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Introduction to Compressed Sensing	1
1.2 General Problem Statements	4
1.2.1 UWB Application Problem	4
1.2.2 Computation Acceleration Problem	11
2 Related Work	13
2.1 Related Work	13
3 Hard Decision Algorithm: Feedback Orthogonal Pruning Pursuit Algorithm	22
3.1 Background and Motivation	22
3.2 System Model and Problem Formulation	23
3.2.1 Problem Formulation	23
3.2.2 Feedback Structure for UWB Receiver	24
3.3 Orthogonal Pruning Pursuit Algorithm	25
3.3.1 Necessary Lemmas	25
3.3.2 Algorithm Description	27
3.4 Numerical Simulation	29

4	Soft Decision Algorithm: Joint Bayesian Compressed Sensing Algorithm	33
4.1	Background and Motivation	33
4.2	Problem Formulation	36
4.3	Mutual Information Transfer in Bayesian Compressed Sensing	39
4.3.1	Bayesian Compressed Sensing Framework	40
4.3.2	Yielding Prior Information	41
4.3.3	Incorporating Prior Information into BCS	46
4.4	Incremental Optimization	47
4.5	Case Study: Turbo Bayesian Compressed Sensing for UWB Systems	50
4.5.1	UWB System Model	50
4.5.2	STTBCS: Structure and Algorithm	53
4.6	Simulation Results	54
4.6.1	Spike Signal	55
4.6.2	UWB Signal	56
5	Applications of Joint BCS Algorithm	66
5.1	Background and Motivation	66
5.2	UWB Positioning System Model	69
5.2.1	UWB Signal Model for Positioning	69
5.2.2	Compressed Sensing UWB Signal Acquisition	70
5.3	Space-Time Bayesian Compressed Sensing	72
5.3.1	Bayesian Compressed Sensing	73
5.3.2	Transfer Prior Information	73
5.3.3	Space-Time Structure	76
5.4	STBCS-TDOA Joint Signal Processing	77
5.4.1	Estimate the Pulse Peak Location	78
5.4.2	Fusing Prior Information	80
5.4.3	TDOA Algorithm	81

5.4.4	Joint STBCS-TDOA Algorithm	83
5.5	Simulation Results	85
5.5.1	UWB Signal Reconstruction	85
5.5.2	Joint STBCS-TDOA Performance	89
5.5.3	3D Positioning Performance	91
6	Hardware Implementation of Bayesian compressed Sensing	96
6.1	Background and Introduction	96
6.2	Cholesky Decomposition on GPUs	98
6.3	Cholesky Decomposition on FPGAs	100
6.3.1	Pipelined Triangular Linear Solver on FPGAs	103
6.4	Compressed Sensing on FPGAs and GPUs	108
6.4.1	Signal Reconstruction Algorithm	108
6.4.2	Compressed Sensing on FPGAs	110
6.4.3	Compressed Sensing on GPUs	112
6.5	Performance Comparison of FPGA and GPU	114
6.6	Hardware Results	117
6.6.1	Customizable Precision on FPGAs	118
6.6.2	Cholesky Decomposition Performance	119
6.6.3	Compressed Sensing Performance	122
7	Conclusions and Future Work	126
7.1	Contributions	126
7.2	Conclusions	129
7.3	Future Work	130
	Bibliography	132
A	Proof of Equations	144
A.1	Proof of Lemma 3.0.2	144

A.2	Proof of Eq. (4.10) and (4.11)	146
A.3	Derivation of Eq. (4.16)	148
A.4	Derivation of Eq. (4.20)	149
A.5	Derivations about Eq.(4.27) and Eq.(4.28)	151
A.6	Prior Information	153
A.7	Proof of Proposition 1	157
A.8	Maximum Posterior	159
Vita		161

List of Tables

2.1	CS reconstruction algorithm	15
4.1	Notation list	39
6.1	Performance and hardware resources usage on FPGA	119

List of Figures

1.1	Illustration of compressed sensing theory	2
1.2	Example of the received UWB pulses	5
1.3	Receiver structure	6
1.4	Illustration of the inner structure of distributed amplifier	7
1.5	Illustration of positioning system	9
1.6	Historical redundancy in UWB signals	9
1.7	Spatial redundancy in UWB signals	10
2.1	Tree of CS theory	16
3.1	The feedback structure for the UWB receiver.	23
3.2	Diagram of algorithm	28
3.3	Reconstructed UWB echo signals using OMP, BP, and OPP. Time is in ns; Length N=512, measurements M=55 and the sparsity K=17. Signals are contaminated by the noise with SNR=12.95dB. The reconstruction percentages are respectively: (a)Original UWB echo signal without noise. (b)OMP: 0% (c)BP: 20.48% (d)OPP: 65.83%	30
3.4	Probability of reconstruction vs. measurements without noise	30
3.5	Probability of reconstruction vs. measurements in the presence of noise	31
3.6	BER vs. SNR using different amount of measurements.	32
4.1	A typical UWB system with one transmitter and several receivers . . .	35

4.2	Block diagram of decentralized turbo Bayesian compressed sensing . . .	37
4.3	The distribution $P(s_j^i \lambda_j^i)$	43
4.4	Block diagram of space-time turbo Bayesian compressed sensing . . .	53
4.5	Spike signal with 20 nonzero elements in random locations	57
4.6	Reconstructed spike signal using MBCS with 75% similarity	57
4.7	Reconstructed spike signal using TBCS with 75% similarity	58
4.8	Performance gain in each iteration with 25% similarity	58
4.9	Performance gain in each iteration with 50% similarity	59
4.10	Performance gain in each iteration with 75% similarity	59
4.11	Performance of original BCS and TBCS	61
4.12	Performance comparison at different similarity levels without noise. .	62
4.13	Performance comparison at different similarity levels in the presence of noise	63
4.14	BER performance using different algorithms	64
5.1	Compressed sensing UWB positioning system	72
5.2	Space-time structure for the STBCS algorithm	77
5.3	Parallel STBCS-TDOA algorithm	84
5.4	Comparison of the reconstructed UWB echo signals, U_1 , U_2 , and U_3 using BCS in (a), (c), and (e) and STBCS in (b), (d), and (f).	87
5.5	Compression ratio versus reconstruction percentage	89
5.6	Noise versus reconstruction percentage	90
5.7	Positioning performance of the OMP, BCS, and STBCS-TDOA algorithm	91
5.8	CS-based positioning system performance	92
6.1	Cholesky decomposition on GPUs. (a), (b), (c), and (d) depict the computational procedure sequentially, where computation in (c) for updating A_{22} using matrix-matrix multiplication is dominant.	99
6.2	Cholesky decomposition computational procedure on FPGAs	103
6.3	Computational procedure for solving triangular equations on FPGAs	105

6.4	Architecture for solving triangular linear systems	106
6.5	Compressed sensing on FPGAs	111
6.6	Compressed sensing on GPUs	113
6.7	Accuracy using customized mantissa bits on FPGAs	118
6.8	Performance comparison for Cholesky decomposition in single precision	120
6.9	Performance comparison for Cholesky decomposition in double precision	120
6.10	Computation cycles for Cholesky decomposition in single precision . .	121
6.11	Computation cycles for Cholesky decomposition in double precision .	121
6.12	Hardware compressed sensing speedup of GPU, FPGA and MAGMA in single precision with respect to CPU execution time	123
6.13	Hardware compressed sensing speedup of GPU, FPGA and MAGMA in double precision with respect to CPU execution time	123

Chapter 1

Introduction

This dissertation addresses signal acquisition problems. We consider the research in the state-of-the-art compressed sensing theory. The research problem and general problem statement will be detailed. Previous work and papers will be investigated. We will discuss the research achievable goals and possible approaches, leading into a description of the dissertation problem.

1.1 Introduction to Compressed Sensing

Compressed sensing (CS) theory [11] [27] is blooming in recent years. Essentially, in CS theory, the original signal is not directly acquired but reconstructed based on the measurements obtained from projecting the signal onto a random distributed matrix. It is well known that most natural signals are sparse, i.e., in a certain transform domain, most elements are zeros or have very small amplitudes. Taking advantage of such sparsity, various CS signal reconstruction algorithms are developed to recover the original signal from a few observations and measurements.

In Fig. 1.1 we first consider a real-valued, N -dimensional, discrete-time signal x , which can be viewed as an $N \times 1$ column vector in R^N with elements $x[n], n = 1, 2, \dots, N$. The signal vector x is the original signal. Any signal in R^N can be represented with respect to some basis set. Using the $N \times N$ basis matrix $\Psi =$

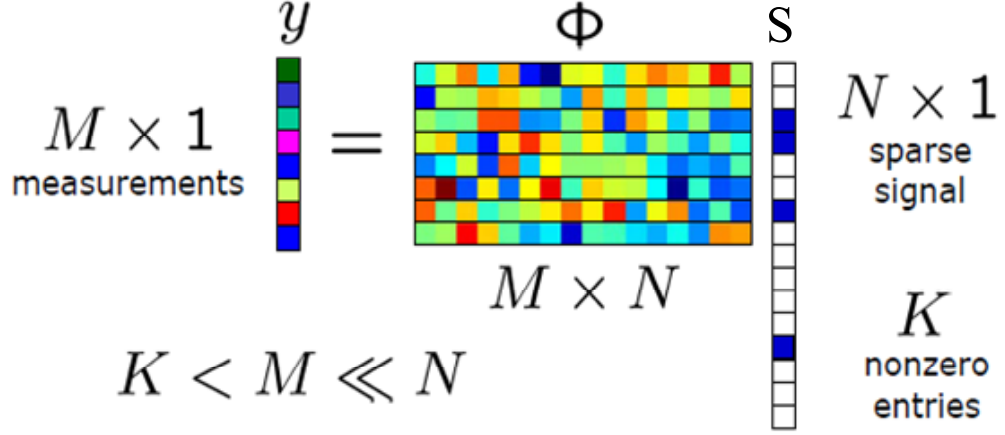


Figure 1.1: Illustration of compressed sensing theory

$[\psi_1|\psi_2|\dots|\psi_N]$ with the vectors ψ_i as columns, a signal x can be expressed as,

$$x = \Psi S, \quad (1.1)$$

where S is the $N \times 1$ column vector. Clearly, x and S are equivalent representations of the signal, with x in the time or space domain and S in the Ψ domain. The signal x is K -sparse if it is a linear combination of only K basis vectors; that is, only K of the coefficients are nonzero and $(N - K)$ are zero. When $K \ll N$, the signal x is compressible if the representation has just a few large coefficients and many small coefficients. Then we consider a general linear measurement process that computes $M < N$ inner products,

$$y = \Phi x = \Phi \Psi S = \Theta S, \quad (1.2)$$

where the measurement vector y is an $M \times 1$ vector. Note that the Gaussian measurement matrix Θ is incoherent. Otherwise, the redundant elements should be removed. More specifically, an $M \times N$ Independent and Identically Distributed (iid) Gaussian matrix can be shown to have the Restricted Isometry Property (RIP) with high probability if $M \geq cK \log(N/K)$, with c a small constant. Therefore,

K sparse and compressible signals of length N can be recovered from only $M \geq cK \log(N/K) \ll N$ random Gaussian measurements [11].

CS theory develops algorithms to reconstruct the original signal vector x or, equivalently, its sparse coefficient vector S , given the M -dimensional measurement vector y , and the random measurement matrix Θ . There are several methods which can be utilized to obtain an estimate of the signal. For example, based on optimization theory, the minimum ℓ_1 optimization is given by

$$\hat{S} = \arg \min \|S\|_1, \quad (1.3)$$

$$\text{subject to } y = \Theta S. \quad (1.4)$$

where a family of convex optimization algorithms, such as Basis Pursuit (BP), and simplex methods, can be utilized to reconstruct the signal by minimizing the ℓ_1 norm.

For the minimum ℓ_2 ,

$$\hat{S} = \arg \min \|S\|_0, \quad (1.5)$$

$$\text{subject to } y = \Theta S. \quad (1.6)$$

where a family of pursuit algorithms, such as Matching Pursuit (MP), and Orthogonal Matching Pursuit (OMP) approaches, can be utilized to reconstruct the signal by minimizing the ℓ_2 norm. Other algorithms are also developed based on machine learning theory[38], Bayesian network [59], and wireless coding/decoding algorithms [24].

Essentially CS theory reconstructs the N -dimensional signal vector indirectly from the limited M measurements. CS theory can be applied to several applications to lower down the sampling rate of direct signal acquisitions. One application is an Ultra-Wideband (UWB) system.

1.2 General Problem Statements

1.2.1 UWB Application Problem

Ultra-wideband (UWB) techniques bring a paradigm shift to the field of wireless communications which can realize giga bit/second data rates for many applications. For example, the UWB technique has been utilized for realizing high rate communication systems, sensor networks, and millimeter precision positioning systems due to advantages of high timing resolution, low power consumption, and simple transmitter architecture [69]. The UWB system utilizes a short-range, high-bandwidth pulse without carrier frequency for communication, positioning, and radar imaging systems. However, the challenge is the acquisition of the high-resolution, ultra-short duration pulses, such as timing information in nano- or subnano-second levels. The extremely high sampling rate requirement for acquiring short impulses (on the order of picoseconds) is one of the biggest challenges in UWB receivers. Fortunately, the emerging theory of compressed sensing (CS)[11] offers an effective approach to achieve sub-Nyquist sampling rate acquisition for UWB channel estimation and UWB communication. In this approach, original signals are linearly mixed to yield indirect measurements. From the measurements obtained by using multiple low sampling rate analog to digital converters (ADC), the original UWB signal is reconstructed using CS algorithms [27].

It is well known that the UWB echo signals are inherently sparse in the time domain, as shown in Fig. 1.2. Propagating through multi-path UWB channels, the UWB echo signals, $s_i(t)$, received at the i -th base station in the continuous time domain are given by

$$s_i(t) = \sum_{l=1}^L a_l p'(t - t_{il}) \quad (1.7)$$

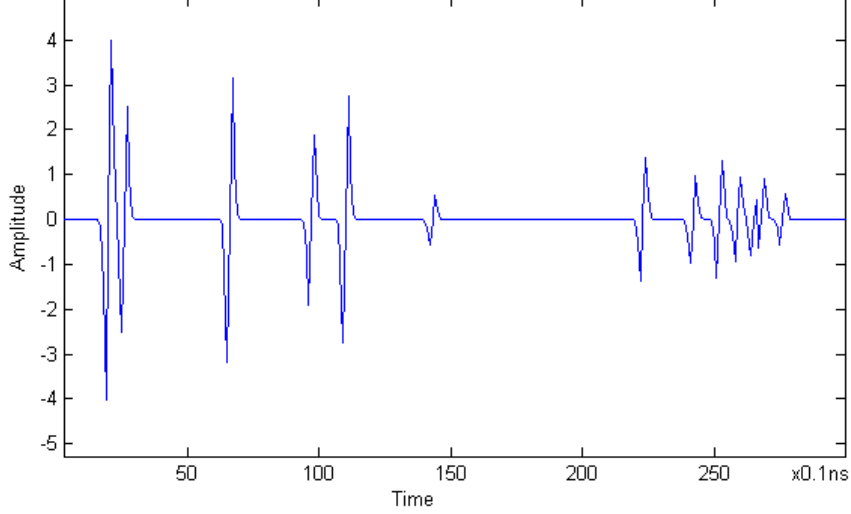


Figure 1.2: Example of the received UWB pulses

where we denote by $p(t)$ the transmitted Gaussian pulse, $p'(t)$ the received distorted pulse resulting from frequency-dependent propagation channels, L the number of resolvable propagation paths, a_l the amplitude attenuation of the signal along the l -th path, and t_{il} the time delay of the l -th path at the i -th base station. In a line-of-sight (LOS) environment, the first arriving pulse always moves through the shortest propagation path, which has the largest amplitude. In the digital domain, we mark the high definition UWB signal vector as $\mathbf{u}$, which is given by

$$\mathbf{u} = \sum_{i=1}^L a_i p'(t - t_{il}) \quad (1.8)$$

To obtain the ultra-high resolution UWB pulses using low sampling rate ADCs, we can apply CS theory into the UWB system[20][19]. However, in the realistic UWB system where multiple UWB echo signals must be reconstructed, it is necessary to develop a novel CS algorithm for joint signal reconstruction.

Receiver Structure

The bottleneck of the UWB system is the ultra-high sampling rate ADC at the receivers since the sampling rate for capturing extremely short UWB pulses (on the

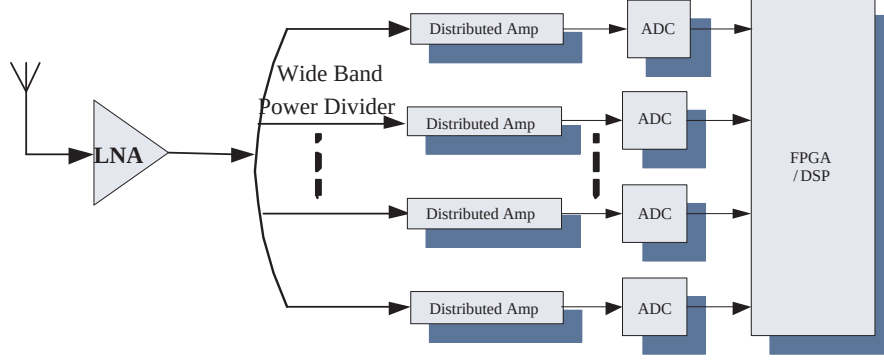


Figure 1.3: Receiver structure

order of picoseconds) of the ADC for real-time signal processing is typically much higher than that of commercially available ADCs. Therefore, we seek help from the fact that UWB pulses are sparse in the time domain (an example is given in Fig. 1.2) and employ the powerful technique of compressed sensing to alleviate the difficulty of high rate sampling for UWB pulses. Note that we consider only one base station in this subsection, thus ignoring the indices of base stations.

The receiver structure is illustrated in Fig. 1.3. Note that the key component in the proposed receiver is the distributed amplifier to yield the measurements for the following ADCs and Field Programmable Gate Array (FPGA) devices.

The distributed amplifier (DA) can be utilized for compressing the analog signal. The DA, first invented in 1936 and studied for many decades, is a microwave circuit for wideband microwave signal amplifiers, which can also be used as microwave FIR filter, and transversal filters. As shown in Fig. 1.4, a DA consists of multiple repeated and identical taps, each containing a section of micro-strip input and output transmission line, and the gain cell. This periodic architecture forms a tunable transmission line [25][75]. The output of a DA at time t is given by

$$y(t) = \sum_{j=1}^m c_n u(t - j\tau) \quad (1.9)$$

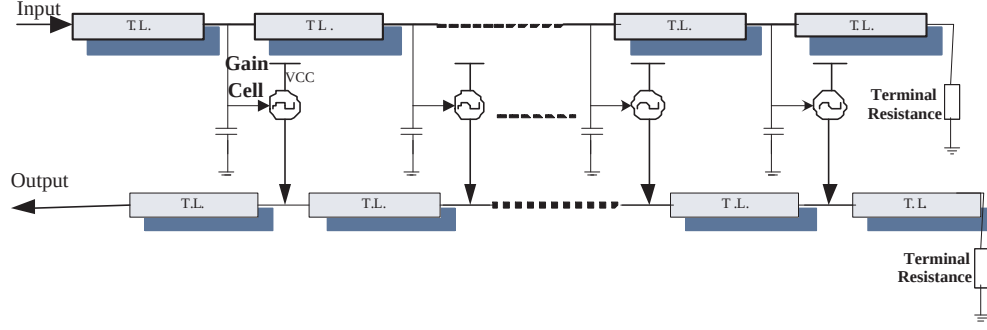


Figure 1.4: Illustration of the inner structure of distributed amplifier

where m is the number of taps in a DA, $\{c_n\}$ are attenuation coefficients of different gain cells which can be set randomly, u is the input signal, and τ is the physically fixed time delay of each section of transmission line, which determines the time resolution of the UWB signal. Note that the gain c_n can be arbitrarily set for different gain cells and different distributed amplifiers. The outputs of the distributed amplifiers are then fed into an array of M ADCs.

The DA is suitable for analog CS processing in the following three aspects:

- The artificial transmission line in DA supports wideband signal propagation and the characteristic impedance of such transmission line changes little over several GHz. Thus, the waveform of the propagating UWB signal, $S(t)$, is maintained.
- The time delays, determined by the length of each section of transmission line along which the signal propagates, can easily achieve 100ps, or even less based on different substrates and technologies without changing the structure of DA[75].
- The co-efficient of the gain cell can be either predefined or reconfigurable to meet the requirements of measurement matrix. In this paper, these gain coefficients are randomly distributed numbers conforming the requirement of the CS measurement matrix.

Note that, in practice, the nonlinearity of the gain cell in different frequency bands will introduce the polynomial multiplication terms. However, under the small signal

model and low gain in our case, the nonlinear effect can be largely alleviated. Also we can use low noise figure components to alleviate noise interference. For simplicity of discussion, we ignore the nonlinearity effect.

Then if we can utilize several DAs to compose the measurement matrix, we mark the coefficient as c_n^i , where i represents the i -th row DA. Therefore, the compressed UWB signal is represented as:

$$\mathbf{Y} = \mathbf{\Phi}\mathbf{u} + \boldsymbol{\epsilon}. \quad (1.10)$$

Note that $\mathbf{Y}$ is an M -vector. A is an $M \times N$ matrix given by

$$\mathbf{\Phi} = \begin{pmatrix} c_1^1 & c_2^1 & \cdots & c_N^1 \\ c_1^2 & c_2^2 & \cdots & c_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^M & c_2^M & \cdots & c_N^M \end{pmatrix}. \quad (1.11)$$

$\mathbf{x}$ is an N -vector given by

$$\mathbf{x} = (x(t - \tau), x(t - 2\tau), \dots, x(t - N\tau))^T, \quad (1.12)$$

where x is noise-free received signal. For simplicity, we write $\mathbf{x}$ as

$$\mathbf{x} = (x_1, x_2, \dots, x_N). \quad (1.13)$$

$\boldsymbol{\epsilon}$ is an additive white Gaussian noise (AWGN) in the distributed amplifier with known mean and variance. It is easy to see that the original signal $\mathbf{x}$ is compressed into the lower dimensional vector $\mathbf{y}$ and $\mathbf{A}$ plays the role of measurement matrix in compressed sensing.

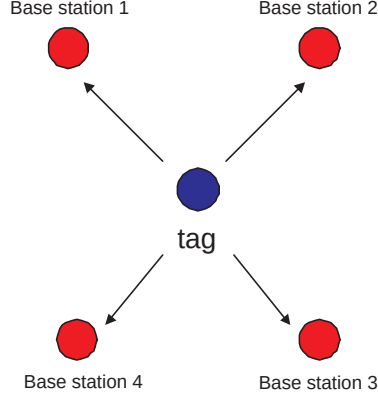


Figure 1.5: Illustration of positioning system

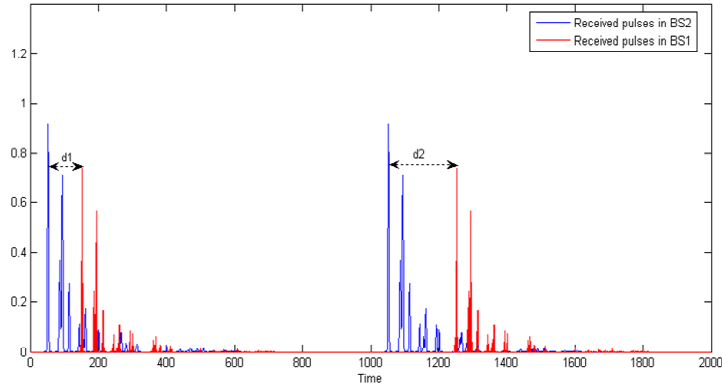


Figure 1.6: Historical redundancy in UWB signals

We then use the Filed Programmable Gate Array (FPGA) to reconstruct $\mathbf{u}$ from $\mathbf{y}$, thus obtaining the UWB pulse arrival times and combining the data from all base stations to estimate the current location.

UWB positioning system

The UWB positioning system basically contains a transmitter (tag) and multiple receivers (base stations or BS), which is illustrated in Fig. 1.5. The task is to estimate the location of the tag via the signal transmitted from the tag and received at the base stations.

We can find some obvious characteristics or redundancies in the unknown UWB signal vector:

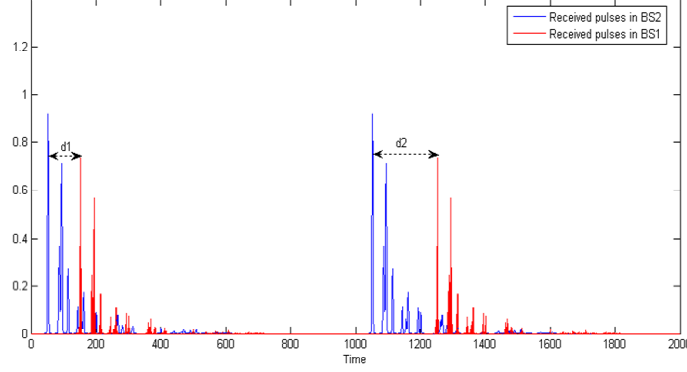


Figure 1.7: Spatial redundancy in UWB signals

- Temporal redundancy: if we assume that the tag does not move rapidly, which is valid for most positioning systems, the arrival time does not change rapidly, so we can roughly predict the locations of UWB pulses. This redundancy is demonstrated in Fig. 1.6.
- Spatial redundancy: the UWB pulses are intercepted at multiple base stations, thus incurring redundancy; therefore, we can combine the information of multiple base stations to exploit the redundancy. This redundancy is displayed in Fig. 1.7.

Based on the above analysis, we can apply the CS theory into the UWB system. However, when applying CS to practical UWB systems, we need to address the following problems:

- Can we reduce the number of measurements required for reliable signal reconstruction? The CS reconstruction algorithm requires a certain number of measurements to recover the original signals. However, in practical systems the measurements realized in the analog domain require expensive hardware resources. Reducing the number of measurements leads to decrease the sampling rate, which means to alleviate the sampling problem for UWB systems.
- How do we alleviate the complexity of CS algorithms for fast signal reconstruction? Most CS algorithms are computationally expensive. A simple CS

reconstruction algorithm is in pressing need for high speed UWB communication systems. In addition, it is also important for the algorithm to combat noise and interference.

- How can we take advantages of temporal and spatial redundancy to improve performance of CS signal reconstruction? By exploiting and integrating the redundant information into the CS signal reconstruction algorithm, can we reduce the number of measurements? Can we improve the performance of combating the noise and alleviate the complexity of the CS algorithm?

1.2.2 Computation Acceleration Problem

CS theory and signal reconstruction algorithms are usually computationally expensive. In particular, most CS algorithms are associated with intensive matrix operations, such as matrix-matrix multiplication, Cholesky decomposition, and matrix inversion. With the growth of the matrix size, computational costs will greatly increase, leading to unacceptably high execution time. This hampers applying the CS algorithm into time sensitive applications, such as real-time video pattern recognition, online model regression, and fast signal reconstruction. Therefore, we seek to obtain a speedup of the CS computation by utilizing special purpose accelerators such as Graphical Processing Units (GPUs) and FPGAs.

The least square method is widely used in various computational applications such as scientific computation, signal processing, and image processing. In particular, the least square problem is the key part in Orthogonal Matching Pursuit (OMP), a powerful signal reconstruction algorithm in compressed sensing theory [1]. However, the computational cost of obtaining least square solutions is very heavy. By optimizing the algorithm and exploiting parallelism through designing dedicated hardware on FPGAs, we achieve a significant computational speedup.

Cholesky decomposition with $O(N^3)$ complexity is dominant in the computation. The traditional Cholesky decomposition needs to utilize the division, multiplication

and square root operation. however, the square root operation involves very long latency, which hampers its use in an efficient pipeline architecture. We need to rearrange the decomposition procedure to improve performance.

In this chapter, we introduced CS theory and associated general problems. In the next chapter, the related work in CS theory will be further investigated.

Chapter 2

Related Work

2.1 Related Work

Compressed Sensing (CS) theory was introduced in [27][11][56] around 2006. Research interest in CS theory focuses on the projection matrix and reconstruction algorithm. A sparse signal is linearly mixed by a projection matrix to yield measurements. From a certain number of measurements, the sparse signal can be exactly recovered. In order to successfully reconstruct the original sparse signal, the Restricted Isometry Property (RIP) of the matrix is studied in [40][32][55]. Associated with the matrix, another research topic is to study the minimum number of measurements for successful reconstruction according to various algorithms. Fig. 2.1 shows a taxonomic tree of CS signal reconstruction algorithms.

At the beginning, for a single frame signal reconstruction, mathematicians and statisticians developed CS reconstruction algorithms based on linear programming techniques [35][43]. The first way to reconstruct the signal is based on hard decision and optimization theory. The most famous algorithm is basis pursuit (BP) using the simplex algorithm by minimizing the norm ℓ_1 . At the expense of slightly more measurements, iterative greedy algorithms have been developed to recover the signal x from the measurements y . Examples include the iterative Orthogonal Matching

Pursuit (OMP) [23], matching pursuit (MP)[60], and modified matching pursuit [31] algorithms. Another modified Matching pursuit algorithm is Stage-wise Orthogonal Matching Pursuit (StOMP) [33]. StOMP is an enhanced version of OMP where multiple coefficients are resolved at each stage of the greedy algorithm, as opposed to only one in the case of OMP.

Our work on hard decision CS algorithms is to develop a feedback structure to introduce the prior information based on optimization theory. Our proposed new algorithm is similar to orthogonal matching pursuit (OMP)[35] and other advanced l_0 norm optimization algorithms, e.g. the Stagewise OMP (StOMP)[10], regularized OMP (ROMP)[9], and sparsity adaptive matching pursuit(SAMP) [64] algorithms. Compared with those traditional CS signal reconstruction algorithms, our work can reduce the complexity. The proposed algorithm can significantly improve the performance by exploiting redundant information.

The second way to develop CS reconstruction algorithms is based on statistical signal processing and information theory. In [24][28], authors discuss and compare CS theory and channel coding/decoding theory, and find that the CS reconstruction algorithm can be realized and analyzed by using channel decoding algorithms. Then the Turbo, Low Density Parity Coding (LDPC), and belief propagation algorithms are also applied into CS theory to recover the sparse signal by recasting CS theory as a channel coding/decoding procedure [24]. Research on applying channel decoding algorithms into CS theory is a trend.

The third way to reconstruct the signal is based on machine learning. For example, the relevance vector machine algorithm [38][39][29] for sparse Bayesian machine learning has been applied to CS, called Bayesian CS theory [59], which has a good capability of combating noise. BCS algorithm has a statistical hierarchy structure which can be utilized to integrate and introduce useful information for joint signal reconstruction.

The CS signal reconstruction algorithms based on machine learning and statistical theory are soft decision based CS signal reconstruction algorithms. Those algorithms

Table 2.1: CS reconstruction algorithm

Algorithms	Setting	Complexity	Minimum M
ℓ_1 Optimization	noise-free	Ω^3	$K \log(1 + \frac{N}{K})$
ℓ_1 Regulation	noisy	NK^2	$K \log(1 + \frac{N}{K})$
OMP	noise-free	$2K \log K^2$	NK^2
StOMP	noise-free	$K \log N$	$K \log N$
Chain Pursuit	noise-free	$K \log^2 N$	$N \log^2 N$
BCS	noisy	Ω^3	$K \log N$
LDPC	noisy	$K \log N$	$K \log N$
Our work	noisy	$< K \log N$	$\ll K \log N$

have good performance on combating the noise. More importantly, those algorithms are easy to modify for integrating the prior information for particular applications.

For joint signal reconstruction, the proposed research is to exploit the prior information in sparse signals for joint signal reconstruction based on sparse Bayesian machine learning theory. Related research about joint CS signal reconstruction has been developed recently. Distributed compressed sensing (DCS) [22][67] studies joint sparsity and joint signal reconstruction. Simultaneous OMP (SOMP) [23] [60] [31] for simultaneous signal reconstruction is developed by extending the traditional OMP algorithm. Serial OMP [46] studies the time sequence signal reconstructions. The joint sparse recovery algorithm [26] is developed associated with the basis pursuit (BP) algorithm. Those algorithms focus on either temporal or spatial joint signal reconstructions. They are developed by extending convex optimization and linear programming algorithms but ignore the impact of possible noise in the measurements.

Other work on sparse signal reconstruction is based on a statistical Bayesian framework. In [28], the authors developed a sparse signal reconstruction algorithm based on a belief propagation framework for the signal reconstruction. The information is exchanged among different elements in the signal vector in a way similar to the decoding of low density parity check (LDPC) codes. In [24], the LDPC coding/decoding algorithm has been extended for real number CS signal reconstruction. Other Bayesian CS algorithms also have been developed

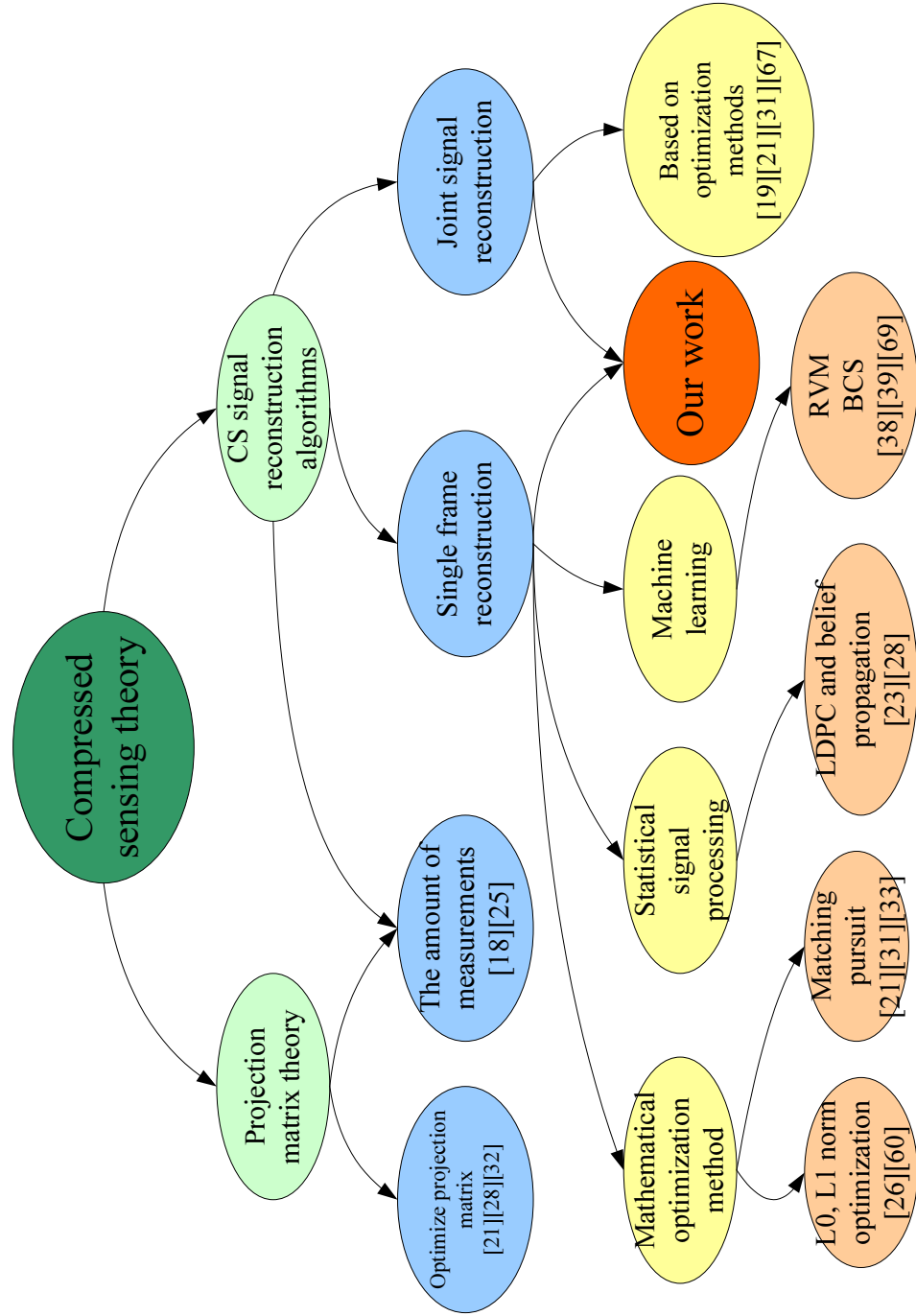


Figure 2.1: Tree of CS theory

in [63][29][53]. However, all these algorithms are designed only for a single procedure signal reconstruction and are not applied for multiple simultaneous signal reconstructions. In [19], the redundancies of UWB signals are incorporated into the framework of Bayesian Compressed Sensing (BCS)[38] [59] and have achieved good performance. However, only a heuristic approach is proposed to utilize the redundancy in [19].

More work related with our research for the joint sparse signal reconstruction includes [61]. The author proposed Multi-task Bayesian Compressive Sensing (MBCS) for simultaneous joint signal reconstruction by sharing the same set of hyperparameters for the signals. The mutual information is directly transferred over multiple simultaneous signal reconstruction tasks. In [74], the authors proposed to utilize the Dirichlet prior distribution. The mechanism of sharing mutual information is similar to the MBCS [61]. This sharing scheme is effective and straightforward. For signals with high similarity, it has a much better performance than the original BCS algorithm. However, for a low level of similarity, prior information may adversely hamper the signal reconstruction, resulting in much worse performance than the original BCS. In the situation where there exist lots of low-similarity signals, this disadvantage could be unacceptable.

Compared with the previous research work on joint signal reconstruction, we propose a novel and flexible Turbo Bayesian Compressed Sensing (TBCS) algorithm for sparse signal reconstruction through exploiting and integrating spatial and temporal redundancies in multiple signal reconstruction procedures performed in parallel, in serial, or both. Moreover, we apply our proposed TBCS algorithm to an UWB positioning system for high positioning accuracy.

In order to alleviate the sampling problem, some previous work proposed methods for acquiring high resolution UWB signals. In our previous work [8][13][45], we utilized a sequential sub-sampler to acquire ultra-high timing resolution UWB pulses by using a low sampling rate ADC. The basic idea of the sequential sampling sub-sampler is to compose one period of UWB signal with many repetitive periods of original signals.

The received UWB signal is assumed to be repeated at the known and fixed pulse repetition frequency. Then the sampling clock at the receiver has an offset compared with the pulse repetition frequency. After many periods of sampling, one complete period of high resolution UWB signals is obtained. More details can be found in [13]. However, such a sequential sub-sampler suffers from many problems. One is that the speed is very slow so that this approach is not suitable for tracking fast moving targets. The positioning accuracy is seriously decreased when the target is moving. The asynchronous clocks between the tag and the receivers prevents the UWB positioning system from achieving a higher accuracy. In order to achieve an ultra-high positioning accuracy in high speed, we are seeking to find an approach for real-time ultra-high speed sampling with low sampling rate ADCs.

In addition, the CS theory[27] has been applied to UWB systems for acquiring high resolution pulses below the Nyquist sampling rate [30] [76][20][12][19][15]. In a certain time interval, suppose that an N -dimensional high resolution signal vector containing the original UWB signals is compressed by a projection matrix to output an M -dimensional measurement vector. From the measurements which are obtained by ADCs, the signal vector can be reconstructed. It is well known that UWB echo signals are naturally sparse in the time domain. In our experiments, we observed that nonzero pulse signals are less than 10% of received signals in one pulse repetition period [13][45][8]. Therefore, the N -dimensional high resolution signal vector can be reconstructed from M -dimensional measurements. In a certain time interval, the number M determines the sampling rate of the ADC. The acquisition of the limited measurements, instead of the original pulse, dramatically decreases the sampling rate of the ADC, which makes it possible to obtain ultra-high resolution UWB pulses at a very low sampling rate. Related work for applying CS theory to UWB systems can be found [30][76], which focus on the acquisition of high resolution UWB pulses using CS algorithms. In [7], CS based pulse signal reconstruction has been studied, but only for single frame pulse reconstruction. In [66], CS theory is applied to positioning systems; however, it is performed in the frequency domain. Also note that existing joint CS

reconstruction algorithms for general cases include distributed compressed sensing (DCS) [22][67] for studying joint sparsity and joint signal reconstruction, simultaneous OMP (SOMP) [23][60] [31][33] for simultaneous signal reconstruction, serial OMP [46] for time sequence signal reconstruction, and joint sparse signal recovery using the basis pursuit (BP) algorithm [26]. However, these algorithms do not consider the redundancies in both space and time and consider only noise-free cases.

The proposed TBCS algorithm is still computationally expensive. The computational cost is a bottleneck in time-sensitive applications[21]. High performance parallel computing for computational acceleration is needed for fast CS signal reconstruction. One of the most computationally expensive steps in CS signal reconstruction algorithms is Cholesky decomposition[5], so we desire to accelerate Cholesky decomposition and CS signal reconstruction by designing a dedicated hardware/software module to exploit parallelism.

Besides CS signal reconstruction in signal/image processing, Cholesky decomposition is one of most widely used decomposition algorithms in various applications and fields[69], such as in solving least square problems, linear regression, and least square fitting in mathematics, machine learning, and economics. Cholesky decomposition has $O(\frac{1}{3}n^3)$ complexity, which is computationally expensive. Even when the matrix size n is small, the computation time is too expensive in many applications, such as real-time CS signal reconstruction[20][21][19]. In many other fields, such as computational chemistry[57], there exist a demand of computing Cholesky decomposition for many small matrices. Modern massively parallel devices, such as FPGAs and GPUs, can accelerate Cholesky decomposition by exploiting parallelism.

Related work on Cholesky decomposition and compressed sensing either focus on FPGA implementation or GPU/CPU programming. In [52], the authors reorder the Cholesky decomposition computation using a 3-dimensional structure for FPGA implementation. However, the standard Cholesky decomposition procedure is complicated and the associated root square operation requires lots of resources and long latency. In [34][6], Cholesky decomposition is implemented on a GPU using

vectorized inner and outer methods based on subroutines provided by CUDA Basic Linear Algebra Subprograms (CUBLAS) and Basic Linear Algebra Subprograms (BLAS). In [65], authors analyze performance trade-offs for various matrix-vector operations and matrix decomposition on GPUs. For CS signal reconstruction, others discuss GPU implementation in [42][58] for the matching pursuit algorithm based on sub-routines from CUBLAS. Orthogonal matching pursuit (OMP) and CS signal reconstruction are also implemented on FPGAs in [4] and [20]. We proposed an optimized GPU and FPGA implementation for a family of CS signal reconstruction algorithms. In our previous work [17][18][16][14], we propose an iterative Cholesky decomposition for solving linear square problems and the Relevance Vector Machine (RVM) algorithm. In this thesis, we expand our work for CS signal reconstruction algorithms on both FPGAs and GPUs.

Other well known parallel computing libraries include Linear Algebra PACKage (LAPACK) [37] for CPUs and Matrix Algebra for GPGPU and multi-core architectures (MAGMA)[41] for a hybrid system with CPU and GPU, which provides plenty of optimized Level 1-3 basic matrix operations. LAPACK can be utilized for single or multi-core high performance parallel computing. MAGMA can exploit both CPU and GPU capabilities for best performance. For example, for Cholesky decomposition, MAGMA can assign parallel tasks to GPUs, such as matrix-matrix multiplication, and perform serial tasks on CPUs, such as solving triangular equations using forward/backward substitution, to exploit the strengths of these different platforms[41]. LAPACK and MAGMA have excellent performance for large matrix decomposition; however, LAPACK and MAGMA may not be a good choice for small matrix Cholesky decomposition due to overhead and data streaming costs. Moreover, the target matrix in the iterative CS signal reconstruction algorithm is actually augmented each iteration so that Cholesky decomposition can be performed based on previous results for good performance. However, LAPACK and MAGMA do not utilize previous results for Cholesky decomposition, so that they may not perform very efficiently for the iterative CS signal reconstruction algorithms. We propose an

iterative Cholesky decomposition on FPGAs and GPUs, which is not only suitable for iterative CS signal reconstruction algorithms but also for other applications.

We have investigated the recent progress about sparse signal reconstruction algorithms and their hardware implementation in CS theory in this chapter. In the following chapters, we are going to detail the sparse signal reconstruction algorithms.

Chapter 3

Hard Decision Algorithm: Feedback Orthogonal Pruning Pursuit Algorithm

3.1 Background and Motivation

CS signal reconstruction algorithms are normally computationally expensive. The hard decision CS signal reconstruction algorithms are based on mathematical optimization theory, which requires exhaustive computation for searching for optimal solutions. However, many applications require a fast signal reconstruction for real time signal processing. How do we alleviate the complexity of CS algorithms for fast signal reconstruction? In addition, it is also important for the algorithm to combat noise and interference.

In this chapter, a novel CS reconstruction algorithm, coined *orthogonal pruning pursuit* (OPP), is proposed for fast signal reconstruction. In contrast to traditional greedy CS algorithms, the key idea of OPP is to prune indices of zero elements based on a given index set for finding true nonzero indices. This makes the signal as sparse as possible and thus improves the capability of defeating noise and interference. OPP

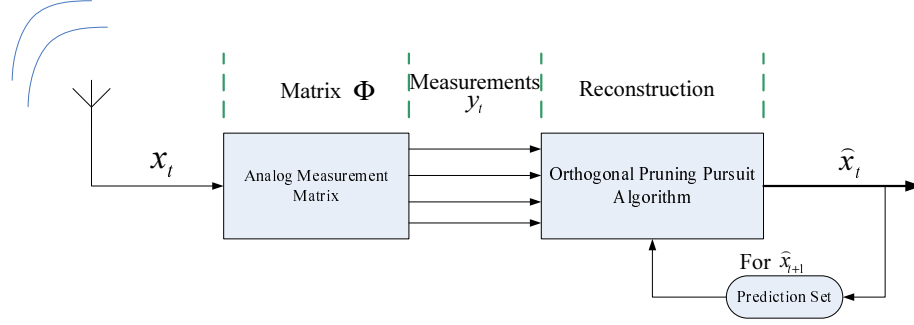


Figure 3.1: The feedback structure for the UWB receiver.

is based on the feedback structure for fast signal reconstruction. The prior knowledge in time sequence signals is exploited, which leads to a substantial reduction of the amount of required measurements [46] [19]. The recovered signal is fed back as prior knowledge into the next reconstruction processing. This low complexity algorithm is quite suitable for parallel computation on FPGAs [16] for fast processing. The UWB system is utilized in the simulation as an example. A performance gain is achieved to improve the capability of de-noising and to reduce the bit error rate (BER) for the UWB high speed communication system.

The remainder of this chapter is organized as follows. Section 4.2 provides the system model and problem formulation by illustrating a scenario of UWB signal and introducing a feedback structure UWB receiver. Section 3.3 gives the derivation and description of the proposed OPP algorithm. Section 3.4 shows numerical simulation results.

3.2 System Model and Problem Formulation

3.2.1 Problem Formulation

We apply sequential CS to UWB communications. Recall that the characteristics of the received UWB signals are discussed in previous chapters. A frame of signal x_t is a high resolution digitalized signal with N -dimensions. Its sparsity (i.e. the number

of nonzero elements) is k and satisfies $k \ll N$. The index set of nonzero elements is defined as the support set, which is given by

$$y_t = \Phi x_t + n_t, \quad (3.1)$$

where n_t is white Gaussian noise. The measurement matrix $\Phi_{M \times N}$ has independent and identically distributed (i.i.d) elements and satisfies the Restricted Isometry Property (RIP). The original signal can be reconstructed successfully from M measurements ($M \ll N$) through CS algorithms[27]. Using low sampling rate ADCs to obtain measurements, ultra-high sampling rate for direct signal acquisition is avoided.

In sequential CS we consider consecutive signal frames. The current frame x_t is highly correlated to the previous one x_{t-1} . We denote the support set of x_{t-1} by T_{t-1} . The range of T_t is statistically predictable from T_{t-1} . Thus, based on transmission rate and T_{t-1} , a prediction index set, denoted as I , is computed for predicting T_t and a feedback structure is proposed to utilize this prediction set I .

3.2.2 Feedback Structure for UWB Receiver

Fig.3.1 shows a feedback structure for sequential CS-based UWB communications. The measurement matrix in Eq.(3.1) is realized in the analog domain [20] [62]. In signal reconstruction, a prediction set I obtained from the previous frame of signal is fed back. The prediction set I is formed by enlarging T_{t-1} based on the known transmission rate. The size of I is several times larger than the size of true T_t but much smaller than the dimension N . The purpose of a large I is to guarantee that T_t is contained in I . The benefit of set I is to reduce the searching range for T_t . Based on this structure, we propose the OPP algorithm to find the T_t and reconstruct x_t , which will be explained in the next section.

3.3 Orthogonal Pruning Pursuit Algorithm

3.3.1 Necessary Lemmas

Given a prediction set I which contains the support set T_t , then how do we shrink I to find T_t ? The following lemmas build a criterion to delete a basis not belonging to T_t . For convenience, we denote y_t by $\mathbf{y}$ and x_t by $\mathbf{x}$. $\hat{\mathbf{x}}$ stands for the estimation of the signal $\mathbf{x}$. The following mathematical notation is used: For a matrix Φ and the index set I , submatrix Φ_I consists of the columns of Φ with indices $i \in I$. Φ_I' is the transpose of matrix Φ_I . The i th column vector in Φ_I is denoted by ϕ_i , $i \in I$. If deleting an index from I , the corresponding submatrix is denoted by Φ_{I-1} . We also define $\langle \cdot \rangle$ as the scalar product.

Lemma 3.0.1. *Given an index set I , suppose $\Phi_I' \Phi_I$ is invertible. The orthogonal projection of given vector y onto subspace S_I is $\mathbf{y}_I$, given by:*

$$\mathbf{y}_I = \Phi_I (\Phi_I' \Phi_I)^{-1} \Phi_I' \mathbf{y}. \quad (3.2)$$

Then the residual $\mathbf{R}_I = \mathbf{y} - \mathbf{y}_I$ is orthogonal to every basis in subspace S_I . $\|\mathbf{R}_I\|^2$ is the minimum distance from $\mathbf{y}$ to any vector $\mathbf{z}$ belonging to S_I , i.e.

$$\|\mathbf{R}_I\|^2 = \inf_{\mathbf{z} \in S_I} \|\mathbf{y} - \mathbf{z}\|^2, \quad \mathbf{z} \in S_I \quad (3.3)$$

The corresponding proof is well known [36] and is thus omitted here. It is easy to verify that the orthogonal projector onto S_I is $P_{S_I} = \Phi_I (\Phi_I' \Phi_I)^{-1} \Phi_I'$. The projection of $\mathbf{y}$ is $\mathbf{y}_I = P_{S_I} \mathbf{y} = \Phi_I (\Phi_I' \Phi_I)^{-1} \Phi_I' \mathbf{y}$. The residual $\mathbf{R}_I = \mathbf{y} - \mathbf{y}_I$ is orthogonal to the subspace S_I , equivalently expressed as $\langle \mathbf{R}_I, \phi_i \rangle = 0$, $\phi_i \in S_I$. From Lemma 1, if deleting a basis from set I , the residual is denoted by $\mathbf{R}_{I-1}$. We also have

$$\|\mathbf{R}_{I-1}\|^2 = \inf_{\mathbf{z} \in S_{I-1}} \|\mathbf{y} - \mathbf{z}\|^2, \quad \mathbf{z} \in S_{I-1} \quad (3.4)$$

Given a set I containing T_t , there exists a unique solution which is the true signal [27] because the matrix Φ is incoherent. Then, we obtain that, when $\|\mathbf{R}_I\| \rightarrow 0$, $\|\hat{\mathbf{x}} - \mathbf{x}\| \rightarrow 0$. Thus, the value in the entry not belonging to the support set should keep small to satisfy a minimum residual and this corresponding index should be deleted. On the other hand, if an index in the support set is pruned, it will lead to a relatively large residual. When shrinking I by deleting redundant indices while keeping the true support set, the residual will correspondingly keep small [43]. Now, a question arises: how do we prune a redundant basis to keep the residual small?

Lemma 3.0.2. *Assume that, after pruning the j th column vector ϕ_j from Φ_I the new matrix Φ_{I-1} spans a subspace S_{I-1} . Through the orthogonal projection from the given vector $\mathbf{y}$ onto S_{I-1} , the residual is $\mathbf{R}_{I-1} = \mathbf{y} - \mathbf{y}_{I-1}$. Let $A = (\Phi_I' \Phi_I)^{-1} \Phi_I'$. Then α_j is the j th row vector in A and $\hat{\mathbf{x}}_j^I$ is the j th coefficient of the estimated signal from projecting $\mathbf{y}$ onto S_I . The difference between the residuals of two projections is expressed as:*

$$\|\mathbf{R}_I - \mathbf{R}_{I-1}\|^2 = \frac{(\hat{\mathbf{x}}_j^I)^2}{\langle \alpha_j, \alpha_j^T \rangle}. \quad (3.5)$$

The corresponding proof can be found in [36]. Obviously for shrinking I the criteria for pruning a basis is to find the one that minimizes Eq.(3.5). However, searching for such a minimum of Eq.(3.5) involves inverting a large matrix, which is computationally expensive. Alternatively, we adopt a heuristic criteria for deleting a basis, which is given by

$$\min \frac{(\hat{\mathbf{x}}_j^I)^2}{\|\phi_j\|^2}. \quad (3.6)$$

In contrast to OMP which searches the most possible true support vector through finding the largest projection [35], Eq.(3.6) uses an opposite approach to delete redundant indices which are least likely in the true support set. In our numerical simulations, Eq.(3.6) always yields the same results as obtained from minimizing

Eq.(3.5). Therefore, in practice, we can compute the solution to Eq.(3.6), rather than Eq.(3.5), to achieve good performance for signal reconstruction.

3.3.2 Algorithm Description

Based on our feedback structure, a prediction set is formed by moderately enlarging the obtained support set of the previous frame. Then it is introduced as an initial index set to the current frame. The procedure of the OPP algorithm is given in details and an illustration is given in Fig. 3.2.

1. Step 1: Initialization:

The index set I is the given prediction set.

The coefficients from orthogonal projection:

$$\hat{\mathbf{x}} := (\Phi_I' \Phi_I)^{-1} \Phi_I' \mathbf{y}$$

2. Step 2: Deleting a basis:

Index $j := \arg \min \frac{\|\hat{\mathbf{x}}_i\|^2}{\|\phi_i\|^2}, i \in I$; ϕ_i is the i th column vector in Φ_I

3. Step 3: Calculating new coefficients:

$$\hat{\mathbf{x}} := (\Phi_{I-1}' \Phi_{I-1})^{-1} \Phi_{I-1}' \mathbf{y}$$

Obtaining residual: $\mathbf{R} = \mathbf{y} - \Phi_{I-1} \hat{\mathbf{x}}$

4. Step 4: Termination:

If the residual is less than an acceptable error threshold, the algorithm is terminated. The support set is forwarded for processing the next frame of signal. Otherwise go to Step 2 to continue shrinking the index set I .

The complexity of this algorithm is mainly in Steps 2 and 3. To avoid the inversion of matrix and speed up computation, Cholesky decomposition can be adopted and implemented on FPGAs for parallel computation[16].

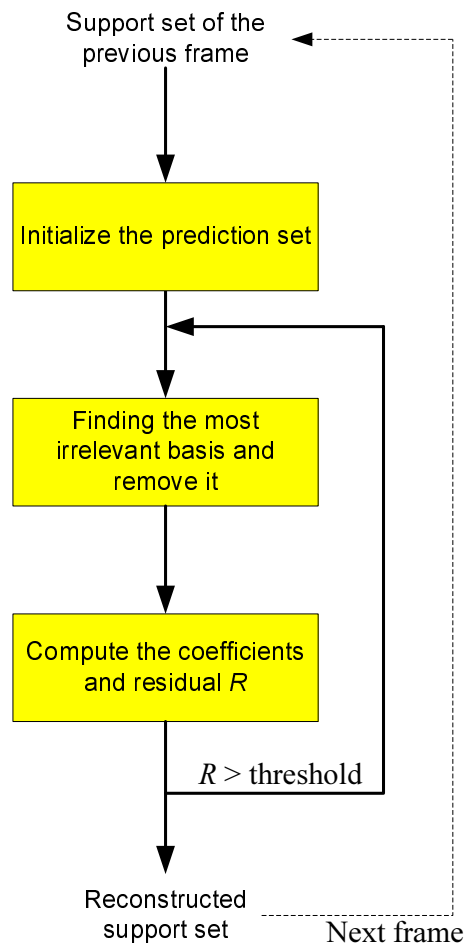


Figure 3.2: Diagram of algorithm

3.4 Numerical Simulation

Based on the proposed feedback structure, we compare the performance of our orthogonal pruning pursuit with OMP and BP. Other l_0 norm optimization CS algorithms, e.g. Stomp, ROMP, and SAMP, have similar performance to OMP[64]. BP is the most used l_1 norm optimization CS algorithm. Therefore, only OMP and BP are simulated to make a comparison.

In UWB communication we first reconstruct the high resolution pulse signals. Fig.3.3(a) shows a section of received UWB signals in a Line-Of-Sight (LOS) environment [1]. The measurement matrix is an i.i.d. The Gaussian random matrix and the measurements are obtained under the Nyquist sampling rate. The measurement number is $M = 55$ ($N = 512$), which is far less than the required number for successful signal recovery when the correlation between different frames is not considered. When the original signal is contaminated by noise, many more measurements are needed for recovery. In OPP algorithm, the prediction set I containing the true support set is seven times larger than the size of the true support set. The reconstruction percentage is defined as $1 - \frac{\|\hat{x}_t - x_t\|}{\|x_t\|}$. The reconstructed signal using OMP, BP, and our feedback OPP are illustrated in Fig.3.3(b), (c), and (d), respectively. Obviously, with so few measurements, the OPP obtains a much better reconstruction performance than the traditional OMP and BP. Hence, by using a feedback structure and the OPP algorithm, one needs only a few measurements to achieve good performance, saving expensive hardware resources.

Furthermore, we exploit the relationship between the number of measurements and reconstruction percentage using different algorithms. We denote by p the ratio of the prediction set size to the true support set size. For the noiseless case, as shown in Fig.3.4, the OPP algorithm needs fewer measurements to achieve an exact recovery compared with OMP and BP. When the prediction set size is large, more measurements are needed. If the prediction set is equal to N , the performance of OPP will be degraded to that of OMP.

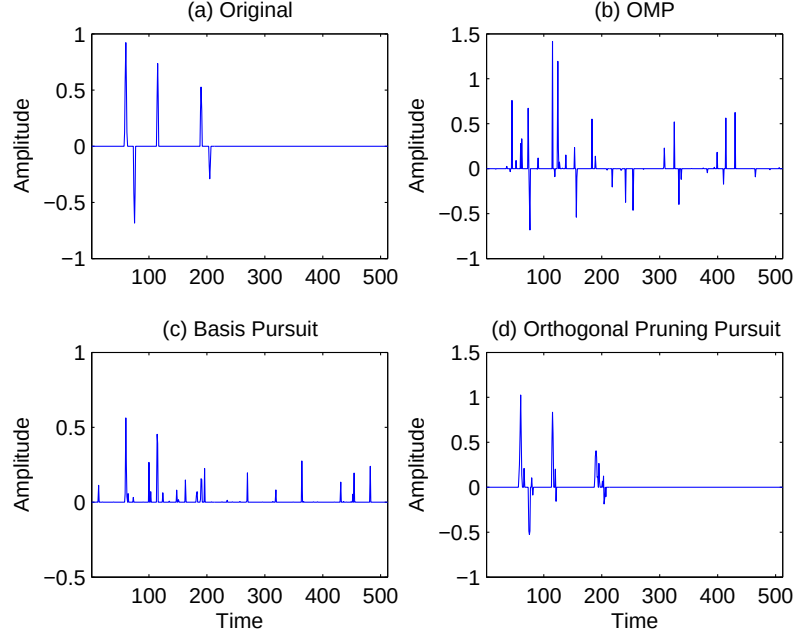


Figure 3.3: Reconstructed UWB echo signals using OMP, BP, and OPP. Time is in ns; Length $N=512$, measurements $M=55$ and the sparsity $K=17$. Signals are contaminated by the noise with $\text{SNR}=12.95\text{dB}$. The reconstruction percentages are respectively: (a)Original UWB echo signal without noise. (b)OMP: 0% (c)BP: 20.48% (d)OPP: 65.83% .

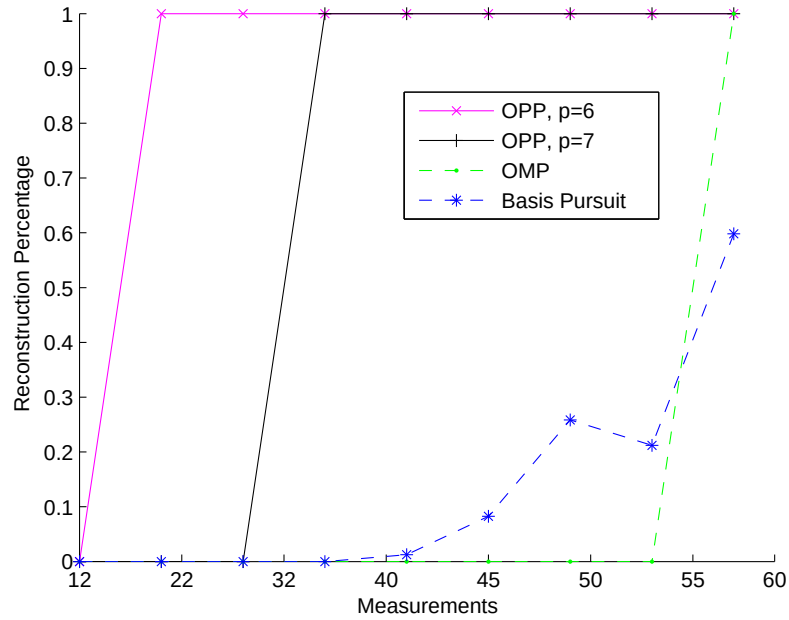


Figure 3.4: Probability of reconstruction vs. measurements without noise

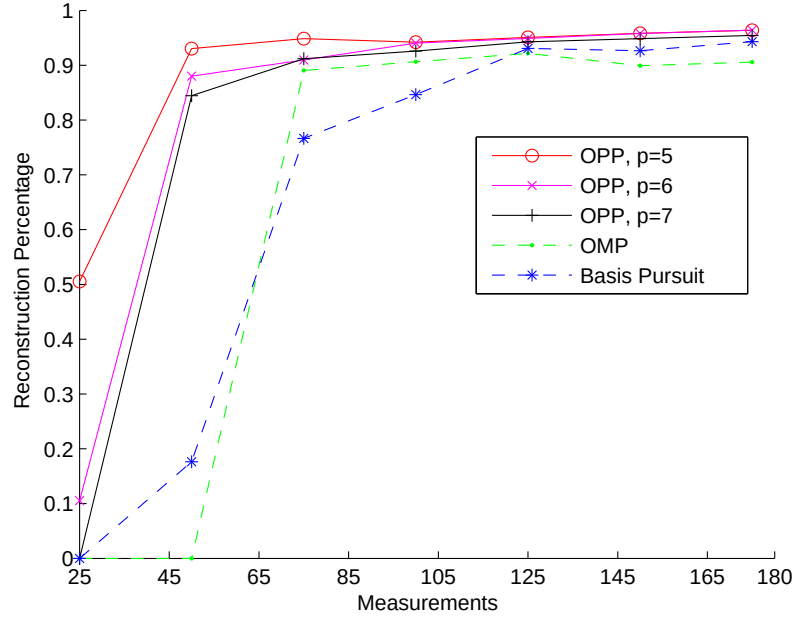


Figure 3.5: Probability of reconstruction vs. measurements in the presence of noise

Considering noise effect, Fig.3.5 shows the reconstruction percentage versus the measurement amount using different algorithms. The signals are contaminated by the noise with SNR=18.65dB. Clearly, to achieve 90% reconstruction percentage, OPP needs fewer measurements compared with OMP and BP. Also the recovered signal from our OPP is sparser than that from traditional OMP. For instance, at 90% reconstruction percentage using 60 measurements, the number of nonzero elements is 35 using OPP with $p = 5$; while it is 85 using traditional OMP. It is also observed that the size of prediction set determines the performance of OPP. A big p will degrade the performance and require more measurements. However, a small p incurs the risk of missing the true support set. An adaptive and more efficient algorithm to construct a small prediction set in OPP is needed, which is our future work.

Fig.3.6 shows BER versus SNR when using 50, 60, and 70 measurements for PAM UWB communications. We can see that our proposed OPP provides better BER performance using the same number of measurements as OMP and BP. When using OPP, more measurements improve the capability to combat the noise and interference,

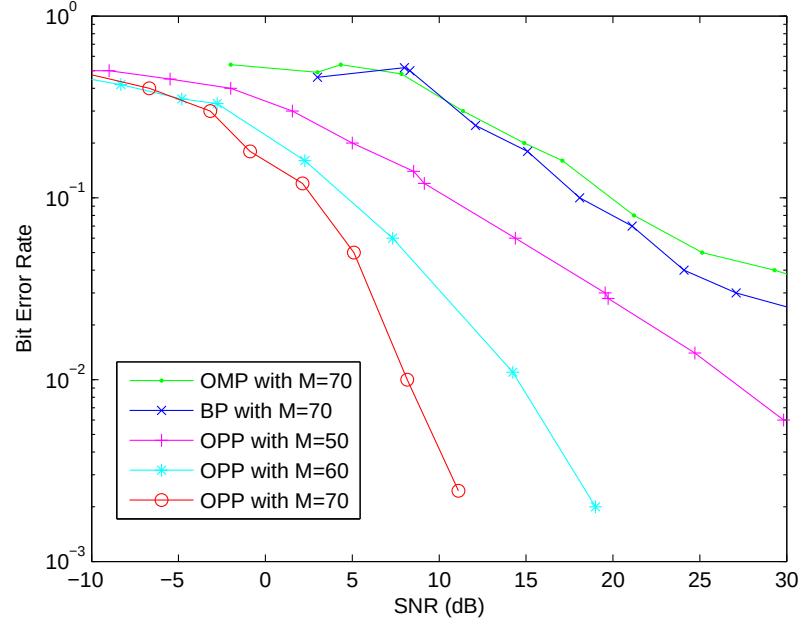


Figure 3.6: BER vs. SNR using different amount of measurements.

leading to a better BER performance for UWB communications. Our scheme can also be modified for other types of pulse modulations, such as PPM and OOK.

In summary, this chapter describes the OPP algorithm for sparse signal reconstruction without the presence of noise. However, the noise is a critical problem in CS theory. We discuss how to handle the noise in signal reconstruction in the next chapter.

Chapter 4

Soft Decision Algorithm: Joint Bayesian Compressed Sensing Algorithm

4.1 Background and Motivation

The soft decision CS signal reconstruction algorithms are based on Bayesian statistical theory. Compared with the hard decision CS algorithms, the soft decision CS algorithms build up a statistical hierarchy structure[59]. One of the typical soft decision CS algorithms is the Bayesian compressed sensing (BCS) algorithm[59], which has good performance to combat the noise. More importantly, BCS can be modified for many applications' requirements. One typical application is to require joint signal reconstruction. For example, in some situations, there are multiple copies of signals that are correlated in space and time, thus providing spatial and temporal redundancies. Take the CS-based Ultra-Wideband (UWB) system * as an example [12][54]. In a typical UWB system as shown in Fig. 4.1, one transmitter

*A UWB system utilizes a short-range, high-bandwidth pulse without carrier frequency for communication, positioning, and radar imaging. One challenge is the acquisition of the high-resolution ultra-short duration pulses. The emergence of CS theory provides an approach to acquiring UWB pulses, possibly under the Nyquist sampling rate [20][76].

periodically sends out ultra-short pulses (typically nano- or subnano-second Gaussian pulses). Surrounding the transmitter, several UWB receivers are receiving the pulses. The received echo signals at one receiver are similar to those received at other receivers in both space and time for the following reasons: (1) at the same time slot, the received UWB signals are similar to each other because they share the same source, which leads to spatial redundancy; and (2) at the same receiver, the received signals are also similar in consecutive time slots because the pulses are periodically transmitted and propagation channels are assumed to change very slowly. Hence the UWB echo signals are correlated both in space and time, which provides spatial and temporal redundancies. Such prior information can be exploited in the joint CS signal reconstruction to improve performance. On the other hand, our work is also motivated to reduce the number of necessary measurements and improve the capability of combating noise. For successful CS signal reconstruction, a certain number of measurements are needed. In the presence of noise, the number of measurement may be greatly increased. However, more measurements lead to more expensive and complex hardware and software in the system [20]. In such a situation, a question arises: can we develop a joint CS signal reconstruction algorithm to exploit temporal and spatial information *a priori* for improving performance in terms of fewer measurements, more noise tolerance, and better quality of reconstructed signal?

Our work and MBCS[61] are both focused on reconstructing multiple signal frames. However, MBCS cannot perform simultaneous multitask signal reconstruction until all measurements have been collected, which is purely in a batch mode and cannot be performed in an online manner. Moreover, MBCS is centralized and is hard to decentralize. Our proposed incremental and decentralized TBCS has a more flexible structure, which can reconstruct multiple signal frames sequentially in time and/or in parallel in space through transferring mutual prior information.

In this chapter, we propose *a novel and flexible Turbo Bayesian Compressed Sensing (TBCS) algorithm for sparse signal reconstruction through exploiting and*

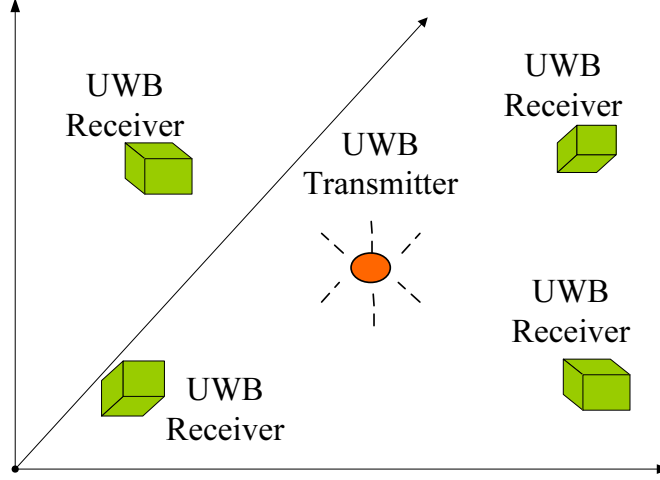


Figure 4.1: A typical UWB system with one transmitter and several receivers

integrating spatial and temporal redundancies in multiple signal reconstruction procedures performed in parallel, in serial, or both. Note the BCS algorithm has an excellent capability of combating noise by employing a statistically hierarchical structure, which is very suitable for transferring prior information. Based on the BCS algorithm, we propose a prior information-based iterative mechanism for information exchange among different reconstruction processes, motivated by the Turbo decoding structure, which is denoted *Turbo BCS*. A primary challenge in the proposed framework is *how to yield and fuse prior information in the signal reconstruction procedure in order to utilize spatial and temporal redundancies*. The proposed algorithm exploits a mathematically elegant framework to impose an exponentially distributed hyperparameter on the existing hyperparameter α of the signal elements. This exponential distribution for the hyperparameter provides an approach to generate and fuse prior information with measurements in the signal reconstruction procedure. An incremental method [39] is developed to find the limited nonzero signal elements, which reduces the computational complexity compared with the expectation maximization (EM) method. A detailed STTBCS algorithm procedure in the case study of UWB systems is also provided to illustrate our algorithm is universal and robust: when the signals have low similarities, the

performance of STTBCS will automatically equal that of the original BCS; on the other hand, when the similarity is high, the performance of STTBCS is much better than the original BCS.

At the end of this chapter, simulation results have demonstrated that our TBCS significantly improves performance. We first use spike signals to illustrate the performance which can be achieved at each iteration employing the original BCS, MBCS, and our TBCS algorithms. The simulations show that our TBCS outperforms the original BCS and MBCS algorithms at each iteration for different similarity levels. We also choose IEEE802.15a [1] UWB echo signals for performance simulation. For the same number of measurements, the reconstructed signal using TBCS is much better compared with the original BCS and MBCS. To achieve the same reconstruction percentage, our proposed scheme needs significantly fewer measurements and is able to tolerate more noise, compared with the original BCS and MBCS algorithms. A distinctive advantage of TBCS is that when the similarity is low, MBCS performance is worse than the original BCS while our TBCS is close to the original BCS and much better than MBCS.

The remainder of this paper is organized as follows. The problem formulation is introduced in Section 4.2. Based on the BCS framework, prior information is integrated into signal reconstruction in Section 4.3. A fast incremental optimization method is detailed in Section 4.4 for the posterior function. Taking UWB systems as a case study, Section 4.5 develops a space-time TBCS algorithm by applying our TBCS into the UWB system. The space-time TBCS algorithm is summarized in Section 4.5. Numerical simulation results are provided in Section 4.6.

4.2 Problem Formulation

Figure 4.2 shows a typical decentralized CS signal reconstruction model. We assume that the signals received at the receiver sides and the transmitted signal are sparse. We also ignore any other effects such as propagation channel and additive noise on

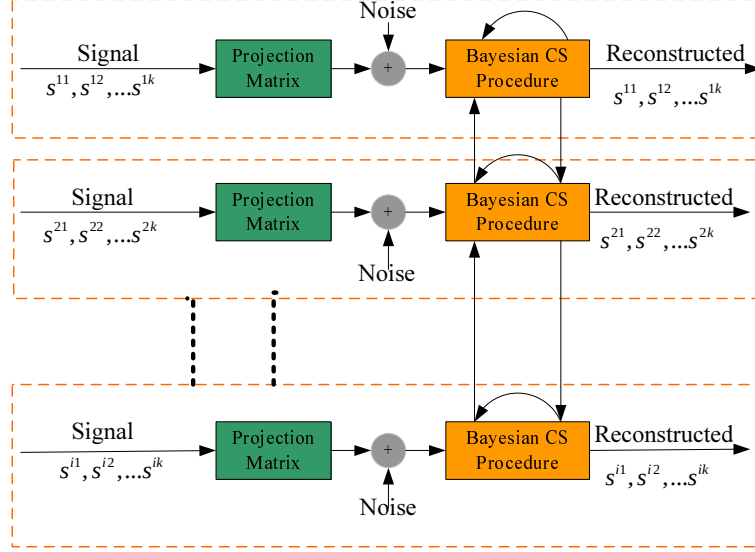


Figure 4.2: Block diagram of decentralized turbo Bayesian compressed sensing

the original signal. Taking the UWB system as an example, the original UWB echo signals, $s^{11}, s^{12}, s^{21}, \dots$ are naturally sparse in the time domain. These signals can be reconstructed in high resolution from a limited number of measurements using low sampling rate ADCs by taking advantage of CS theory. We define a *procedure* as a signal reconstruction process from measurements to recover the signal vector. Signal reconstruction procedures are performed distributively. We will develop a decentralized TBCS reconstruction algorithm to exploit and transfer mutual prior information among multiple signal reconstruction procedures in time sequence and/or in parallel.

We assume that the time is divided into K frames. Temporally, a series of K original signal vectors at the first procedure is denoted as $\mathbf{s}^{11}, \mathbf{s}^{12}, \dots$, and $\mathbf{s}^{1k}$ ($\mathbf{s}^{1k} \in \mathbb{R}^N$), which can be correspondingly recovered from the measurements $\mathbf{y}^{11}, \mathbf{y}^{12}, \dots$, and $\mathbf{y}^{1k}$ ($\mathbf{y}^{1k} \in \mathbb{R}^M$) by using the projection matrix Φ^1 . All the measurement vectors are collected in time sequence. Spatially, at the same time slot, e.g. the k -th frame, a set of I original signal vectors, denoted as $\mathbf{s}^{1k}, \mathbf{s}^{2k}, \dots$, and $\mathbf{s}^{Ik}$ ($\mathbf{s}^{ik} \in \mathbb{R}^N$) are reconstructed from the M -vector measurements, corresponding to $\mathbf{y}^{1k}, \mathbf{y}^{2k}, \dots$, and

$\mathbf{y}^{Ik} (\mathbf{y}^{ik} \in \mathbb{R}^M)$ by using the different projection matrix $\Phi^1, \Phi^2, \dots, \Phi^I$. All the spatial measurement vectors are collected at the same time.

The measurements are linear transforms of the original signals, contaminated by noise, which are given by

$$\mathbf{y}^{ik} = \Phi^i \mathbf{s}^{ik} + \epsilon^{ik} \quad (4.1)$$

with $k = \{1, 2, \dots, K\}$ and $i = \{1, 2, \dots, I\}$; the matrix Φ^i , ($\Phi^i \in \mathbb{R}^{M \times N}$) is the projection matrix with $M \ll N$. The ϵ^{ik} are additive white Gaussian noise with unknown but stationary power β^{ik} . The noise level for different i and k may be different; however, the stationary noise variance can be integrated out in BCS and does not affect the signal reconstruction [38][39][59]. For mathematical convenience, we assume the β^{ik} are identical for all i and k and denote it by β . Without loss of generality, we assume that $\mathbf{s}^{ik}$ is sparse, i.e. most elements in $\mathbf{s}^{ik}$ are zero.

Signal reconstruction is performed among different BCS procedures in parallel and in time sequence. Information is transferred in parallel and serially. Note that the original signals, $\mathbf{s}^{11}, \mathbf{s}^{12}, \mathbf{s}^{22}, \dots$ may be correlated with each other because of the spatial and temporal redundancies. However, without loss of generality, we do not specify the correlation model among the signals at different BCS procedures. This similarity leads to information which can be introduced into decentralized TBCS signal reconstruction *a priori* for improving performance in terms of reducing the number of measurements and improving the capability of combating noise.

For notational simplicity, we abbreviate $\mathbf{s}^{ik}$ into $\mathbf{s}^i$ to utilize one superscript representing either the temporal or spatial index, or both. We use the subscript to represent the element index in the vector. The main notation used throughout this paper is stated in Table 1.

Table 4.1: Notation list

$s_j^i, \mathbf{s}^i, \mathbf{s}: s_j^i$	s_j^i is the j -th signal element of the original signal vector $\mathbf{s}^i$ at the i -th spatial procedure or the i -th time frame; the signal vector $\mathbf{s}^i$ is $\mathbf{s}^i = \{s_j^i\}_{j=1}^N$, which can be abbreviated as $\mathbf{s}$.
$y_j^i, \mathbf{y}^i, \mathbf{y}: y_j^i$	y_j^i is the j -th element of the measurement vector for reconstructing the signal vector $\mathbf{s}^i$ is collected at either i -th spatial procedure or i -th time frame, which has $\mathbf{y}^i = \{y_j^i\}_{j=1}^M$; $\mathbf{y}^i$ can be abbreviated as $\mathbf{y}$.
Φ^i :	the measurement matrix utilized for compressing the signal vector $\mathbf{s}^i$ to yield $\mathbf{y}^i$.
β :	the noise variance.
$\alpha_j^i, \boldsymbol{\alpha}^i, \boldsymbol{\alpha}: \alpha_j^i$	α_j^i is the j -th hyperparameter imposed on the corresponding signal element s_j^i ; it can be abbreviated as α_j ; and it has $\boldsymbol{\alpha}^i = \{\alpha_j^i\}_{j=1}^N$; $\boldsymbol{\alpha}^i$ can be abbreviated as $\boldsymbol{\alpha}$.
$\lambda_j^i, \boldsymbol{\lambda}^i, \boldsymbol{\lambda}: \lambda_j^i$	λ_j^i is the parameter controlling the distribution of the corresponding hyperparameter α_j^i for mutual prior information transfer, where $\boldsymbol{\lambda}^i = \{\lambda_j^i\}_{j=1}^N$ and it can be abbreviated as $\boldsymbol{\lambda}$.

4.3 Mutual Information Transfer in Bayesian Compressed Sensing

In this section, we propose a Turbo BCS algorithm to provide a general framework for yielding and fusing prior information from other parallel or serial reconstructed signals. We first introduce the standard BCS framework, in which selecting the hyperparameter $\boldsymbol{\alpha}^i$ imposed on the signal element is the key issue. Then we impose an exponential prior distribution on the hyperparameter $\boldsymbol{\alpha}^i$ with parameter $\boldsymbol{\lambda}^i$. The previous reconstructed signal element will impact the parameter $\boldsymbol{\lambda}^i$ to affect the $\boldsymbol{\alpha}^i$

distribution, yielding prior information. Next, prior information will be integrated into the current signal estimation.

4.3.1 Bayesian Compressed Sensing Framework

Starting with Gaussian distributed noise, the BCS framework [59][38] builds a Bayesian regression approach to reconstruct the original signal with additive noise from the compressed measurements. In the BCS framework, a Gaussian prior distribution is imposed over each signal element, which is given by

$$\begin{aligned} P(\mathbf{s}^i | \boldsymbol{\alpha}^i) &= \prod_{j=1}^N \left(\frac{\alpha_j^i}{2\pi} \right)^{1/2} \exp \left(-\frac{(s_j^i)^2 \alpha_j^i}{2} \right) \\ &\sim \prod_{j=1}^N \mathcal{N}(s_j^i | 0, (\alpha_j^i)^{-1}), \end{aligned} \quad (4.2)$$

where α_j^i is the hyperparameter for the signal element s_j^i . The zero-mean Gaussian prior is independent for each signal element. By applying Bayes' rule, the *a posteriori* probability of the original signal is given by

$$\begin{aligned} P(\mathbf{s}^i | \mathbf{y}^i, \boldsymbol{\alpha}^i, \beta) &= \frac{P(\mathbf{y}^i | \mathbf{s}^i, \beta) P(\mathbf{s}^i | \boldsymbol{\alpha}^i)}{P(\mathbf{y}^i | \boldsymbol{\alpha}^i, \beta)} \\ &\sim \mathcal{N}(\mathbf{s}^i | \boldsymbol{\mu}^i, \boldsymbol{\Sigma}^i), \end{aligned} \quad (4.3)$$

where $\mathbf{A} = \text{diag}(\boldsymbol{\alpha}^i)$. The covariance and the mean of the signal are given by

$$\boldsymbol{\Sigma}^i = (\beta^{-2}(\boldsymbol{\Phi}^i)^T \boldsymbol{\Phi}^i + \mathbf{A})^{-1} \quad (4.4)$$

and

$$\boldsymbol{\mu}^i = \beta^{-2} \boldsymbol{\Sigma}^i (\boldsymbol{\Phi}^i)^T \mathbf{y}^i \quad (4.5)$$

Then, we obtain the estimation of the signal, $\hat{\mathbf{s}}^i$, which is given by

$$\hat{\mathbf{s}}^i = ((\Phi^i)^T \Phi^i + \beta^2 \mathbf{A})^{-1} (\Phi^i)^T \mathbf{y}^i \quad (4.6)$$

In order to estimate the hyperparameters α^i and $\mathbf{A}$, the maximum likelihood function based on observations is given by

$$\begin{aligned} \alpha^i &= \arg \max_{\alpha^i} P(\mathbf{y}^i | \alpha^i, \beta) \\ &= \arg \max_{\alpha^i} \int P(\mathbf{y}^i | \mathbf{s}^i, \beta) P(\mathbf{s}^i | \alpha^i) d\mathbf{s}^i \end{aligned} \quad (4.7)$$

where, by integrating out $\mathbf{s}^i$ and maximizing the posterior with respect to α^i , the hyperparameter diagonal matrix $\mathbf{A}$ is estimated. Then, the signal can be reconstructed using Eq. (4.6).

The matrix $\mathbf{A}$ plays a key role in the signal reconstruction. The hyperparameter diagonal matrix $\mathbf{A}$ can be used to transfer the mutual prior information by sharing the same $\mathbf{A}$ among all signals [61]. In such a way, if signals have many common nonzero elements, the signal reconstruction will benefit from such a similarity. However, when the similarity level is low, the transferred “wrong” information may impair the signal reconstruction[61].

Alternatively, we find a soft approach to integrating information *a priori* in a robust way. An exponential priori distribution is imposed on the hyperparameter α^i controlled by the parameter λ^i . The previously reconstructed signal elements will impact the λ^i and change the α^i distribution to yield prior information. Then, the hyperparameter α^i conditioned on λ^i will join the current signal estimation using the maximum a posterior (MAP) criterion.

4.3.2 Yielding Prior Information

The key idea of our TBCS algorithm is to *impose an exponential distribution on the hyperparameter α_j^i and exchange information among different BCS signal*

reconstruction procedures using the exponential distribution in a turbo iterative way.

In each iteration, the information from other BCS procedures will be incorporated into the exponential prior and then used for the signal reconstruction of the current BCS signal reconstruction procedure being considered. Note that, in the standard BCS [38], a Gamma distribution with two parameters is used for α_j^i . The reason we adopt an exponential distribution here is that we need to handle only one parameter for the exponential distribution, which is much simpler than the Gamma distribution, while both distributions belong to the same family of distributions.

We assume that hyperparameter α_j^i satisfies the exponential prior distribution, which is given by,

$$P(\alpha_j^i | \lambda_j^i) = \begin{cases} \lambda_j^i e^{-\lambda_j^i \alpha_j^i}, & \text{if } \alpha_j^i \geq 0 \\ 0, & \text{if } \alpha_j^i < 0 \end{cases}, \quad (4.8)$$

where λ_j^i ($\lambda_j^i > 0$) is the hyperparameter of the hyperparameter α_j^i . By assuming mutual independence, we have

$$P(\boldsymbol{\alpha}^i | \boldsymbol{\lambda}^i) = \left(\prod_{j=1}^N \lambda_j^i \right) \exp \left(\sum_{j=1}^N -\lambda_j^i \alpha_j^i \right). \quad (4.9)$$

By choosing the above exponential prior, we can obtain the marginal probability distribution of the signal element depending on the parameter λ_j^i by integrating α_j^i out, which is given by:

$$\begin{aligned} P(s_j^i | \lambda_j^i) &= \int P(s_j^i | \alpha_j^i) P(\alpha_j^i | \lambda_j^i) d\alpha_j^i \\ &= (2\pi)^{-\frac{1}{2}} \Gamma\left(\frac{3}{2}\right) \lambda_j^i \left(\lambda_j^i + \frac{(s_j^i)^2}{2} \right)^{-\left(\frac{3}{2}\right)} \end{aligned} \quad (4.10)$$

where $\Gamma(\cdot)$ is the gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. The detailed derivation is shown in Appendix A.2.

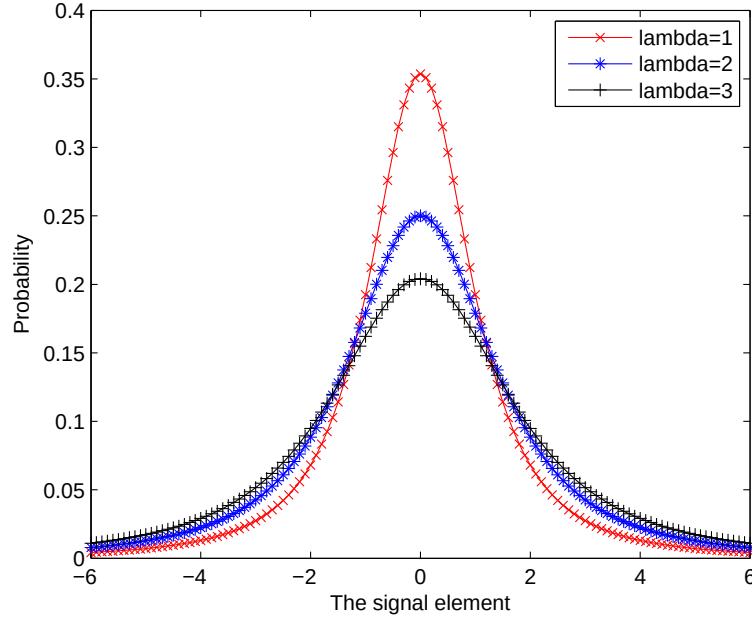


Figure 4.3: The distribution $P(s_j^i | \lambda_j^i)$

Figure 4.3 shows the signal element distribution conditioned on the hyperparameter λ_j^i . Obviously, the bigger the parameter λ_j^i , the more likely the corresponding signal element can take a larger value. Intuitively, this looks very much like a Laplace prior which is sharply peaked at zero [63]. Here, λ_j^i is the key of introducing information *a priori* based on reconstructed signal elements.

Compared with the Gamma prior distribution imposed on the hyperparameter λ_j^i [38][39], the exponential distribution has only one parameter while the Gamma distribution has two degrees of freedom. In many applications (e.g., communication networks), transferring one parameter is much easier and cheaper using the exponential distribution than handling two parameters. The exponential prior distribution does not degrade the performance, which can encourage the sparsity (see Appendix A). Also, it is computationally tractable to use the exponential distribution.

Now the challenge is that, given the j -th reconstructed signal element s_j^b from the b -th BCS procedure, how does one yield prior information to impact the hyperparameters in the i -th BCS procedure for reconstructing the j -th signal element

s_j^i ? When multiple BCS procedures are performed to reconstruct the original signals (no matter whether they are in time sequence or in parallel), the parameters of the exponential distribution, λ_j^i , can be used to convey and incorporate information from other BCS procedures. To this end, we consider the conditional probability, $P(\alpha_j^i | s_j^b, \lambda_j^i)$, for α_j^i , given an observation element from another BCS procedure, s_j^b ($b \neq i$), and λ_j^i . Since the proposed algorithm does not use a specific model for the correlation of signals at different BCS procedures, we propose the following simple assumption when incorporating the information from other BCS procedures into λ_j^i , for facilitating the TBCS algorithm:

Assumption: For different i and b , we assume $\alpha_j^i = \alpha_j^b, \forall i, b$.

Essentially, this assumption implies the same locations of nonzero elements for different BCS procedures. In other words, the hyperparameter α_j^i for the j -th signal element is the same over different signal reconstruction procedures. Then, mutual information can be transferred through the shared hyperparameter α_j^i as proposed in [61]. However, the algorithm in [61] is a centralized MBCS algorithm, so the signal reconstructions for different tasks cannot be performed until all measurements are collected. Note that this technical assumption is only for deriving the algorithm for information exchange. It does not mean that the proposed algorithm only works for the situation in which all signals share the same locations of nonzero elements. Our proposed algorithm based on this assumption can provide a flexible and decentralized way to transfer mutual information.

Based on the assumption, we obtain

$$\begin{aligned}
P(\alpha_j^i | s_j^b, \lambda_j^i) &= \frac{P(s_j^b, \alpha_j^i | \lambda_j^i)}{P(s_j^b, \lambda_j^i)} \\
&= \frac{P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{\int P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^i) d\alpha_j^i} \\
&= \frac{(\tilde{\lambda}_j^i)^{\frac{3}{2}} \exp(-\tilde{\lambda}_j^i \alpha_j^i)}{\Gamma(\frac{3}{2})}
\end{aligned} \tag{4.11}$$

where $\Gamma(\cdot)$ is the gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. The detailed derivation is given in Appendix A.2.

Obviously, the posterior $(\alpha_j^i | s_j^b, \lambda_j^i)$ also belongs to the exponential distribution [2]. Compared with the original prior distribution in Eq. (4.8), given the j -th reconstructed signal element s_j^b from the b -th BCS procedure, the hyperparameter λ_j^i in the i -th BCS procedure controlling the prior distribution is actually updated to $\tilde{\lambda}_j^i$, which is given by

$$\tilde{\lambda}_j^i = \lambda_j^i + \frac{(s_j^b)^2}{2}. \quad (4.12)$$

If information from n BCS procedures $b_1, \dots, b_n$ are introduced, the parameter $\tilde{\lambda}_j^i$ is then updated to

$$\begin{aligned} & P(\alpha_j^i | s_j^{b_1}, s_j^{b_2}, \dots, s_j^{b_n}, \lambda_j^i) \\ &= \frac{(\tilde{\lambda}_j^i)^{\frac{2n+1}{2}} \exp(-\tilde{\lambda}_j^i \alpha_j^i)}{\Gamma(\frac{2n+1}{2})}. \end{aligned} \quad (4.13)$$

where

$$\tilde{\lambda}_j^i = \lambda_j^i + \frac{\sum_{i=1}^n (s_j^{b_i})^2}{2}. \quad (4.14)$$

The derivation details are given in Appendix A.

Eq. (4.12) and Eq. (4.14) show that how the single or multiple signal elements $s_j^{b_n}, j = 1, 2, \dots, N, n = 1, 2, \dots$ from other BCS procedures impact the hyperparameter of the signal element $s_j^i, j = 1, 2, \dots, N$ at the same location in the i -th BCS signal reconstruction. Note that the b -th BCS signal reconstruction may be previously performed or is ongoing with respect to the i -th BCS procedure. This provides significant flexibility to apply our TBCS in different situations.

4.3.3 Incorporating Prior Information into BCS

Now, we study how to incorporate the prior information obtained in the previous subsection into the signal reconstruction procedure. In order to incorporate prior information, provided by the external information, we maximize the log posterior based on Eq. (4.7), which is given by

$$\begin{aligned} L(\boldsymbol{\alpha}^i) &= \log P(\mathbf{y}^i | \boldsymbol{\alpha}^i, \beta) P(\boldsymbol{\alpha}^i | \{\mathbf{s}^b\}, \boldsymbol{\lambda}^i) \\ &= \log P(\mathbf{y}^i | \boldsymbol{\alpha}^i, \beta) + \log P(\boldsymbol{\alpha}^i | \{\mathbf{s}^b\}, \boldsymbol{\lambda}^i) \end{aligned} \quad (4.15)$$

Therefore, the estimation of $\boldsymbol{\alpha}^i$ not only depends on the local measurements, which are in the first term $\log P(\mathbf{y}^i | \boldsymbol{\alpha}^i, \beta)$, but also relies on the external signal elements $\{\mathbf{s}^b\}$ through the parameter $\boldsymbol{\lambda}^i$, which are in the second term $\log P(\boldsymbol{\alpha}^i | \{\mathbf{s}^b\}, \boldsymbol{\lambda}^i)$.

An expectation maximization (EM) method can be utilized for the signal estimation. Recall that the signal vector $\mathbf{s}^i$ is Gaussian distributed conditioned on $\boldsymbol{\alpha}^i$, while $\boldsymbol{\alpha}^i$ also conditionally depends on the parameters $\boldsymbol{\lambda}^i$. Eq. (4.3) shows that the conditional distribution of $\mathbf{s}^i$ satisfies $\mathcal{N}(\mu, \Sigma)$. Then, applying a similar argument to that in [38], we consider $\mathbf{s}^i$ as hidden data and then maximize the following posterior expectation, which is given by

$$E_{\mathbf{s}^i | \mathbf{y}^i, \boldsymbol{\alpha}^i} [\log P(\mathbf{s}^i | \boldsymbol{\alpha}^i, \beta) P(\boldsymbol{\alpha}^i | \boldsymbol{\lambda}^i)] . \quad (4.16)$$

By differentiating (4.16) with respect to $\boldsymbol{\alpha}^i$ and setting the differentiation to zero, we obtain an update, which is given by

$$\alpha_j^i = \frac{3}{(s_j^i)^2 + \Sigma_{jj}^i + 2\lambda_j^i}, \quad (4.17)$$

where Σ_{jj}^i is the j -th diagonal element in the matrix Σ^i . The detail of the derivation is given in Appendix A.3. Basically, the hyperparameters $\boldsymbol{\alpha}^i$ are interactively estimated

and most of them will tend to infinity, which means most corresponding signal elements are zero. Only the nonzero signal elements are estimated.

Considering the computation of the matrix inverse (with complexity $O(n^3)$) associated with the process, the EM algorithm has a large computational cost. Even though a Cholesky decomposition can be applied to alleviate the calculation [14][16], the EM method still incurs a significant computational cost. We will provide an incremental method for the optimization to reduce the computational cost.

4.4 Incremental Optimization

In this section, we utilize an incremental optimization to incorporate transferred prior information and optimize the posterior function. Due to the inherit sparsity of the signal, the incremental method finds the limited nonzero elements by separating and testing a single index one by one, which alleviates the computational cost compared with the EM algorithm. Note that the key principle is similar to that of the fast relevance vector machine algorithm in [38]. However, the incorporation of the hyperparameter λ^i brings significant difficulty for deriving the algorithm. For convenience, we abbreviate α^i as α and y^i as y , because we are focusing on the current signal estimation.

In order to introduce prior knowledge, the target log posterior function can be written as

$$\begin{aligned}
\alpha &= \arg \max_{\alpha} L(\alpha) \\
&= \arg \max_{\alpha} (\log P(y|\alpha, \beta^2) P(\alpha|x)) \\
&= \arg \max_{\alpha} (\log P(y|\alpha, \beta^2) + \log P(\alpha|\{s^b\}, \lambda)) \\
&= \arg \max_{\alpha} (L_1(\alpha) + L_2(\alpha))
\end{aligned} \tag{4.18}$$

where $L_1(\alpha)$ is the signal estimation term from local observation and $L_2(\alpha)$ introduces information *a priori* from other external BCS procedures.

In contrast to the complex EM optimization, the incremental algorithm starts by searching for a nonzero signal element and iteratively adds it to the candidate index set for the signal reconstruction, an algorithm which is similar to the greedy pursuit algorithm. Hence, we isolate one index, assuming the j -th element, which is given by

$$\begin{aligned} L(\boldsymbol{\alpha}) &= L(\boldsymbol{\alpha}_{-j}) + l(\alpha_j) \\ &= L_1(\boldsymbol{\alpha}_{-j}) + l_1(\alpha_j) + L_2(\boldsymbol{\alpha}_{-j}) + l_2(\alpha_j) \end{aligned} \quad (4.19)$$

where $l_1(\alpha_j)$ is the separated term associated with the j -th element from the posterior function $L(\boldsymbol{\alpha}_i)$. The remaining term is $L_1(\boldsymbol{\alpha}_{-j})$, resulting from removing the j -th index.

Initially, all the hyperparameters $\lambda_j, j = \{1, 2, \dots, N\}$ are set to zero. When the transferred signal elements are zero, i.e. $s_j^b = 0; j = \{1, 2, \dots, N\}$, the updated hyperparameters will also be zeros, i.e. $\tilde{\lambda}_j^i = 0, j = \{1, 2, \dots, N\}$ according to Eq. (4.12) and Eq. (4.14). This implies no prior information and the term $L_2(\boldsymbol{\alpha}) = 0$ based on Eq. (4.8), which is equivalent to the original BCS algorithm[39][59].

Suppose that the external information from other BCS procedures is incorporated, i.e., $s_j^b \neq 0, \tilde{\lambda}_j^i \neq 0$, and $L_2(\boldsymbol{\alpha}) \neq 0$. We target maximizing the separated term by considering the remaining term $L(\boldsymbol{\alpha}_{-j})$ as fixed. Then, the posterior function separating a single index is given by

$$\begin{aligned} l(\alpha_j) &= l_1(\alpha_j) + l_2(\alpha_j) \\ &= \frac{1}{2} \left(\log \frac{\alpha_j}{\alpha_j + g_j} + \frac{h_j^2}{\alpha_j + g_j} \right) + \log \lambda_j - \lambda_j \alpha_j \end{aligned} \quad (4.20)$$

where,

$$g_j = \phi_j^T E_{-j}^{-1} \phi_j, \quad (4.21)$$

$$h_j = \phi_j^T E_{-j}^{-1} y, \quad (4.22)$$

and

$$E_{-j} = \beta^2 I + \sum_{k \neq j} \alpha_k^{-1} \phi_k \phi_k^{-1} \quad (4.23)$$

where ϕ_j is the j -th column vector of the matrix Φ . The detailed derivation is provided in Appendix A. Then, we seek for a maximum of the posterior function, which is given by

$$\alpha_j^* = \arg \max_{\alpha_j} l(\alpha_j) = \arg \max_{\alpha_j} (l_1(\alpha_j) + l_2(\alpha_j)). \quad (4.24)$$

When there is no external information incorporated, the optimal hyperparameter is given by [39]

$$\alpha_j^* = \arg \max_{\alpha_j} (l_1(\alpha_j)), \quad (4.25)$$

where

$$\alpha_j^* = \begin{cases} \frac{h_j^2}{g_j^2 - h_j}, & \text{if } g_j^2 > h_j; \\ \infty, & \text{otherwise.} \end{cases} \quad (4.26)$$

When external information is incorporated, to maximize the target function Eq.(4.20), we compute the first order derivative of $l(\alpha_j)$, which is given by

$$\begin{aligned} l'(\alpha_j) &= \frac{g_j}{2\alpha_j(\alpha_j + g_j)} - \frac{h_j^2}{2(\alpha_j + g_j)^2} - \lambda_j \\ &= \frac{f(\alpha_j, g_j, h_j, \lambda_j)}{\alpha_j(\alpha_j + g_j)^2}, \end{aligned} \quad (4.27)$$

where $f(\alpha_j, g_j, h_j, \lambda_j)$ is a cubic function with respect to α_j . By setting Eq. (4.27) to zero, we get the optimum α_j^* .

By setting Eq.(4.27) to zero, we get the optimum solution for the posterior likelihood function $l(\alpha_j)$, which is given by

$$\alpha_j = \begin{cases} \alpha_j^*, & \text{if } g_j^2 > h_j; \\ \infty, & \text{otherwise.} \end{cases} \quad (4.28)$$

The details are given in Appendix A.

Therefore, in each iteration only one signal element is isolated and the corresponding parameters are evaluated. After several iterations, most of the nonzero signal elements are selected into the candidate index set. Due to the sparsity of the signal, after a limited number of iterations, only a few signal elements are selected and calculated, which greatly increases the computational efficiency.

4.5 Case Study: Turbo Bayesian Compressed Sensing for UWB Systems

The TBCS algorithm can be applied in various applications. A typical application is the UWB communication/positioning system. Our proposed TBCS algorithm will be applied to the UWB system to fully exploit the redundancies in both space and time, which is called *Space-Time Turbo BCS (STTBCS)*. In this section, we first introduce the UWB signal model. Then, the structure to transfer spatial and temporal prior information in the CS-based UWB system is explained in detail. Finally, we summarize the STTBCS algorithm.

4.5.1 UWB System Model

In a typical UWB communication/positioning system, suppose there is only one transmitter, which transmits UWB pulses on the order of nano- or subnano-seconds. As shown in Figure 4.1, several receivers, or base-stations, are responsible for receiving

the UWB echo signals. The time is divided into frames. The received signal at the i -th base station and the k -th frame in the continuous time domain is given by

$$\mathbf{s}^{ik}(t) = \sum_{l=1}^L a_l p'(t - t_l), \quad (4.29)$$

where L is the number of resolvable propagation paths, a_l is the attenuation coefficient of the l -th path, and t_l is the time delay of the l -th propagation path. We denote by $p(t)$ the transmitted Gaussian pulse and by $p'(t)$ the corresponding received pulse which is close to the original pulse waveform but has more or less distortion resulting from the frequency dependent propagation channels. At the same frame or time slot, there is only one transmitter but multiple receivers which are nearby in the environment. Therefore, the received echo UWB signals at different receivers are similar at the same time, thus incurring spatial redundancy. In other words, the received signals share many common nonzero element locations. Typically, around 30%-70% of the nonzero element indices are the same in one frame according to our experimental observation [13]. In particular, no matter what kind of signal modulation is used for UWB communication, such as pulse amplitude modulation (PAM), on-off keying (OOK), or pulse position modulation (PPM), the UWB echo signals among receivers are always similar, thus the spatial redundancy always exists. In this case, the spatial redundancy can be exploited for good performance using the proposed space TBCS algorithm.

In one base station, the consecutively received signals can also be similar. Suppose that in UWB positioning systems the pulse repetition frequency is fixed. When the transmitter moves, the signal received at the i -th base station and the $(k + 1)$ -th frame can be written as

$$\mathbf{s}^{i(k+1)}(t) = \sum_{l=1}^{L'} a'_l p'(t - \tau - t_l), \quad (4.30)$$

Compared with Eq. (4.29), τ stands for the time delay which comes from the position change of the transmitter. In high precision positioning/tracking systems, this τ is always relatively small, which makes the consecutive received signals similar. Due to the similar propagation channels, the numbers L and L' , as well as a_l and a'_l are similar in consecutive frames. This leads to the temporal redundancy. In our experiments, about 10%-60% of the nonzero element locations in two consecutive frames are the same[13]. Then, this temporal redundancy can be exploited for good performance by using the Time TBCS algorithms. Actually, there exist both spatial and temporal redundancies in the UWB communication/positioning system. Therefore we can utilize the STBCS algorithm for good performance.

To archive a high precision of positioning and a high speed communication rate, we have to acquire ultra-high resolution UWB pulses, which demands ultra-high sampling rate ADCs. For instance, it requires pico-second level time information and 10Gsample/s or even higher sampling rate ADCs to achieve millimeter (mm) positioning accuracy for UWB positioning systems [14], which is prohibitively difficult. UWB echo signals are inherently sparse in the time domain. This property can be utilized to alleviate the problem of an ultra-high sampling rate. Then the high resolution UWB pulses can be indirectly obtained and reconstructed from measurements acquired using lower sampling rate ADCs.

The system model of the CS-based UWB receiver can use the same model as that in Figure 4.2. The received UWB signal at the i -th base station is first “compressed” using an analog projection matrix [20]. The hardware projection matrix consists of a bank of Distributed Amplifiers (DA). Each DA functions like a wideband FIR filter with different configurable coefficients [20]. The output of the hardware projection matrix can be obtained and digitized by the following ADCs to yield measurements. For mathematical convenience, the noise generated from the hardware and ADCs is modeled as Gaussian noise added to the measurements. When several sets of measurements are collected at different base stations, a joint UWB signal reconstruction can be performed. This process is modeled in Eq. (4.1).

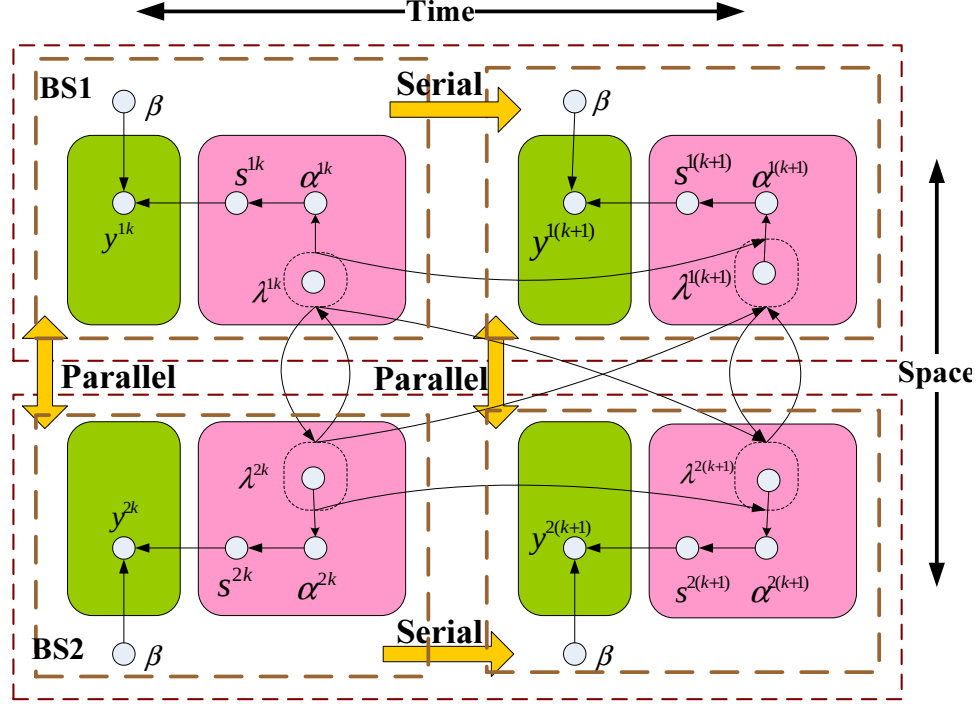


Figure 4.4: Block diagram of space-time turbo Bayesian compressed sensing

4.5.2 STTBCS: Structure and Algorithm

We apply the proposed TBCS to UWB systems to develop the STTBCS algorithm. Figure 4.4 illustrates the structure of our STTBCS algorithm and explains how mutual information is exchanged. For simplicity, only two base stations (BS1 and BS2) and two consecutive frames of UWB signals (the k -th and $(k+1)$ -th) in each base station are illustrated. For each BCS procedure, Figure 4.4 also depicts the dependence among measurements, noise, signal elements, and hyperparameters.

In the STTBCS, multiple BCS procedures in multiple time slots are performed. Between BS1 and BS2, the signal reconstruction for $\mathbf{s}^{1(k+1)}$ and $\mathbf{s}^{2(k+1)}$ are carried out simultaneously while the information in $\mathbf{s}^{1k}$ and $\mathbf{s}^{2k}$, the previous frame, is also used.

Procedure 1 shows the details of the STTBCS algorithm. We start with the initialization of the noise, hyperparameters α , and the candidate index set Ω (an index set containing all possibly nonzero element indices). Then, the information

from previous reconstructed signals and from other base stations are utilized to update the hyperparameter $\boldsymbol{\lambda}$. The term g_j and h_j are also computed. The term $g_j^2 > h_j$ is then used to add the j -th element from the candidate index set. A convergence criterion is used to test whether the differences between successive values for any $\alpha_j, j = \{1, 2, \dots, N\}$ are sufficiently small compared to a certain threshold. When the iterations are completed, the noise level β will be re-estimated from setting $\frac{\partial L}{\partial \beta} = 0$ using the same method in [38], which is given by:

$$(\beta^2)^{new} = \frac{\|y - \Sigma S\|^2}{N - \sum_{i=1}^M (1 - \alpha_i \Sigma_{ii})}, \quad (4.31)$$

where Σ_{ii} is the diagonal element in the matrix Σ . The details of the above STTBCS algorithm are summarized in Procedure 2. Note that only the nonzero signal elements from the local measurements can introduce prior information and thus update the hyperparameter $\tilde{\lambda}_j$. In other words, only if it satisfies $g_j^2 > h_j$ can the parameter $\tilde{\lambda}_j$ be updated. This avoids the adverse effects from wrong prior information to add a zero signal element into the candidate index set.

4.6 Simulation Results

Numerical simulations are conducted to evaluate the performance of the proposed TBCS algorithm, compared with the MBCS [61] and original BCS algorithms [59]. We use spike signals and experimental UWB echo signals [1] for the performance test. The quality of the reconstructed signal is measured in terms of the reconstruction percentage, which is defined as

$$1 - \frac{\|\mathbf{s} - \tilde{\mathbf{s}}\|_2}{\|\mathbf{s}\|_2}, \quad (4.32)$$

where $\mathbf{s}$ is the true signal and $\tilde{\mathbf{s}}$ is the reconstructed signal.

Our TBCS algorithm performance is largely determined by how the introduced signal is similar to the objective signal. In other words, we consider how many common nonzero element locations are shared between the objective signal and the introduced signals. Then we define the similarity as

$$P_s = \frac{K_{com}}{K_{obj}} \quad (4.33)$$

where K_{obj} is the number of nonzero signal elements in the objective unrecovered signal; K_{com} is the number of the common nonzero element locations among the transferred reconstructed signals and objective signal; and P_s represents the similarity level as a percentage. Note that, without loss of generality, we only consider the relative number of common nonzero element locations to measure the similarity, ignoring any amplitude correlation. Hence, when $P_s = 100\%$, it does not mean that the signals are the same but means that they have the same nonzero element locations; the amplitudes may not be the same.

Our TBCS algorithm performance is compared with MBCS and BCS using different types of signals, different similarity levels, noise powers, and measurement numbers.

4.6.1 Spike Signal

We first generate four scenarios of spike signals with the same length $N = 512$, which have the same number (20) of nonzero signal elements with random locations and Gaussian distributed (mean=0, variance=1) amplitudes. One spike signal is selected as the objective signal, as shown in Figure 4.5. With respect to the objective signal, the other three signals have a similarity of 25%, 50%, and 75%, which will be introduced as prior information. The objective signal is then reconstructed using the original BCS, MBCS, and TBCS algorithms, respectively, with the same number of measurements ($M=62$) and the same noise variance 0.15 ($\text{SNR} \simeq 6\text{dB}$). We also

investigate the performance gain (in terms of reconstruction percentage) at each iteration.

Figures 4.6 and 4.7 show the reconstructed spike signal using MBCS and TBCS, respectively, by introducing the spike signal with a similarity of 75%. The reconstruction percentage using TBCS is 92.7% while it is 57.5% using MBCS. The comparison of the two figures shows that TBCS can recover most of the original signal while MBCS fails to reconstruct the signal with so few measurements ($M=62$) in spite of using a high similarity signal as *a priori* information.

Figures 4.8, 4.9, and 4.10 show when transferred signals have a similarity of 25%, 50%, and 75%, respectively, how much signal reconstruction percentage can be achieved at each iteration using the BCS, MBCS, and TBCS algorithms. The simulations are run 100 times, over which the results are averaged. It is clear that our proposed TBCS is much better than the BCS at each iteration. Particularly, when the similarity is 25%, MBCS is worse than BCS while our TBCS achieves higher performance at each iteration than BCS. For instance, at iteration 25 in Figure 4.8, TBCS can achieve a reconstruction percentage of 61.7%, while BCS can reach 42.2% and MBCS only recovers 35.6%. It shows that, at a low similarity, our TBCS can still achieve good performance at every iteration, compared with MBCS and BCS. Moreover, with a high similarity, the performance gap between TBCS and MBCS is enlarged at each step. For example, at iteration 21 with a similarity of 25% in Figure 4.8, TBCS can achieve a reconstruction percentage of 59.7% while MBCS can reach 28.2%. Hence, the performance gap is 31.5%. When the similarity is 75% in Figure 4.10, the performance gap is increased to 50.9% because TBCS can reach 80.5% while MBCS achieves 29.6% at the 21st iteration.

4.6.2 UWB Signal

The tested scenarios are the experimental UWB echo pulses from various UWB propagation channels in practical indoor residential, office and clean, line-of-sight

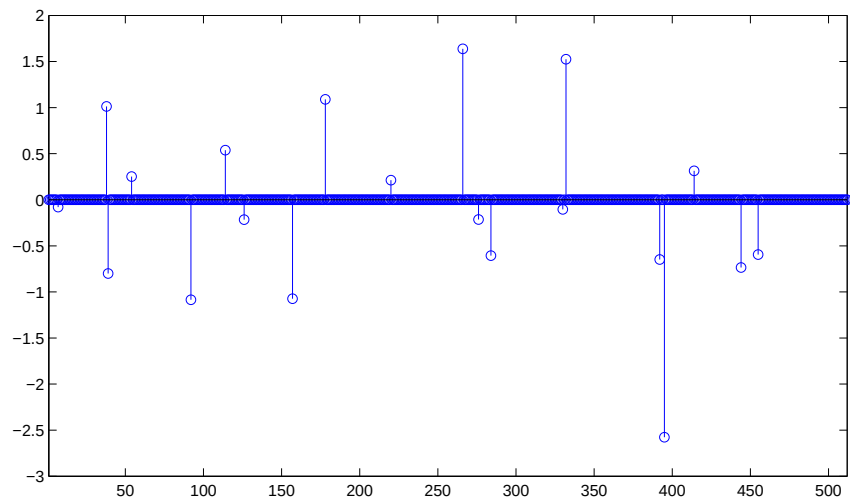


Figure 4.5: Spike signal with 20 nonzero elements in random locations

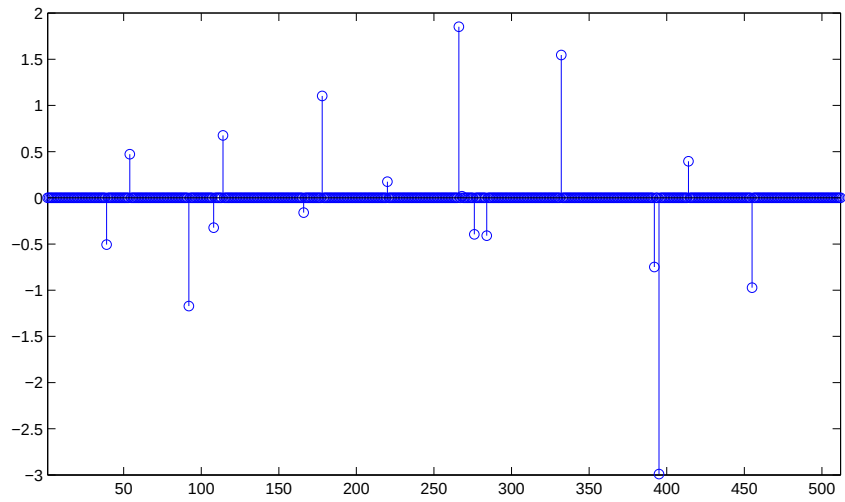


Figure 4.6: Reconstructed spike signal using MBCS with 75% similarity

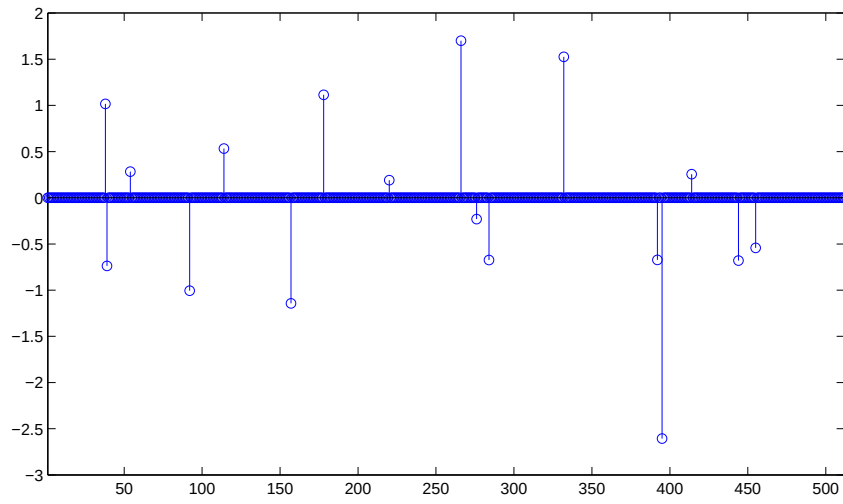


Figure 4.7: Reconstructed spike signal using TBCS with 75% similarity

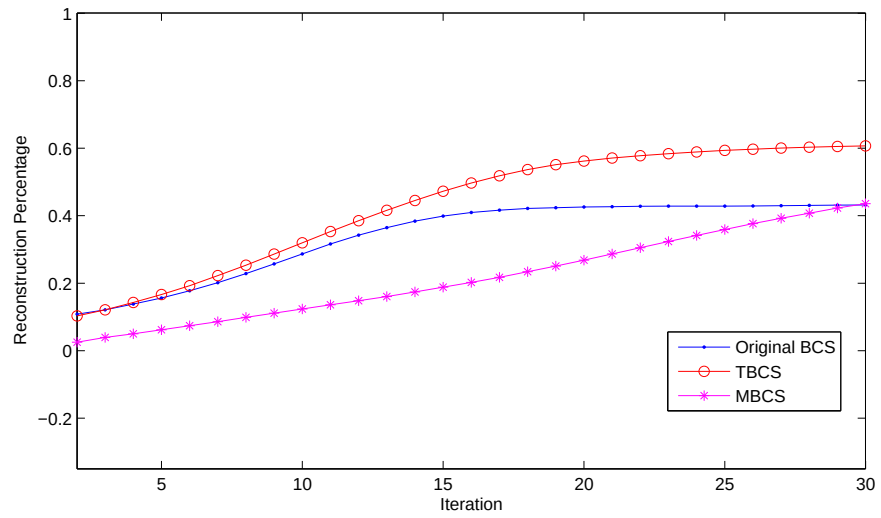


Figure 4.8: Performance gain in each iteration with 25% similarity

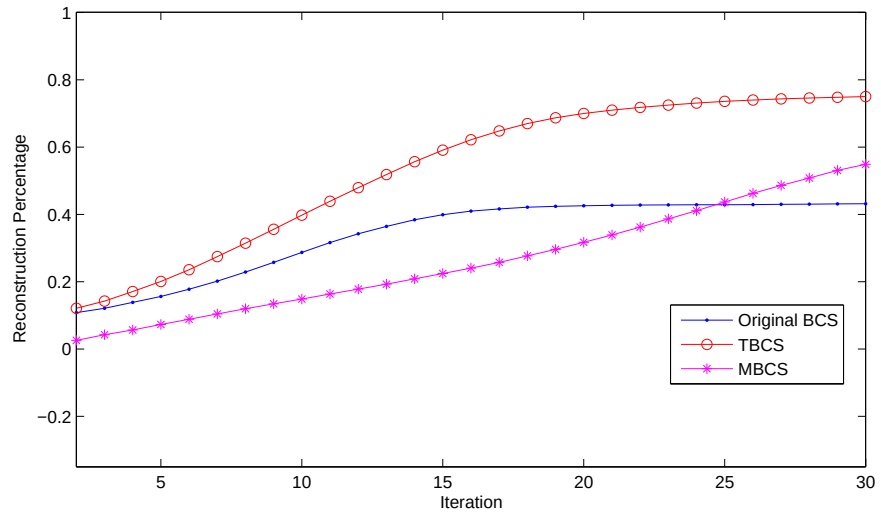


Figure 4.9: Performance gain in each iteration with 50% similarity

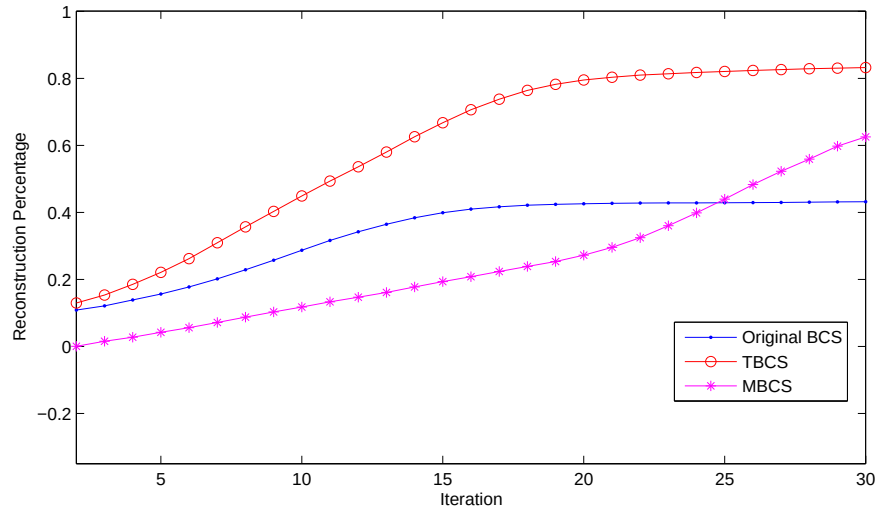


Figure 4.10: Performance gain in each iteration with 75% similarity

(LOS) and non-line-of-sight (NLOS) environments, which are drawn from experimental IEEE 802.15.4a UWB propagation models [1]. In a typical UWB communication/positioning system where receivers are distributed in the same environment, the received UWB echo signals are more or less similar. We test the performance of the original BCS, TBCS, and MBCS algorithms with different similarity levels.

Fig 4.11 shows the reconstructed UWB echo signals using the original BCS and our TBCS algorithms. The test UWB echo signals S_0 (not shown in Fig.4.11), S_1 , S_2 , S_3 , and S_4 are drawn from the IEEE802.15 UWB propagation model [1], in which the reconstructed S_0 is transferred to the other four signal scenarios. The UWB echo signals S_1 , S_2 , S_3 , and S_4 with length $N=512$ are reconstructed using the BCS and TBCS algorithms but only a section (length=150) is shown. In the TBCS algorithm, the reconstructed signal S_0 (not shown) is transferred to other signal reconstruction procedures. The number of measurements, SNR, similarity, and reconstruction percentage are: (a)(b) measurements $M=60$; SNR=9.2dB; with respect to S_0 , the similarity in S_1 is 11.5%; the reconstruction percentages of S_1 using BCS and TBCS algorithms are 81.2% and 84.4%, respectively. (c)(d) $M=60$, SNR=17.7dB; with respect to S_0 , similarity in S_2 is 31.3%; the reconstruction percentages are 46.4% and 89.7%. (e)(f) $M=50$, SNR=12.4dB; 61.0% similarity; the reconstruction percentages are 14.9% and 92.8%. (g)(h) $M=70$, SNR=15.1dB; 98.1% similarity; the reconstruction percentages are 77.0% and 93.2%. With respect to S_0 , the similarity levels in S_1 , S_2 , S_3 , and S_4 are 11.5%, 31.3%, 61.0%, and 98.1%, respectively. For each signal, both algorithms utilize the same number of measurements with the same SNR level for reconstruction. For clarity, only a portion of the UWB signal scenario is expanded to illustrate the waveform details of the reconstructed pulses. It is clearly observed from Figure 4.11 that our TBCS is much better than the original BCS for different similarity levels. The reconstruction percentages using TBCS are much higher than using original BCS by introducing prior information with the same number of measurements. Moreover, the performance gap is increasing

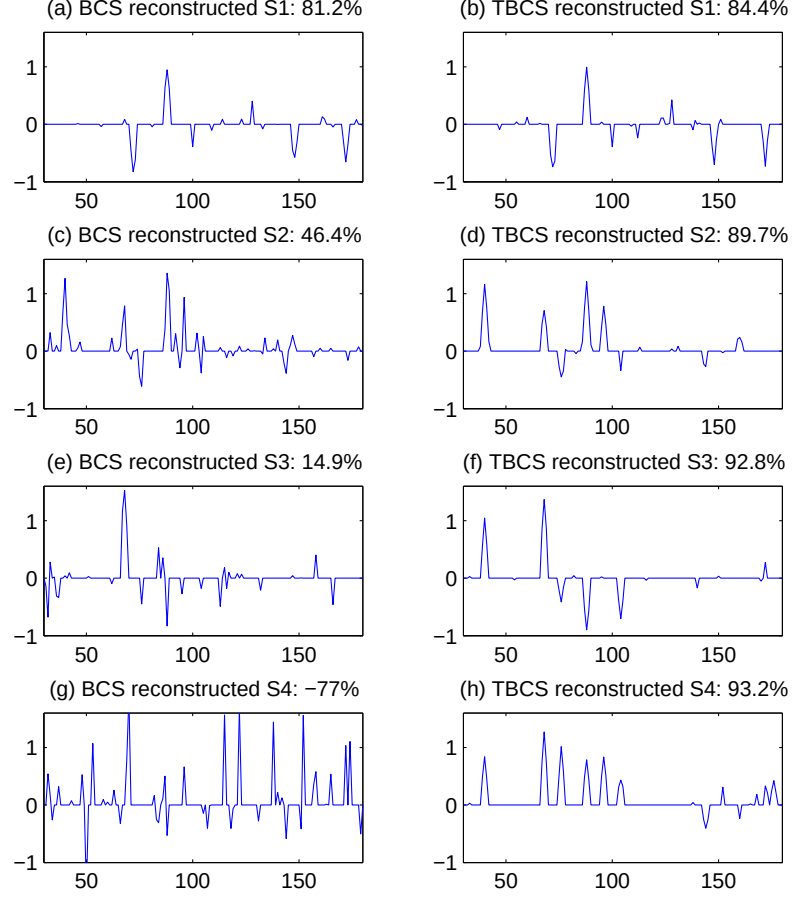


Figure 4.11: Performance of original BCS and TBCS

with the growth of the similarity level. For instance, with a similarity of 11.5% for reconstructing the signal $S1$ in Figure 4.11 (a)(b), the difference of reconstruction percentages using BCS and TBCS is only 3.2% (84.4%-81.2%). When the similarity level is 98.1% for reconstructing the signal $S4$ in Figure 4.11 (g)(h), the difference is increased to 170.2% (93.2%-(-77%)). Therefore, with a higher similarity level, higher performance gain can be achieved.

The performance of the original BCS, MBCS, and TBCS at different similarity levels are then compared. We select three UWB echo signals $S5$, $S6$, and $S7$ with the same dimension $N = 512$. The additive noise variance is only 0.01, implying a very high SNR. The reconstructed signals $S6$ and $S7$ as *a priori* information are

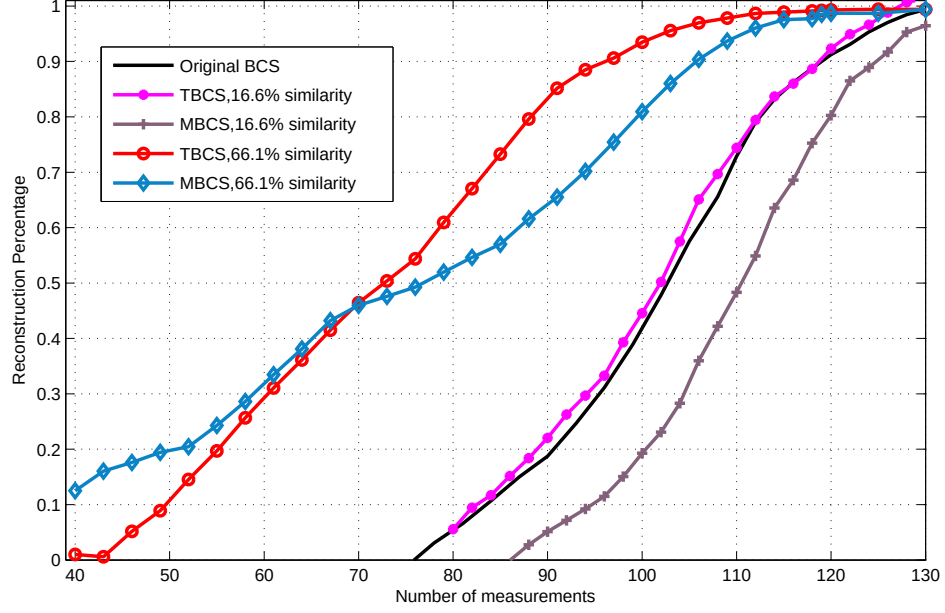


Figure 4.12: Performance comparison at different similarity levels without noise.

transferred to the signal reconstruction for $S5$. With respect to $S6$ and $S7$, the similarities in $S5$ are 16.3% and 64.4%, respectively. The signal $S5$ is recovered with different numbers of measurements using the original BCS, TBCS, and MBCS algorithms. Figure 4.12 shows the reconstruction percentages versus the number of measurements for the signal $S5$. Obviously, at a low similarity level, the MBCS performance is substantially worse than the original BCS whereas our TBCS achieves a performance equaling that of the original BCS performance. For a high similarity level, both MBCS and TBCS are much better than the original BCS due to the benefits of high similarity transferred from the signal $S7$. This demonstrates that our TBCS achieves a good balance between local observations and *a priori* information, leading to a more robust performance than the MBCS.

In the presence of more noise interference, our TBCS still outperforms MBCS and BCS, as shown in Figure 5.6. We use the same signals $S5$, $S6$, and $S7$ but the noise variance is increased to 0.4. We observe that our TBCS exhibits good performance as shown in Figure 4.12. Particularly in the presence of noise, when the number of measurements is large enough ($M > 150$). At a low similarity level, the MBCS can

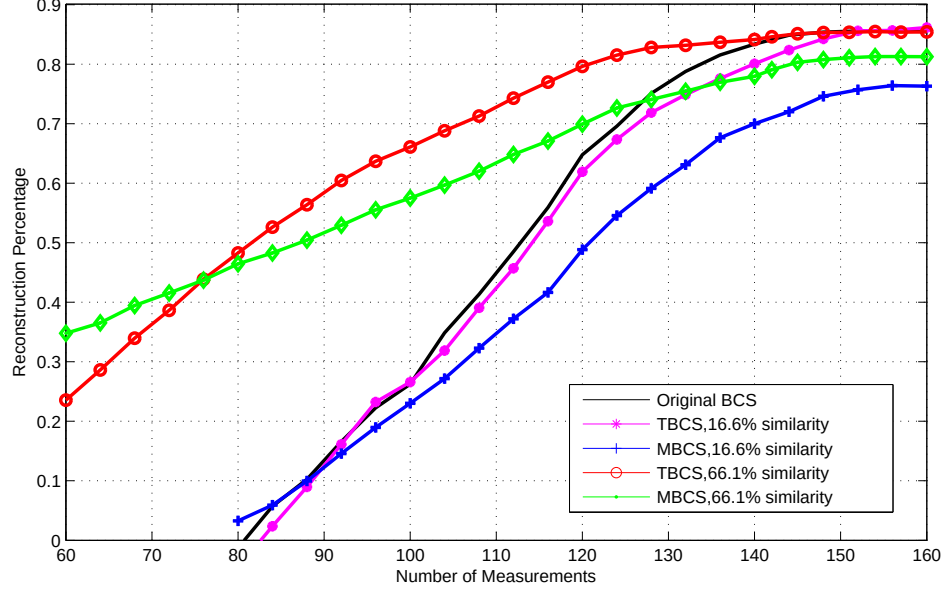


Figure 4.13: Performance comparison at different similarity levels in the presence of noise

achieve a maximum reconstruction percentage of 74.5% while our TBCS algorithm is able to accomplish a maximum reconstruction percentage of 86.9%. At a high similarity level, MBCS can reach a maximum of 80.1% while our TBCS algorithm is still able to accomplish a maximum of 86.9%. Therefore, by introducing prior information, the proposed TBCS algorithm can significantly reduce the number of measurements and improve the capability of combating noise.

Fig. 4.14 shows the Bit Error Rate (BER) for an example UWB communication system using different algorithms. We utilize Binary Phase Shift Keying (BPSK) modulation to transfer the data since bi-phase modulation is one of the easiest methods to implement. The performance of the TBCS, MBCS, and the original BCS algorithms are compared for the UWB communication system. The BER is tested using different noise levels with the same number of measurements ($M = 112$). With so few measurements, using the BCS algorithm leads to a high BER at different SNR. It is also observed that, at a low similarity level, the TBCS performance is much better than the MBCS algorithm. At a high similarity level, the BER performance using the TBCS and MBCS algorithms are much better than using the original BCS

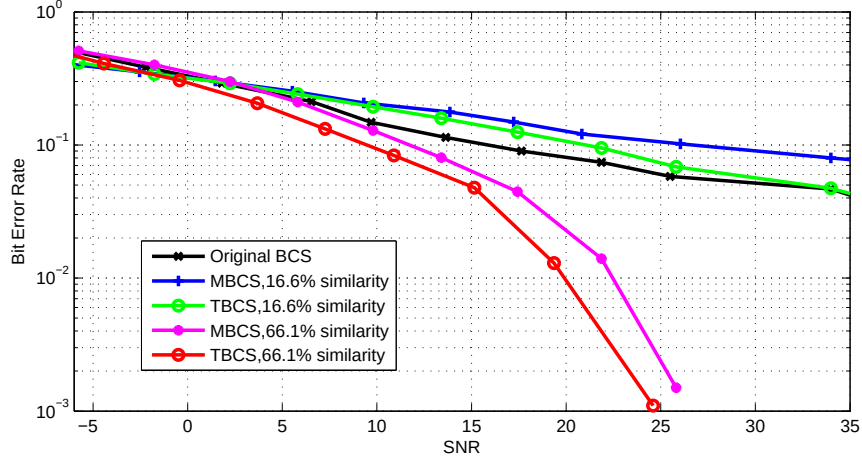


Figure 4.14: BER performance using different algorithms

algorithm, while TBCS is the best. Therefore, by applying our TBCS algorithm in the UWB communication system, it can reduce the BER, provide more tolerance of the noise, and thus achieve the best performance when compared with the MBCS and BCS algorithms.

We presented a Turbo Bayesian CS algorithm in this chapter for general joint signal reconstruction. In the next chapter, the developed TBCS algorithm will be applied to the UWB positioning system to reduce sampling rates for high positioning accuracy.

Procedure 1 Space-Time Turbo Bayesian Compressed Sensing Algorithm

- 1: The hyperparameter α is set to $\alpha = [\infty, \dots, \infty]$.
The candidate index set $\Omega = \emptyset$.
The noise is initialized to a certain value without any prior information, or utilize the previous estimated value.
The parameter of the hyperparameter $\lambda : \lambda = [0, \dots, 0]$;
 - 2: Update λ using Eq.(4.12) and (4.14) from the previous reconstructed nonzero signal elements. This introduces temporal prior information.
 - 3: **repeat**
 - 4: Check and receive the ongoing reconstructed signal elements from other simultaneous BCS reconstruction procedures to update the parameter. λ ; this is to fuse spatial prior information.
 - 5: Choose a random j -th index; Calculate the corresponding parameter g_j and h_j as shown in Eq.(A.22) and (A.23).
 - 6: **if** $(g_j)^2 > h_j$ and $\tilde{\lambda}_j \neq 0$ **then**
 - 7: Add a candidate index: $\Omega = \Omega \cup j$;
 - 8: Update α_j by solving Eq.(4.27).
 - 9: **else**
 - 10: **if** $(g_j)^2 > h_j$ and $\tilde{\lambda}_j = 0$ **then**
 - 11: Add a candidate index: $\Omega = \Omega \cup j$
 - 12: Update α_i using Eq.(4.26).
 - 13: **else if** $(g_j)^2 < h_j$ **then**
 - 14: Delete the candidate index: $\Omega = \Omega \setminus \{j\}$ if the index is in the candidate set.
 - 15: **end if**
 - 16: **end if**
 - 17: Compute the signal coefficients $\mathbf{s}_\Omega$ in the candidate set using Eq.(4.6).
 - 18: Send out the ongoing reconstructed signal elements $\mathbf{s}_\Omega$ to other BCS procedures as spatial prior information.
 - 19: **until** converged
 - 20: Re-estimate the noise level using Eq.(4.31) and send out the noise level for the next usage.
 - 21: Send out the reconstructed nonzero signal elements for the next time utilization as temporal prior information.
 - 22: Return the reconstructed signal.
-

Chapter 5

Applications of Joint BCS Algorithm

5.1 Background and Motivation

Ultra-wide band (UWB) technology, taking advantage of large transmission bandwidth, low power consumption, and simple transceiver architecture, has been widely applied in many applications, such as UWB communications, wall penetrating radar [13] and vehicle positioning. Particularly, due to the extremely short duration of the transmitted pulse, the UWB pulse can be utilized for indoor high precise positioning navigation systems. The ultra-short pulse, typically on the order of nano- or subnano-second width, can provide very precise timing information (e.g., pico-second level) for a millimeter (mm) or sub-mm precision positioning system. On one hand, the duration of the pulse is inversely proportional to the positioning accuracy: the shorter the duration of the pulse is, the higher positioning precision can be achieved [13][45]. On the other hand, it brings a primary challenge of acquiring the high resolution narrow duration pulse. For instance, in order to realize a millimeter

accuracy UWB positioning system, we should have the capability to acquire pico-second level timing resolution*. However, such an ultra-high sampling rate ADC is either extremely expensive or commercially unavailable, which is the main problem for UWB positioning systems to achieve an ultra high position accuracy.

In order to alleviate the sampling problem, a sequential sub-sampler is utilized [13] to acquire ultra-high timing resolution UWB pulses by using a low sampling rate ADC. However, such a sequential sub-sampler suffers from many problems. One is that the speed is very slow, so it is not suitable for tracking fast moving targets. The positioning accuracy is seriously decreased when the target is moving. The asynchronous clocks between the tag and the receivers prevents the UWB positioning system from achieving a higher accuracy. In order to achieve an ultra-high positioning accuracy in high speed, we seek to find an approach for real-time ultra-high speed sampling with low sampling rate ADCs. CS theory [27] has been applied to UWB systems acquiring high resolution pulses below the Nyquist sampling rate [30] [76][20][12][19][15]. However, we seek to develop a joint CS reconstruction algorithm for UWB positioning systems.

A typical CS-based UWB positioning system is shown in the previous chapter in Fig. 4.1. A moving UWB pulse transmitter, called *the tag*, periodically sends out ultra-short duration pulses (about 300-ps duration) at a certain frequency. Surrounding the tag, multiple receivers, called *base stations* (BS), are responsible for receiving the transmitted pulses. The received UWB signals are compressed using some analog circuits, such as transversal filters in [20], to yield measurements by mixing the original signal. The measurements are acquired and digitized using a low sampling rate ADC. From the measurements, the high resolution UWB signals at different base stations can be jointly reconstructed using our proposed CS reconstruction algorithm. From the reconstructed signal, the pulse arrival time information is obtained. We connect and synchronize clocks at base stations through

*Assuming the UWB pulse propagation speed is the speed of light, i.e., $c = 3 \times 10^8 m/s$. Then the time delay for UWB pulses propagating 1mm is about 3ps. ($\frac{1mm}{3 \times 10^8(m/s)} \approx 3ps$)

wires such that the time differences of pulses arriving at different base stations reflect the geometrical difference. Hence, the position of the tag can be calculated using the Time Difference Of Arrival (TDOA) algorithm.

In this chapter, we propose a novel front-end scheme for a high precision CS-based UWB positioning system. Focusing on the properties of the UWB positioning system, we developed a joint Bayesian Compressed Sensing (BCS) signal reconstruction algorithm, which is tightly integrated with the TDOA algorithm for fast tag tracking. The key idea in the proposed scheme is to *utilize the spatial and temporal redundancies existing in received UWB signals among base stations*. In one base station, the received UWB echo signals are similar in time because the tag moves very slowly compared with the pulse repetition frequency, which results in the temporal redundancy. Among different base stations, the received UWB echo signals are also similar, which yields spatial redundancy.

At the end of this chapter, numerical simulation results investigate the performance of the proposed STBCS algorithm compared with the traditional OMP, BCS, and MBCS algorithms using UWB echo signals drawn from the IEEE802.11b UWB standards [1]. With a few measurements, our STBCS algorithm can achieve a good reconstruction percentage while other algorithms fail to reconstruct signals. Also our STBCS algorithm has the best ability to combat noise. Moreover, in the CS-based UWB positioning system, the STBCS-TDOA algorithm is able to calculate the tag position at each iteration before the signal is fully reconstructed. Finally, simulation shows that UWB positioning systems using our proposed STBCS-TDOA algorithm can achieve much higher accuracy than the traditional sequential sampling method.

The remainder of this chapter is organized as follows. The signal model of the CS-based UWB positioning system is formulated and introduced in Section 5.2. The STBCS algorithm is detailed in Section 5.3 for joint BCS signal reconstruction procedures. In Section 5.4, the STBCS-TDOA algorithm for integrating STBCS with TDOA for UWB positioning systems is discussed. Simulation results in Section 6.6

demonstrate the performance of the proposed STBCS-TDOA algorithms compared with other CS algorithms and UWB positioning systems.

5.2 UWB Positioning System Model

5.2.1 UWB Signal Model for Positioning

The UWB pulse is periodically transmitted from the tag. After propagating through multi-path UWB channels, the signal, $u_i(t)$, received at the i -th base station in the continuous time domain is given by

$$u_i(t) = \sum_{m=1}^M a_m p'(t - t_{im}), \quad (5.1)$$

where we denote by $p(t)$ the transmitted Gaussian pulse, $p'(t)$ the received distorted pulse resulted from frequency-dependent propagation channels, M the number of resolvable propagation paths, a_m the amplitude attenuation of the signal along the m -th path, and t_{im} the time delay of the m -th path at the i -th base station.

We assume the UWB positioning system is in a clean line-of-sight (LOS) propagation environment. In such an environment, the first arriving pulse is always the transmitted pulse which goes through the shortest propagation path, having the largest amplitude and power and indicating the pulse arrival time [1][13][69]. We assume that the signal element with the maximum amplitude represents the UWB pulse peak. Also, the time location of the pulse peak indicates the pulse arrival time. Therefore, we need to detect the time index of the element with the maximum amplitude in order to determine the arrival time of the pulse for localizing the position of the tag.

In order to handle the asynchronous problem between the tag and base stations, we calculate the time difference of the pulse arriving at different base stations. Among the base stations, the time difference τ_{ji} is obtained when the pulses arrive at the j -th

and i -th base stations, which is given by

$$\tau_{ji} = (t_{j1} - t_{i1}), \quad (5.2)$$

where t_{j1} represents the time when the first pulse arrivals at the j -th base station. In a clean LOS environment, the first arrival pulse always has the maximum amplitude due to the shortest propagation path. t_{j1} can be regarded as the time index of the signal element with the maximum amplitude in the digital domain. Similarly, t_{i1} represents the first pulse arrival time at the i -th base station. The time difference τ_{ji} implies the distance difference information. When multiple time differences are collected from the received UWB signals at several base stations, the position of the tag can be calculated and the asynchronous problem is removed by using the TDOA algorithm [13][45].

Obviously, the key to precision in UWB positioning systems is to obtain accurate pulse arrival times. In order to acquire the fine timing information, we apply CS theory in the UWB positioning system for acquiring high resolution UWB signals using low sampling rate ADCs.

5.2.2 Compressed Sensing UWB Signal Acquisition

The CS-based UWB positioning system is illustrated in Fig. 5.1. Since the pulse repetition frequency is fixed, we define one pulse repetition period as one *frame*. At the k -th frame and in the i -th BS, the received signal u^{ik} is firstly compressed by using a hardware projection matrix Φ^i , ($\Phi^i \in R^{M \times N}$). A possible analog hardware implementation scheme of the projection matrix can be found in [20]. The measurement vector y^{ik} is given by

$$y^{ik} = \Phi^i u^{ik} + \epsilon^{ik}, \quad (5.3)$$

where ϵ^{ik} is additive white Gaussian noise with unknown but stationary power β^{ik} . By assuming that β^{ik} is identical for all i and k , we abbreviate β^{ik} into β for convenience. In a certain time interval, the analog measurement can be obtained by the subsequent ADC, which forms an M -dimensional measurement vector $\mathbf{v}^{ik}$. Then the N -dimensional sparse UWB signal $\mathbf{u}^{ik}$ at the i -th BS can be recovered from the M -dimensional measurement vector $\mathbf{y}^{ik}$ using CS signal reconstruction algorithms. Since the UWB signal $\mathbf{u}^{ik}$ is inherently sparse, we have $M \ll N$ so that the sampling rate to obtain the measurements $\mathbf{y}^{ik}$ is much slower than that to obtain the signal $\mathbf{u}^{ik}$ directly. The sampling rate to obtain measurements S_M is given by

$$S_M = \frac{M}{N} \times \frac{1}{\Delta_N} = \frac{M}{N} \times S_N = C_r \times S_N, \quad (5.4)$$

where Δ_N is the time resolution of the reconstructed N -dimensional signal vector, which is the inverse of the sampling rate S_N for direct signal acquisition; and C_r is the compression ratio, $C_r = \frac{M}{N} = \frac{S_M}{S_N}$. The compressed ratio represents the reduction of the sampling rate by applying the CS theory compared with direct signal acquisition.

In UWB positioning systems, the received signals are correlated, which provides spatial and temporal redundancies. Without loss of generality, we do not specify the correlation model because the correlation varies for UWB positioning systems in different propagation environments. Based on many experimental observations, we found the received signals at different frames and different base stations many nonzero signal elements share the same time locations[13][45]. The correlation is measured by the similarity level, which is defined as

$$P_s = \frac{K_{com}}{K_{obj}} \quad (5.5)$$

where K_{obj} represents the total number of nonzero signal elements in the objective unrecovered signal, K_{com} is the number of common nonzero element locations among the transferred reconstructed signals and objective signal, and P_s means the similarity

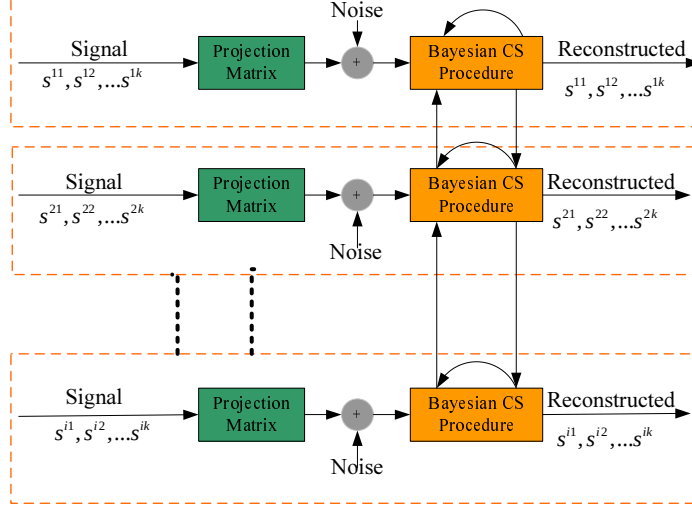


Figure 5.1: Compressed sensing UWB positioning system

level as a percentage. Note that, when $P_s = 100\%$ it does not mean that the signals are the same but that they have the same nonzero element locations; the amplitudes may not be the same.

For notational simplicity, we utilize a superscript to represent temporal and spatial indices and a subscript to denote the element index in the vector. We denote $\mathbf{u}^{ik}$ as the received signal vector at the i -th base station and the k -th time frame. It is abbreviated as $\mathbf{u}$ without considering the spatial and temporal index for representing a single vector. u_j^{ik} is the j -th signal element in the vector $\mathbf{u}^{ik}$, which has $\mathbf{u}^{ik} = \{u_j^{ik}\}_{j=1}^N$. This notation is also applied into other letters. $\mathbf{y}, \mathbf{y}^{ik}, y_j^{ik}$ are abbreviated as $\mathbf{y}$. Similar notation is used for the Φ and α .

5.3 Space-Time Bayesian Compressed Sensing

In this section, we develop a joint signal reconstruction algorithm, namely the STBCS algorithm, to exploit and integrate space-time prior information for the CS-based UWB positioning system. The framework of the BCS algorithm is first introduced. We then detail how to utilize the gamma prior imposed on the exponentially distributed hyperparameters to yield prior information, which is the key in our STBCS

algorithm. The transferred reconstructed signal elements will impact the parameters which control the exponential distribution of the hyperparameters α , yielding prior information. Next, a flexible structure to transfer mutual spatial and temporal prior information is explained in detail. Our STBCS algorithm can be utilized for exploiting and integrating spatial, temporal, and space-time information *a priori* for the best performance.

5.3.1 Bayesian Compressed Sensing

The key of the BCS [59] and relevance vector machine (RVM) algorithms [39] is to impose an exponentially distributed hyperparameter on each signal element. The BCS theory and algorithms have been detailed in previous chapters and ignored here.

5.3.2 Transfer Prior Information

In this chapter, we are using the Gamma prior [59], which is imposed on the hyperparameter α^{ik} with parameters a^{ik} and b^{ik} . By assuming independent distribution, we have the gamma prior distribution, which is given by

$$\begin{aligned}
& P(\alpha^{ik} | a^{ik}, b^{ik}) \\
&= \prod_{j=1}^N \Gamma(\alpha_j^{ik} | a_j^{ik}, b_j^{ik}) \\
&= \prod_{j=1}^N \frac{(b_j^{ik})^{a_j^{ik}} (\alpha_j^{ik})^{(a_j^{ik}-1)} \exp(-b_j^{ik} \alpha_j^{ik})}{\Gamma(a_j^{ik})}, \tag{5.6}
\end{aligned}$$

where a_j^{ik} and b_j^{ik} are the parameters controlling the hyperparameter α_j^{ik} distribution for the j -th signal element in the i -th BCS procedure at the k -th frame.

In order to bridge the mutual information transfer among different BCS procedures, we assume that, for different i and k they share the same hyperparameter, which has

Assumption 1. $\forall i, k$, assume $\alpha_j^{ik} = \alpha_j$.

Now assume we have the j -th reconstructed signal element, e.g., u_j^{cd} from the c -th BCS procedure at the d -th frame. The problem is how to transfer prior information through updating the parameters a_j^{ik} and b_j^{ik} based on the signal element u_j^{cd} .

Based on the above assumption, we can integrate the hyperparameter α_j out:

$$\begin{aligned}
& P(u_j^{cd}|a_j^{ik}, b_j^{ik}) \\
&= \int P(u_j^{cd}|\alpha_j)P(\alpha_j|a_j^{ik}, b_j^{ik})d\alpha_j \\
&= \frac{(b_j^{ik})^{a_j^{ik}}\Gamma(a_j^{ik} + \frac{1}{2})}{(2\pi)^{\frac{1}{2}}\Gamma(a_j^{ik})} \left(b_j^{ik} + \frac{(u_j^{cd})^2}{2} \right)^{-(a_j^{ik} + \frac{1}{2})}, \tag{5.7}
\end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. The derivation is detailed in Appendix A. In particular, when $a_j^{ik} = 2b_j^{ik}$, it becomes a student-t distribution where a_j^{ik} is the number of degrees of freedom. Here, the parameters a_j^{ik}, b_j^{ik} play the key to transferring prior information. The transferred information will change the parameters a_j^{ik}, b_j^{ik} , which will impact the corresponding hyperparameter and the signal element.

Given the transferred signal element u_j^{cd} , we can update the parameters a_j^{ik} and b_j^{ik} to bring prior information in the i -th BCS procedure. We have

$$\hat{a}_j^{ik} = \arg \max_{a_j^{ik}} \prod_{c,d=1}^n P(u_j^{cd}|a_j^{ik}, b_j^{ik}) \tag{5.8}$$

$$\hat{b}_j^{ik} = \arg \max_{b_j^{ik}} \prod_{c,d=1}^n P(u_j^{cd}|a_j^{ik}, b_j^{ik}). \tag{5.9}$$

However, it is not easy to estimate the parameters by maximizing the likelihood function. Alternatively, based on Bayes theory, we utilize the posterior to update the

parameters, which is given by

$$\begin{aligned}
& P(\alpha_j | u_j^{cd}, a_j^{ik}, b_j^{ik}) \tag{5.10} \\
&= \frac{P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{\int P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik}) d\alpha_j} \\
&= \frac{\left(\tilde{b}_j^{ik}\right)^{\tilde{a}_j^{ik}} (\alpha_j)^{(\tilde{a}_j^{ik}-1)} \exp\left(-\tilde{b}_j^{ik} \alpha_j\right)}{\Gamma(\tilde{a}_j^{ik})}. \tag{5.11}
\end{aligned}$$

By comparing Eq. (A.11) with Eq. (5.6), we have the updated parameters, which are given by

$$\tilde{a}_j^{ik} = a_j^{ik} + \frac{1}{2} \tag{5.12}$$

$$\tilde{b}_j^{ik} = b_j^{ik} + \frac{(u_j^{cd})^2}{2}. \tag{5.13}$$

Based on the signal element u_j^{cd} , the parameters a_j^{ik} and b_j^{ik} are updated to $\tilde{a}_j^{ik}$ and $\tilde{b}_j^{ik}$. Obviously, the posterior is still the gamma distribution, which provides convenience for parameter updating. This is the reason why we utilize the gamma priors to transfer mutual prior information.

When multiple reconstructed signal elements are transferred, e.g., u_j^{c1} , u_j^{c2} , and u_j^{cn} , the parameters are updated to

$$\tilde{a}_j^{ik} = a_j^{ik} + \frac{n}{2} \tag{5.14}$$

$$\tilde{b}_j^{ik} = b_j^{ik} + \frac{\sum_{i=1}^n (u_j^{ci})^2}{2}. \tag{5.15}$$

The derivations of above equations are detailed in Appendix A.6.

Therefore, based on the given signal elements transferred from other BCS procedures, parameters are updated to introduce the prior information into the current BCS procedure. Note that, initially, the parameters a_j^{ik} and b_j^{ik} are set to zero, implying no prior information, which is equivalent to the original BCS algorithm.

If and only if the transferred signal elements are nonzero, the parameters will be updated. Furthermore, we develop a space-time structure for exploiting and utilizing prior information.

5.3.3 Space-Time Structure

In the CS-based UWB positioning system, the joint BCS signal reconstruction procedures are performed both in parallel and in serial. In order to exploit spatial, temporal, or space-time prior information, we develop a flexible space-time structure for the STBCS algorithm. At one base station, the temporal prior information can be exploited and utilized for signal reconstruction in the time sequence, which is namely the Time BCS (TBCS) algorithm. At the same time among several base stations, spatial prior information can also be transferred for joint signal reconstruction in parallel, which is the Space BCS (SBCS) algorithm. In CS-based UWB positioning system, our STBCS algorithm can exploit and fuse the space-time prior information for the best performance.

Fig. 5.2 shows the structure to transfer space-time prior information in the STBCS algorithm. For simplicity, we only show three base stations (BS1, BS2, and BS3). Only the BCS signal reconstruction procedures at the k -th frame in each BS are illustrated. On one hand, temporal prior information is transferred by using the TBCS algorithm. In BS1, for example, the BCS procedure is performed for reconstructing the signal $\mathbf{u}^{1k}$ and $\mathbf{u}^{1(k+1)}$ in serial at the k and $(k+1)$ -th frames. Then the reconstructed signal $\mathbf{u}^{1k}$ is forwarded to the next frame to update hyperparameters $\mathbf{a}^{1(k+1)}$ and $\mathbf{b}^{1(k+1)}$ in the $(k+1)$ -th frame. It is the same in BS2 and BS3 by using the TBCS algorithm. On the other hand, mutual spatial prior information is transferred in parallel using the SBCS algorithm. At the k -th frame, BCS procedures for reconstructing $\mathbf{u}^{1k}$, $\mathbf{u}^{2k}$ and $\mathbf{u}^{3k}$ are performed simultaneously. For BS1, the ongoing reconstructed $\mathbf{u}^{1k}$ can be transferred to BS2 and BS3 to update hyperparameters $\mathbf{a}^{2k}$ and $\mathbf{b}^{2k}$, as well as $\mathbf{a}^{3k}$ and $\mathbf{b}^{3k}$. Similarly, the information can

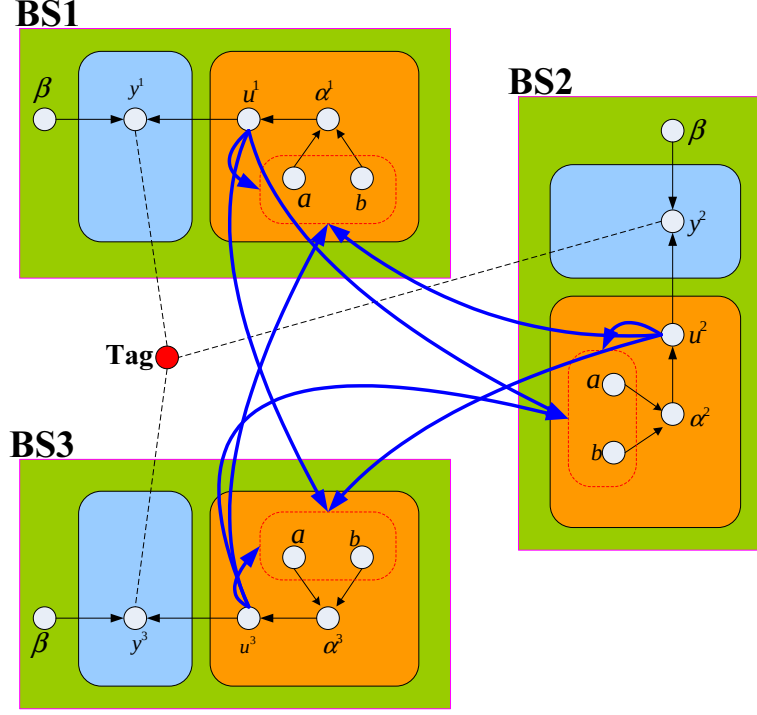


Figure 5.2: Space-time structure for the STBCS algorithm

be also transferred from BS2 to BS3 and BS1, and from BS3 to BS1 and BS2. All together, space-time prior information can be transferred and utilized both in serial and in parallel, which is called the Space-Time BCS (STBCS) algorithm.

5.4 STBCS-TDOA Joint Signal Processing

In this section, we develop a joint STBCS-TDOA algorithm for the CS-based UWB positioning system. The transferred prior information is integrated into the STBCS signal reconstruction procedure. Along with the TDOA algorithm, the iterative STBCS-TDOA algorithm is able to compute the tag position at each iteration even though the received UWB signal is not fully reconstructed. We first discuss how to select the nonzero signal element which is most likely the pulse peak based on the incremental optimization method. Then given the transferred prior information and measurements, we will discuss how to fuse the prior information into the signal

reconstruction procedure. Next, based on the obtained pulse peak location and pulse arrival time, the TDOA algorithm can calculate the position of the tag. Finally, the procedure and structure of the proposed STBCS-TDOA algorithm will be summarized. Since the current signal reconstruction is performed at the i -th base station and the k -th frame, for convenience we abbreviate $\mathbf{u}^{ik}$, $\mathbf{y}^{ik}$, and $\boldsymbol{\alpha}^{ik}$ into $\mathbf{u}$, $\mathbf{y}$, and $\boldsymbol{\alpha}$, respectively.

5.4.1 Estimate the Pulse Peak Location

The incremental optimization method will sequentially find and select one nonzero signal element in each iteration. How can we estimate the largest signal element for determining the pulse arrival time?

The key step of the BCS signal reconstruction algorithm is to estimate the hyperparameters $\mathbf{A}$ by maximizing the marginal log-likelihood function [39][59]. The incremental optimization method separates one index from the target function, e.g., the j -th element, which is given by

$$\begin{aligned}
L(\boldsymbol{\alpha}) &= \log p(\mathbf{y}|\boldsymbol{\alpha}, \beta) \\
&= \log \int P(\mathbf{y}|\mathbf{u}, \beta) P(\mathbf{u}|\boldsymbol{\alpha}) d\mathbf{u} \\
&= -\frac{1}{2}(N \log 2\pi + \log |E| + \mathbf{y}^T E^{-1} \mathbf{y}) \\
&= L_{-j}(\boldsymbol{\alpha}) + l(\alpha_j),
\end{aligned} \tag{5.16}$$

The derivation details are shown in Appendix A. Actually, at the beginning iterations, we have a pool of candidate indices which correspond to possible nonzero signal elements. Based on Appendix A, we define the candidate index set Λ , which is given by

$$\begin{cases} \alpha_j = \frac{h_j^2}{g_j^2 - h_j}, g_j^2 - h_j > 0; & \text{if } j \in \Lambda \\ \alpha_j = \infty, g_j^2 - h_j \leq 0; & \text{if } j \notin \Lambda \end{cases} \tag{5.17}$$

where the details about the terms g_j and h_j are shown in Appendix A.

Then one index in Λ will be selected and added into the nonzero signal index Ω , e.g., the α_j . The signal elements in Ω with the new element will be estimated. The next iteration, we will recalculate parameters $g_j, h_j, j = \{1, 2, \dots, N\}$ and form a new candidate index set Λ . Another new index will be selected from Λ and added into Ω . This process will be repeated until all nonzero signal elements are found after a limited number of iterations.

Traditionally, the index is randomly selected and added from the candidate index set Λ into the nonzero index set Ω each iteration. However, for our BCS-based UWB positioning system, instead of a random choice, we hope to find the nonzero signal element which is most likely the maximum signal element, indicating the peak location. Hence the pulse arrival time can be estimated without waiting until the signal is fully reconstructed. For this purpose, we propose the following Proposition about how to select an index from the candidate index set Λ .

Proposition 1. *During a given iteration, the estimated signal based on the current nonzero signal index Ω is denoted as $\tilde{\mathbf{u}}$. Then the j -th hyperparameter α_j in the current index set Λ which can maximize the marginal log-likelihood function $L(\boldsymbol{\alpha})$ is also able to minimize the term $(\|\mathbf{y}\| - \mathbf{y}^T \Phi \tilde{\mathbf{u}})$, which is given by*

$$\begin{aligned} [j] &= \arg \max_{\alpha_j, j \in \Lambda} (L(\boldsymbol{\alpha})) \\ &= \arg \min_{\alpha_j, j \in \Lambda} (\|\mathbf{y}\| - \mathbf{y}^T \Phi \tilde{\mathbf{u}}). \end{aligned} \quad (5.18)$$

Proof. Proposition 1 bridges between the selected α_j and the signal element u_j by decomposing the log-likelihood function $L(\boldsymbol{\alpha})$. The derivation details are shown in Appendix A. $\square$

Therefore, in each iteration the index in the candidate index set which can maximize the target function $L(\boldsymbol{\alpha})$ will be selected into the nonzero signal index set Ω for signal reconstruction. The selected index and its corresponding signal element most likely has the largest amplitude, which indicates the pulse peak.

Therefore, we can estimate the pulse arrival time even though the signal is not fully reconstructed. Essentially, our STBCS-TDOA algorithm is like a greedy algorithm, where the nonzero signal element which has the most impact on the measurements will be estimated first to minimize the residual. Note that the index selection is only based on local measurements, but the estimation of the hyperparameter α_j with the selected index j should be based on both local measurements and transferred prior information.

5.4.2 Fusing Prior Information

When the selected α_j is estimated, we consider fusing the transferred prior information, if available. Then the target log-likelihood function is given by

$$\begin{aligned} L(\boldsymbol{\alpha}) &= \log P(\mathbf{y}|\boldsymbol{\alpha}) + \log P(\boldsymbol{\alpha}|\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) \\ &= L_{-j}(\boldsymbol{\alpha}) + l(\alpha_j), \end{aligned} \tag{5.19}$$

where the first term $\log P(\mathbf{y}|\boldsymbol{\alpha})$ is only about local measurements. The second term $\log P(\boldsymbol{\alpha}|\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ introduces the transferred prior information with updated parameters $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}$.

In order to estimate the selected α_j , we maximize the posterior function, which is given by

$$l(\alpha_j) = \frac{1}{2} \left(\log \frac{\alpha_j}{\alpha_j + g_j} + \frac{h_j^2}{\alpha_j + g_j} \right) + a_j \log \alpha_j - b_j \alpha_j, \tag{5.20}$$

where the terms g_j and h_j are shown in Appendix A.

The first derivative of Eq. (5.20) is given by

$$\begin{aligned} l'(\alpha_j) &= \frac{g_j}{2\alpha_j(\alpha_j + g_j)} - \frac{h_j^2}{2(\alpha_j + g_j)^2} + \frac{a_j}{\alpha_j} - b_j \\ &= \frac{f(\alpha_j, g_j, h_j, a_j, b_j)}{\alpha_j(\alpha_j + g_j)^2}, \end{aligned} \quad (5.21)$$

where $f(\alpha_j, g_j, h_j, a_j, b_j)$ is a cubic function of the hyperparameter α_j . By setting Eq. (A.29) to zero, we obtain the optimum estimation α_j^* , which is given by

$$\arg \max_{\alpha_j} L(\boldsymbol{\alpha}) = \begin{cases} \alpha_j^*, & \text{if } g_j^2 - h_j > 0; \\ \infty, & \text{otherwise.} \end{cases} \quad (5.22)$$

The detailed derivations are given in Appendix A.8. Actually, the optimal hyperparameter is not only based on the local measurements but also takes advantage of the transferred prior information. Optimizing the posterior function Eq. (5.20) provides a soft way to balance the measurements and transferred prior information for estimating the hyperparameters $\boldsymbol{\alpha}$. Based on the estimated hyperparameters, the signal elements can be reconstructed each iteration. The pulse peak is detected so that the pulse arrival time is obtained. The TDOA algorithm is used for localizing the tag each iteration.

5.4.3 TDOA Algorithm

Let (x_i, y_i, z_i) , $i = 1, 2, \dots, I$ be the known position coordinates of the i -th base station. Let (x_t, y_t, z_t) be the unknown tag location. The distance from the i -th base station to the tag is denoted by D_i . Then, between the 1st and i -th base station, the difference of the pulse arrival time τ_{1i} can be obtained from the received UWB signal using Eq. (5.2). The range difference D_{1i} is given by

$$\tau_{1i} = cD_{1i}, \quad i = 2, 3, \dots, I, \quad (5.23)$$

and

$$D_{1i} = \sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2 + (z_1 - z_t)^2} - \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2 + (z_i - z_t)^2}, \quad (5.24)$$

where c is the propagation speed. Taking the derivative on both sides of the equation, we have

$$dD_{1i} = \frac{(x_1 - x_t)dx_t + (y_1 - y_t)dy_t + (z_1 - z_t)dz_t}{\sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2 + (z_1 - z_t)^2}} - \frac{(x_i - x_t)dx_t + (y_i - y_t)dy_t + (z_i - z_t)dz_t}{\sqrt{(x_i - x_t)^2 + (y_i - y_t)^2 + (z_i - z_t)^2}}. \quad (5.25)$$

For the i -th base station with respect to the 1st base station, the matrix is given by

$$\begin{pmatrix} dD_{12} \\ dD_{13} \\ \vdots \\ dD_{1i} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \vdots & \vdots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \alpha_{i3} \end{pmatrix} \begin{pmatrix} dx_t \\ dy_t \\ dz_t \end{pmatrix}, \quad (5.26)$$

where, we have

$$\alpha_{i1} = \frac{x_1 - x_t}{D_1} - \frac{x_i - x_t}{D_i} \quad (5.27)$$

$$\alpha_{i2} = \frac{y_1 - y_t}{D_1} - \frac{y_i - y_t}{D_i} \quad (5.28)$$

$$\alpha_{i3} = \frac{z_1 - z_t}{D_1} - \frac{z_i - z_t}{D_i}. \quad (5.29)$$

TDOA computation starts with an initial guess position of the tag, $(\hat{x}_t, \hat{y}_t, \hat{z}_t)$. By iteratively updating and solving Eq. (5.26), TDOA will gradually converge to the true position. The computation continues until the error is below a certain threshold,

which is given by

$$E_r = \sqrt{(dx_t)^2 + (dy_t)^2 + (dz_t)^2}. \quad (5.30)$$

Note that the TDOA algorithm relies heavily on the initial position. If the initial guess is far away from the true position, more iterations and computations are needed. Then we develop the STBCS-TDOA algorithm, which is able to output the approximate position of the tag each iteration so that the TDOA algorithm has a good initial position.

5.4.4 Joint STBCS-TDOA Algorithm

The proposed STBCS-TDOA algorithm for our UWB positioning system is described in Procedure 2. The temporal and spatial prior information among different base stations is exploited and utilized in the STBCS algorithm. At each iteration, the information about the pulse arrival time is forwarded to the TDOA algorithm for fast calculation of the tag position.

The STBCS-TDOA algorithm is performed in a parallel pipeline mode for the best performance, as shown in Fig. 5.3. Traditionally, the pulse arrival time cannot be obtained until the UWB echo signal is fully reconstructed. This serial mode needs substantial computation time because both the STBCS signal reconstruction and TDOA algorithms are computationally expensive. Our parallel STBCS-TDOA algorithm is able to calculate the position of the tag each iteration while the UWB signal reconstruction procedures are ongoing. It is based on the fact that the acquisition of the pulse arrival time does not require a full signal recovery. Even though the pulse arrival time is not accurate when the signal is far away from being well reconstructed, the approximate value will help the TDOA converge quickly.

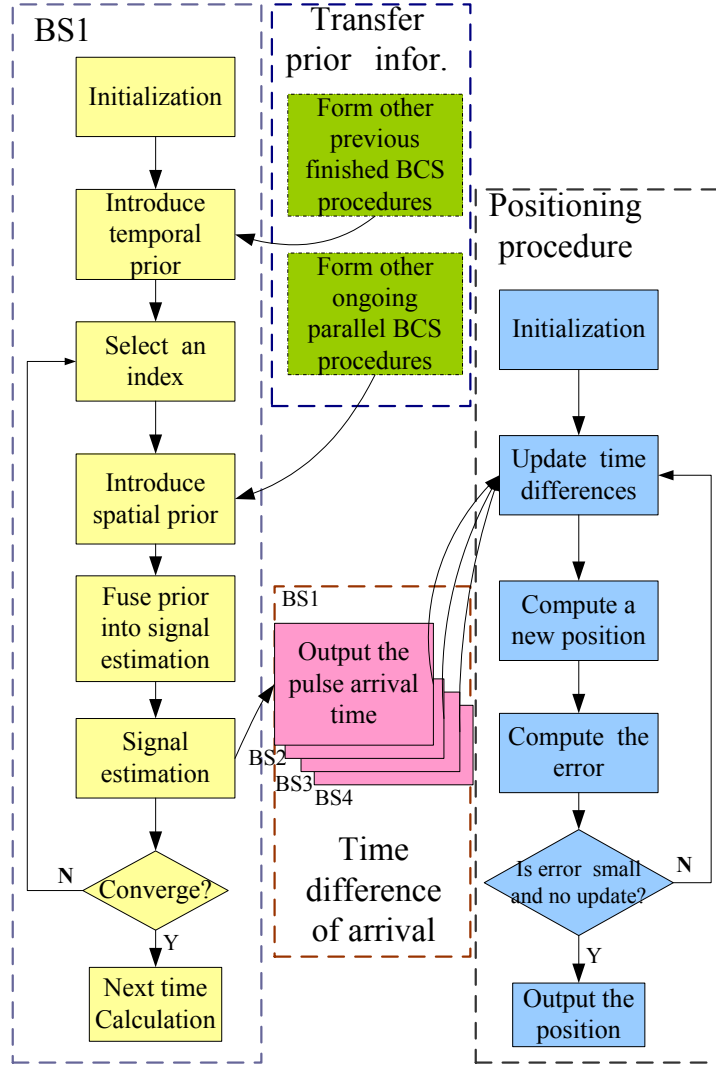


Figure 5.3: Parallel STBCS-TDOA algorithm

5.5 Simulation Results

Numerical simulations are conducted to investigate the performance of the STBCS-TDOA algorithm in the CS-based UWB positioning system. The tested UWB signals are drawn from the experimental IEEE 802.15.4a UWB propagation standards [1]. We first demonstrate the performance of the STBCS algorithm for UWB signal reconstruction with different compression ratios and different noise levels. The positioning performance of using the STBCS, OMP, and BCS algorithms is then compared in the CS-based UWB positioning system. Finally, in a 3D scenario, our CS-based UWB positioning system using the proposed STBCS algorithm is compared with the UWB positioning system using the traditional sequential sampling method. The quality of the reconstructed signal is measured in terms of the reconstruction percentage, which is defined as

$$P_r = 1 - \frac{\|\mathbf{u} - \tilde{\mathbf{u}}\|_2}{\|\mathbf{u}\|_2}, \quad (5.31)$$

where $\mathbf{u}$ is the true signal and $\tilde{\mathbf{u}}$ is the reconstructed signal. Recall that the compression ratio is defined as $C_r = \frac{M}{N} = \frac{S_M}{S_N}$, which means how many sampling rate is reduced for obtaining measurements compared with the sampling rate for direct signal acquisition.

5.5.1 UWB Signal Reconstruction

Fig. 5.4 compares the reconstructed UWB echo signals in the time domain using the original BCS and our STBCS algorithm. Four scenarios of UWB echo signals, U_0 (not shown in Fig. 5.4), U_1 , U_2 , and U_3 are utilized for testing performance, where the reconstructed U_0 is utilized as prior information for UWB signal reconstruction. U_0 , U_1 , U_2 , and U_3 have different similarity levels. For clarity, only a small section ($N=200$) of all the UWB signals ($N=512$) are shown to illustrate the reconstructed waveforms. Four scenarios of UWB signals, U_0 , U_1 , U_2 , and U_3 , are reconstructed

using both algorithms under the same number of measurements and SNR. In detail, (a) and (b) compare the reconstructed signal $U1$ using BCS and STBCS with the same measurements $M = 60, C_r \approx 0.12$ and SNR=9.2dB; with respect to $U0$, the similarity in $U1$ is 31.5%. The reconstruction percentages of $U1$ using the BCS and STBCS algorithms are 84.2% in (a), and 90.4% in (b), respectively. (c) and (d) compare the reconstructed signal $U2$ using BCS and STBCS with $M = 60, C_r \approx 0.12$ and SNR=17.7dB; with respect to $U0$, the similarity in $U2$ is 67.3%. The reconstruction percentages are 68.4% for BCS and 91.7% for STBCS. (e) and (f) compare the reconstructed signal $U3$ with $M = 60, C_r \approx 0.1$, SNR=12.4dB. The similarity level in $U3$ is 87.0%. The reconstruction percentages are -10.9% for BCS and 93.57% for STBCS. Obviously, it is observed that the reconstruction percentages are increasing with the growth of the similarity level. For instance, in (a) and (b) with a similarity of 11.5% for $U1$, the performance gap using BCS and STBCS is only 6.2% (90.4%-84.2%). In (c) and (d), when the similarity level is 98.1%, the performance gap increases to 23.3% (91.7%-(68.4%)). In (e) and (f), it increases to 104.4% (93.5%-(-10.9%)). This because that a higher similarity level implies more useful prior information, which is helpful to improve the performance. Therefore, our STBCS is much better than the original BCS. The higher the similarity level is, the better performance is gained in the STBCS algorithm.

Fig. 5.5 shows the performance comparison of the original BCS, OMP, MBCS, and STBCS algorithms for reconstructing the same UWB signal at different compression ratios. The tested UWB signal is denoted as $U5$ ($N=512$), which will be reconstructed with a different number of measurements under the same level noise with a high SNR ($\text{SNR} \approx 24\text{dB}$). In the original BCS and OMP algorithms, there is no prior information introduced for reconstructing the signal $U5$. We test the performance, in terms of reconstruction percentage, by introducing spatial, temporal, and space-time prior information. The experiment is run 100 times and results are averaged. Since the MBCS algorithm is for parallel multitask signal reconstruction, our spatial BCS (SBCS) algorithm is compared with MBCS by introducing spatial prior information.

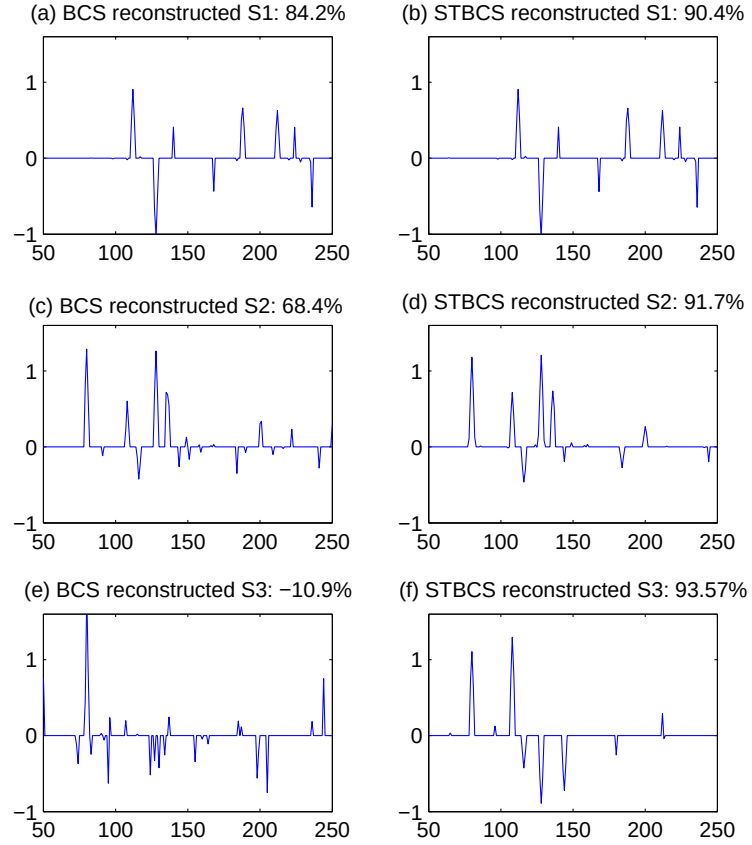


Figure 5.4: Comparison of the reconstructed UWB echo signals, U_1 , U_2 , and U_3 using BCS in (a), (c), and (e) and STBCS in (b), (d), and (f).

In the MBCS and SBCS algorithms, we introduce the UWB signal $U6$ as spatial prior information for recovering $U5$. The signal reconstruction for $U5$ and $U6$ is performed in parallel in both algorithms with the same number of measurements. With respect to $U6$, the similarity in $U5$ is 33.6%. At the same compression ratio, it is observed that both STBCS and MBCS algorithms are much better than the BCS and OMP algorithms. Our STBCS algorithm outperforms the MBCS algorithm at a low similarity level. Also, another reconstructed signal scenario $U7$ is utilized as temporal prior information and transferred into the STBCS algorithm for reconstructing the signal $U5$. The similarity in $U5$ is 66.1% with respect to $U7$. Next, the unreconstructed signal $U6$ and reconstructed signal $U7$ are utilized as space-time prior information and transferred into the STBCS algorithm. The similarity level is increased to 89.1% in $U5$ with respect to $U6$ and $U7$. Clearly, it is observed that performance is increased with the growth of the similarity level. For example, with a low compression ratio $C_r \approx 0.15$ ($M = 80$), our STBCS algorithm through introducing prior information with similarity can achieve a 92.6% signal reconstruction percentage. While at the same compression ratio, the TBCS algorithm can accomplish 21.8% reconstruction percentage. The performance of the SBCS, MBCS, OMP, and BCS algorithms are much worse ($< 8\%$) at such a low compression ratio. Therefore, our STBCS algorithm can significantly reduce the compression ratio and the sampling rate of ADCs by utilizing the space-time prior information for CS-based UWB systems.

In Fig. 5.6, we further test the performance of the algorithms under different noise levels with the same compression ratio ($C_r \approx 0.2, M = 100$) for reconstructing the signal $U5$. The test procedure and other configurations are the same as in Fig. 5.5. It is observed that by introducing prior information, the proposed STBCS algorithm exhibits a much better capability of combating noise compared with the MBCS, OMP, and BCS algorithms. For instance, when SNR=12dB, the STBCS utilizing prior information with a high similarity level (89.1%) can achieve a 93.8% reconstruction percentage while the reconstruction percentage is 21.3% for the OMP algorithm and

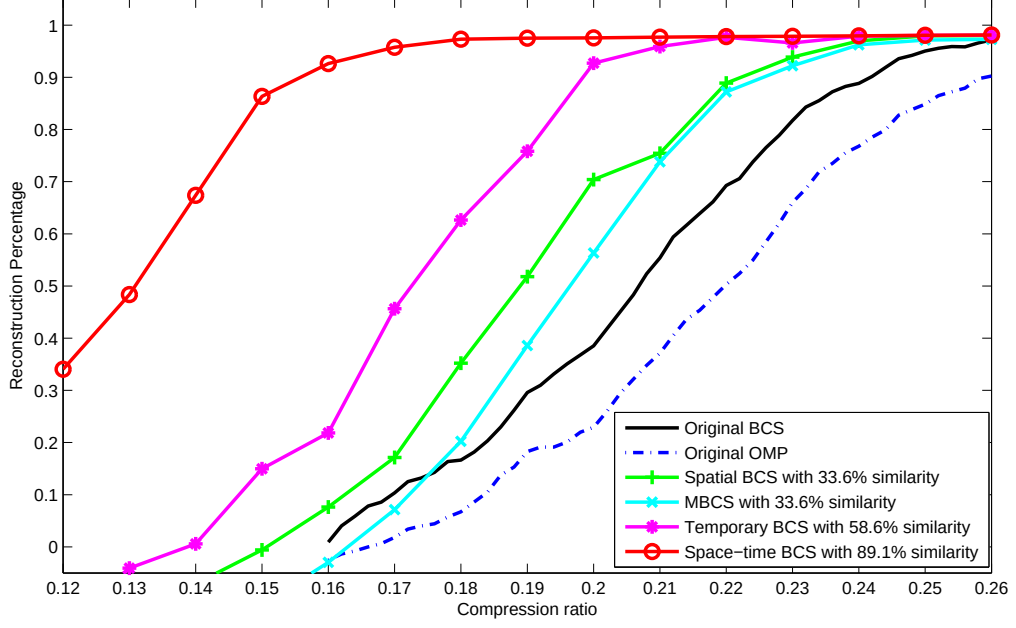


Figure 5.5: Compression ratio versus reconstruction percentage

35.1% for the BCS algorithm at the same noise level and compression ratio. With the same similarity level, our SBCS algorithm outperforms the MBCS algorithm at different noise levels. Therefore, at a low compression ratio with the same number of measurements, by utilizing prior information, our STBCS algorithm can achieve much better performance of combating noise than the MBCS, original BCS, and OMP algorithms.

5.5.2 Joint STBCS-TDOA Performance

We further investigate the positioning performance of the OMP, BCS, and STBCS-TDOA algorithms in the CS-based UWB positioning system.

We utilize the UWB signal $U5$ for testing the positioning performance and other configurations are the same for Fig. 5.5. We first calculate the difference of the pulse peak time index in the reconstructed UWB signal and in the true signal $U5$. Since the pulse peak indicates the pulse arrival time, we measure the positioning error as the product of the propagation speed and difference of pulse arrival time

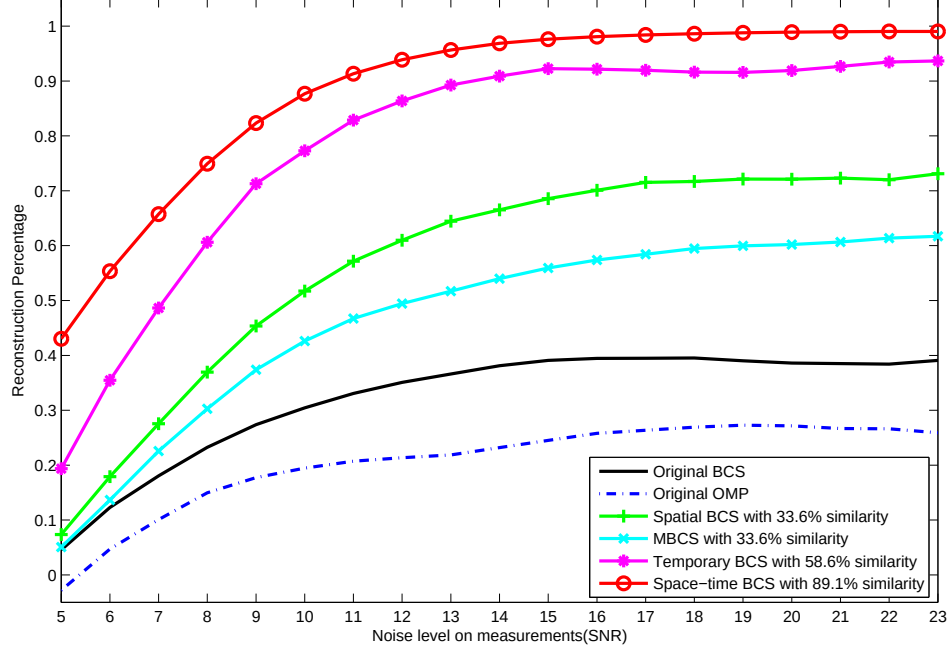


Figure 5.6: Noise versus reconstruction percentage

between the reconstructed signal and the true signal. At each iteration, the iterative OMP, BCS, and STBCS-TDOA algorithms output the reconstructed signal U_5 . Our STBCS-TDOA algorithm will introduce prior information with different similarity levels. The positioning error is measured at each iteration even though the UWB signal is not fully reconstructed. Noise is added ($\text{SNR} \approx 15\text{dB}$) and the compression ratio is low ($C_r \approx 0.2$, $M = 100$) in signal reconstruction. The experiment is run 100 times with different random measurement matrices and results are averaged.

The iterative STBCS algorithm is able to output the peak location, or the arrival time index in each iteration while the signal reconstruction is performing. For clarity, we just measure the 1D error. The error is measured by the offset of the reconstructed UWB signal and the true UWB signal. The error is measured in distance, which is the product of propagation speed (light speed) and pulse arrival time.

Figure 5.7 shows the positioning error at each iteration using the OMP, BCS, and STBCS-TDOA algorithms. The errors from OMP and BCS algorithms are large and fluctuating due to a low compression ratio and a high noise level. In sharp

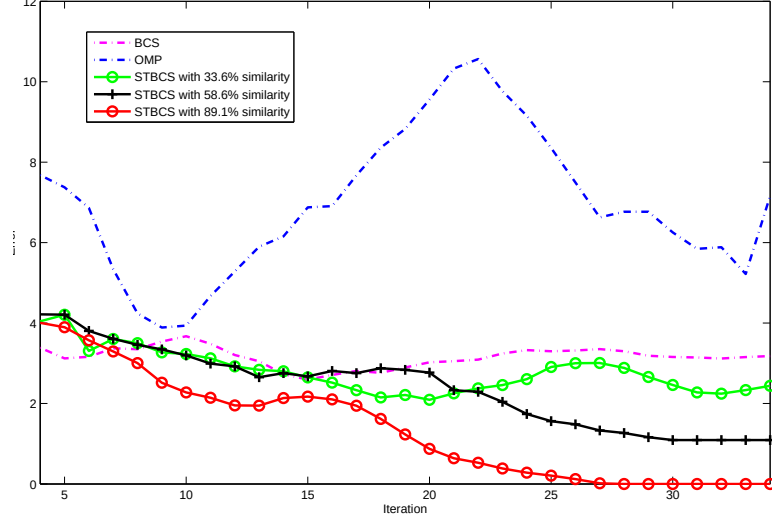
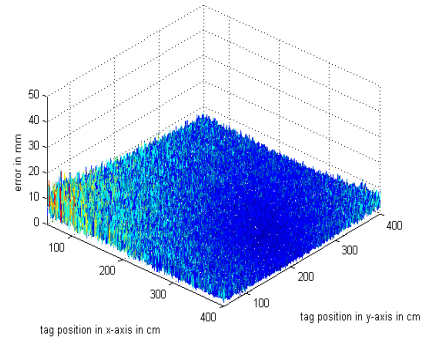


Figure 5.7: Positioning performance of the OMP, BCS, and STBCS-TDOA algorithm

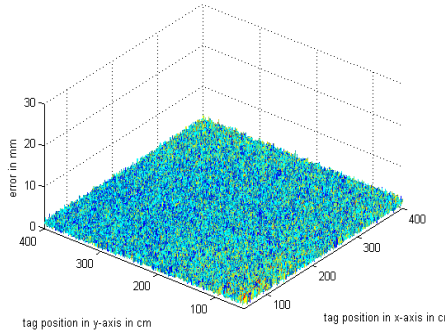
contrast, our STBCS algorithm by introducing prior information exhibits much better performance than the OMP and BCS algorithm at each iteration. Moreover, it is clearly observed that in the STBCS-TDOA algorithm, the higher the similarity level, the better positioning accuracy can be achieved with a limited number of iterations. For example, after 23 iterations, the STBCS algorithm with 89.1% similarity can detect the correct pulse arrival time with zero error and achieve an error on the order of millimeters. The performance of BCS and OMP are even worse because of the reconstruction failure due to insufficient measurements and noise. Therefore, our STBCS-TDOA algorithm can detect the pulse arrival time quickly with fewer iterations than the OMP and BCS algorithms for CS-based UWB positioning systems.

5.5.3 3D Positioning Performance

We investigate the 3D positioning performance using the proposed STBCS-TDOA algorithm and the sequential sampling method [8][44][13][45] for UWB positioning systems. The simulation is performed in a $5\text{m} \times 5\text{m} \times 4\text{m}$ room, where four base stations are placed at the following positions: (0, 0, 170), (4000, 0, 1855), (4410, 4435, 2860) and (0, 4545, 3260) (in millimeters). Four scenarios of UWB echo signals drawn



(a)



(b)

Figure 5.8: CS-based positioning system performance

from the UWB IEEE 802.15.4a model are used to represent the received UWB signals at the four base stations. The noise is added to the original signals ($\text{SNR} \approx 15\text{dB}$). The UWB positioning error is based on the received UWB signals at base stations using the sequential sampling method and the proposed STBCS-TDOA algorithms.

The dominating factors in the UWB positioning system using the sequential sampling method are essentially different from the CS-based UWB positioning system using the STBCS-TDOA algorithm. The sequential sampling method is the bottleneck to achieve a very high positioning accuracy in the UWB positioning system [13][45] because it is not a real-time signal acquisition method. Based on our previous realistic UWB experiments [13][45][8], we found that, for the sequential sampling based UWB positioning system, the dominating factor of positioning error is the asynchronous clocks between the pulse repetition clock in the tag and the sampling

clock in the receiver. The sequential sampling method can stretch the UWB pulse by K times using the K periods of UWB signals. The scalar K is determined by the offset of the pulse repetition clock and the sampling clock. Since those two clocks are asynchronous, the scale K is not fixed but fluctuating based on the relative jitter of those two clocks. We model such a behavior as

$$\tau'_{ji} = K(1 \pm j\%) \tau_{ji}, \quad (5.32)$$

where τ_{ji} is the time difference of pulse arrival time and K is the stretch scale number. In the wireless UWB positioning system, the relative time jitter at the tag and the base station will cause a drift scalar $j\%$, which introduces a positioning error and geometrical dilution. The error is linearly increased with the time difference τ_{ji} . This physical limitation makes it very difficult for the sequential sampling based UWB positioning systems to achieve high (e.g., sub-millimeter) accuracy. In the simulation, we adopt a state-of-the-art high stable oscillator with $\pm 5ppm$ time jitter [13] for simulations. In contrast, in the CS-based UWB positioning system, it is a real-time UWB signal acquisition. The positioning errors in CS-based UWB positioning systems are mainly from the reconstructed UWB signals. The peak of pulse and arrival time will be blurred due to a limited number of measurements. To overcome this limitation, we utilize the STBCS-TDOA algorithm to reconstruct the high resolution UWB signal with a low compression ratio (15%) and ultra low sampling rate ADCs for obtaining measurements.

Fig. 5.8 (a) and (b) show the performance comparison of the UWB positioning system using the proposed STBCS-TDOA algorithm and the sequential sampling method. It is seen that the positioning accuracy is significantly improved by using the STBCS-TDOA algorithm in the CS-based UWB positioning system. The expected absolute value of error using the CS-based sampling method is 0.8mm in Fig.5.8 (b) while it is about 5.52mm by using the sequential sampling method in (a). Moreover, Fig.5.8 (a) shows that the error is unevenly distributed. The minimum error is

achieved when the tag is at the geometrical center point, where the distance difference is zero ($\tau_j i = 0$). When the tag is moving close to the base station and the distance differences are significantly increasing, the error will be dramatically enlarged due to geometrical dilution. However, when we apply CS theory into the UWB positioning system, the error is evenly distributed, as demonstrated in Fig.5.8 (b). The standard error variance using the sequential sampling method is 6.65mm in (a) while it is only 0.52mm using our proposed scheme in (b). This represents a 12.8x improvement in positioning accuracy. Besides improving the positioning accuracy, other advantages of using STBCS-TDOA algorithm in CS-based UWB positioning system include real-time and high speed processing. Also note that a CS-based UWB positioning system using the proposed STBCS-TDOA algorithm will be a breakthrough, which has a potential to achieve much higher accuracy. Therefore, the positioning performance using the STBCS-TDOA algorithm in the CS-based UWB positioning system can be significantly improved compared with the system using the traditional sequential sampling method. The STBCS-TDOA algorithm can utilize prior information in CS-based UWB positioning systems to obtain good timing information but using a low compression ratio and low sampling rate ADCs to achieve a sub-mm accuracy.

We discussed how to apply TBCS to UWB position systems for high accuracy positioning performance in this chapter. However, the signal reconstruction algorithm is computationally expensive. In the next chapter, we will accelerate the computation by mapping the CS signal reconstruction algorithms on parallel devices.

Procedure 2 STBCS-TDOA Algorithm

- 1: Initialization: the hyperparameter α is set to $\alpha = \{\infty\}$; the nonzero signal index set $\Omega = \emptyset$; the parameters $\mathbf{a}, \mathbf{b} : \mathbf{a}, \mathbf{b} = \{0, 0\}$; the position of the tag is set to an initial position.
 - 2: Introducing temporal prior information: update $\mathbf{a}, \mathbf{b}$ using Eq.(5.8), (5.9), (5.12) and (5.13) from the previous reconstructed nonzero signal elements.
 - 3: **repeat**
 - 4: Introducing spatial prior information: receive the ongoing reconstructed signal elements from other simultaneous BCS procedures to update the parameter $\mathbf{a}, \mathbf{b}$.
 - 5: Calculate the components g_j and h_j (shown in Appendix A) and form the candidate set Λ based on Eq. (5.17).
 - 6: According to Proposition 1, select an index from the index set Λ . Assume it is the j -th index. Then add it into index set Ω : $\Omega = \Omega \cup j$;
 - 7: Update α_j by solving Eq.(5.22) if the parameters are updated so that $a_j \neq 0, b_j \neq 0$. Otherwise update α_j using Eq.(4.26).
 - 8: **if** $\Omega \cup \Lambda \neq \Omega$ **then**
 - 9: Delete the index: $\Omega = \Omega \setminus \{j\}$, where $j \in \Omega$ but $j \notin \Lambda$.
 - 10: **end if**
 - 11: Compute the signal elements whose index are in Ω . Output the index of the maximum signal based on the current reconstructed signal vector as the pulse arrival time for the TDOA algorithm for computing the position of the tag.
 - 12: Send out the ongoing reconstructed signal elements to other BCS procedures as spatial prior information.
 - 13: **until** converged
 - 14: Send out the reconstructed nonzero signal elements for the next frame utilization as temporal prior information.
 - 15: In parallel, the TDOA algorithm receives values of pulse arrival times from different BCS signal reconstruction procedures on base stations. With respect to one base station, the differences of pulse arrival and corresponding distance differences are calculated based on Eq.(5.23).
 - 16: The tag position is calculated using Eq. (5.26) with an initial start position.
 - 17: Go to step 16 to check any new updates. If there are new updates, utilizing the current calculated tag position as the initial position value recalculate the tag position. Otherwise iteratively compute the position of the tag until TDOA converges.
-

Chapter 6

Hardware Implementation of Bayesian compressed Sensing

6.1 Background and Introduction

Compressed Sensing (CS) can acquire signals at below the Nyquist sampling rate[27]. CS using signal reconstruction algorithms can indirectly acquire and reconstruct the signal from a small number of measurements obtained at a very low sampling rate; however, CS signal reconstruction algorithms are very computationally expensive[11] [27], becoming a bottleneck in time-sensitive applications[21]. High performance parallel computing for computation acceleration is needed for fast CS signal reconstruction. One of the most computationally expensive steps in CS signal reconstruction algorithms is Cholesky decomposition[5], so we desire to accelerate Cholesky decomposition and CS signal reconstruction by designing a dedicated hardware/software module to exploit parallelism.

Besides CS signal reconstruction in signal/image processing, Cholesky decomposition is one of most widely used decomposition algorithms in various applications and fields[69], such as in solving least square problems, linear regression, and least square fitting in mathematics, machine learning, and economics. Cholesky decomposition has

$O(\frac{1}{3}n^3)$ complexity, which is computationally expensive. Even when the matrix size n is small, the computation time is too expensive in many applications, such as real-time CS signal reconstruction[20][21][19]. In many other fields, such as computational chemistry[57], there exist a demand of computing Cholesky decomposition for many small matrices. Modern massively parallel devices, such as FPGAs and GPUs, can accelerate Cholesky decomposition by exploiting parallelism.

This chapter presents a high performance iterative Cholesky decomposition and CS signal reconstruction on FPGAs and GPUs. At end of this chapter, results show that FPGA and GPU implementations for Cholesky decomposition outperform LAPACK and MAGMA for small matrices. For the 256×256 matrix Cholesky decomposition, Cholesky decomposition on FPGAs show the best performance. Our proposed GPU implementation for Cholesky decomposition outperforms LAPACK and MAGMA when the matrix size is smaller than 4096×4096 . Based on the proposed iterative Cholesky decomposition, we implement a CS signal reconstruction algorithm on FPGAs and GPUs in single and double precision. Our FPGA and GPU implementation can achieve a high speedup compared with MAGMA and LAPACK for computation acceleration for fast CS signal reconstruction. Compared with the CS signal reconstruction on the CPU using LAPACK and the hybrid CPU/GPU mode, our FPGA implementation for CS signal reconstruction can achieve about 15x speedup and GPU implementation can achieve a 38x speedup.

The remainder of this paper is organized as follows. Section 6.2 introduces a novel Cholesky decomposition on a GPU with memory access optimization. Section 6.3 presents Cholesky decomposition on FPGAs by using one dedicated triangular equation solver. Section 6.4 summarizes a family of CS signal reconstruction algorithms and discusses implementation on both GPUs and FPGAs. Performance of GPU and FPGA implementations are compared from different perspectives in Section 6.5. Section 6.6 shows the performance for Cholesky decomposition on GPUs and FPGAs, as well as the performance of CS signal reconstruction algorithms implemented on GPUs, FPGAs, CPUs, and a hybrid platform.

6.2 Cholesky Decomposition on GPUs

A linear equation system, $Ax = b$, where A is a dense $n \times n$ matrix, and x and b are column vectors of size n , can be solved using a decomposition technique, LU for instance. If the matrix is symmetric and positive definite, Cholesky decomposition with complexity of $\frac{1}{3}O(N^3)$ is the most efficient in solving the system[36] and CS signal reconstruction algorithms. We developed a high performance Cholesky decomposition implementation on a GPU.

Procedure 3 lists the computational procedure for Cholesky decomposition on a GPU. The computation on GPUs include (1)a blocked Cholesky decomposition for factoring sub-matrix A_{11} , $A_{11} = L_{11}L_{11}^T$; (2)solving the triangular linear equations for updating A_{21} , $A_{21} := L_{21} = A_{21}L_{11}^{-1}$; and (3)triangular matrix-matrix multiplication for updating A_{22} , $A_{22} := A_{22} - L_{21}L_{21}^T$. Those computation is repeated until the whole matrix A is entirely factored and overwritten into the lower triangular matrix. Note that in each step, the size of sub-matrices and computation are tuned to exploit potential parallelism and maximize vector thread usage for good performance.

Procedure 3 Cholesky decomposition Algorithm on GPUs

- 1: Begin Cholesky decomposition
 - 2: **repeat**
 - 3: Partition the matrix $A = \{A_{11}, A_{12}; A_{21}, A_{22}\}$
 - 4: Compute Cholesky decomposition and overwrite the sub-matrix $A_{11} \leftarrow L_{11}$, where $A_{11} = L_{11}D_{11}L_{11}^T$
 - 5: Update and overwrite $A_{21} \leftarrow L_{21} = A_{21}L_{11}^{-T}$
 - 6: Overwrite A_{22} for the next iteration computation: $A_{22} \leftarrow A_{22} - L_{21}L_{21}^T$;
 - 7: Let $A = A_{22}$ and decompose A_{22} using the same procedure
 - 8: **until** Cholesky Decomposition reaches the right bottom sub-matrix so that the entire matrix A is factored into a lower triangular matrix
 - 9: Return the factored matrix A
-

Fig. 6.1 depicts the computation procedure of Cholesky decomposition on a GPU in detail. Due to the symmetric matrix, we only show half of the matrix which is needed in GPU global memory. Correspondingly, for GPU implementation, we have a total of four kernels. Sequentially, the first kernel performs the standard Cholesky

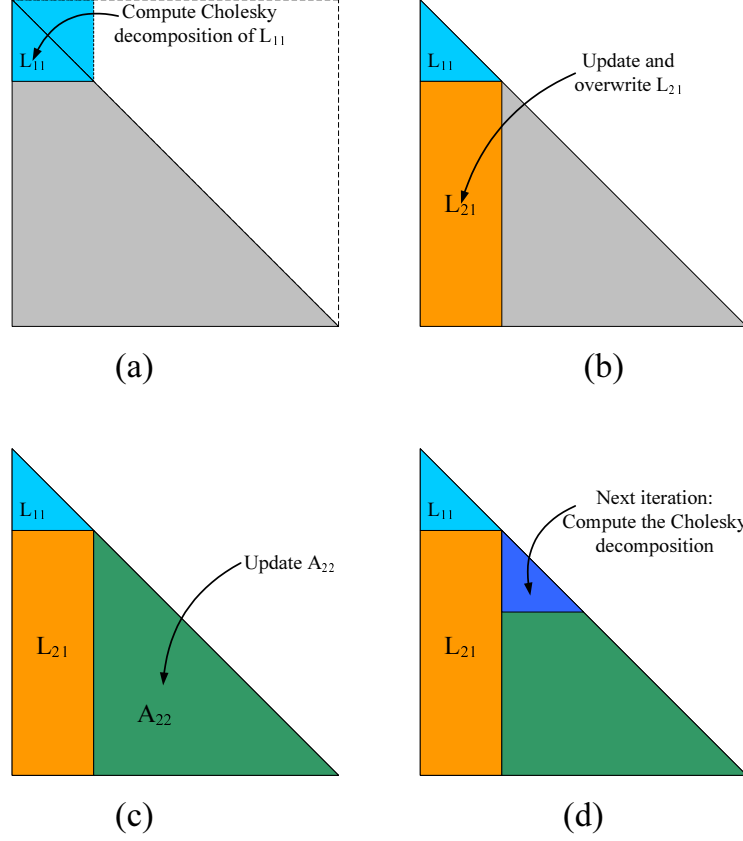


Figure 6.1: Cholesky decomposition on GPUs. (a), (b), (c), and (d) depict the computational procedure sequentially, where computation in (c) for updating A_{22} using matrix-matrix multiplication is dominant.

decomposition for factoring the first upper left sub-matrix A_{11} . The second kernel updates the elements in the strip, i.e., A_{21} , below the current working block by solving triangular linear equations. The third kernel updates the diagonal in the block. Finally, the fourth kernel updates the rest of the lower matrix A_{22} by doing a matrix-matrix multiplication. We loop through the blocks by sending block offsets until the entire matrix is factored. For best performance, we tune the sub-matrix size to achieve full-occupancy of vector threads on GPUs. We let A_{22} , the biggest sub-matrix, each iteration take advantage of massively parallel resources on GPUs for matrix-matrix multiplication. Compared with the standard Cholesky decomposition procedure, the proposed Cholesky decomposition can minimize copying operation, optimize memory access pattern and exploit potential parallelism for good performance.

Optimization is critical for achieving good performance on GPUs. Through optimizing the memory access pattern, performance can be greatly improved. The first optimization technique is to improve the use of shared memory. We load the current working block information into shared memory and reuse it repeatedly without having to go back to global memory. Next, we looked at coalesced global memory accesses to improve performance for GPU implementation. This was accomplished by loading the data into shared memory so that each thread loads data in a consecutive manner as described in [10]. Likewise, we store data in the same way. The third optimization technique was to avoid shared memory banking conflicts since the banking conflicts are causing non-coalesced reads and writes. To do this, we simply increased our shared memory size by one extra value. After these optimizations, memory access is not a limiting factor any longer. Note that the LDL^T Cholesky decomposition can be easily implemented on a GPU by simply modifying the computational procedure and corresponding kernels.

6.3 Cholesky Decomposition on FPGAs

For FPGA implementation, we adopt a modified Cholesky decomposition to solve the linear equation system $Ax = b$, which is given by

$$A = LDL^T \tag{6.1}$$

where D is a diagonal matrix and L is a unit lower triangular matrix with all unit elements on the diagonal.

The reason we adopt LDL^T , rather than the standard U^TU Cholesky decomposition, is multifold. First, compared with the standard Cholesky decomposition, the LDL^T Cholesky decomposition will remove the square root operation, which can save hardware resources and increase the data throughput[14]. Second, the LDL^T Cholesky decomposition can avoid the division dependency from the pipelines. Third,

the modified Cholesky decomposition with one more diagonal matrix D does not require more memory because the lower triangular matrix L has units on the diagonal that can be utilized for saving the diagonal matrix D .

We developed a novel iterative algorithm for calculating the LDL^T Cholesky decomposition for solving Eq.6.1. Compared with the standard Cholesky decomposition, the proposed augment algorithm only needs to iteratively solve triangular linear equations without any square root operations. The computation procedure is as below. The n by n symmetric matrix A is firstly partitioned into a 2×2 block matrix consisting of an $n-1$ by $n-1$ matrix A_{n-1} , a column vector t , and a scalar number g , which is shown in Eq. (6.2):

$$\begin{aligned}
A &= \left(\begin{array}{ccc|c} a_{11} & \dots & a_{1,n-1} & a_{1n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & \dots & a_{n-1,n-1} & a_{n-1,n} \\ \hline a_{n1} & \dots & a_{n,n-1} & a_{nn} \end{array} \right) \\
&= \left(\begin{array}{c|c} A_{n-1} & t^T \\ \hline t & g \end{array} \right) = LDL^T \\
&= \left(\begin{array}{c|c} L_{n-1} & 0 \\ \hline w & 1 \end{array} \right) \left(\begin{array}{c|c} D_{n-1} & 0 \\ \hline 0 & d_n \end{array} \right) \left(\begin{array}{c|c} L_{n-1}^T & w^T \\ \hline 0 & 1 \end{array} \right) \quad (6.2)
\end{aligned}$$

where g ($g = a_{nn}$) and d_n are scalars. Elements in matrix A are denoted by a_{ij} , ($i, j = 1, \dots, n$). Elements in the lower triangular matrix are denoted as l_{ij} , ($i, j = 1, \dots, n$). The vectors t and w are $1 \times (n-1)$ column vectors, where $t = \{a_{1n}, a_{2n}, \dots, a_{n-1,n}\}$ and $w = \{l_{1n}, l_{2n}, \dots, l_{n-1,n}\}$.

Obviously, we can find $A_{n-1} = L_{n-1}D_{n-1}L_{n-1}^T$, which is just the Cholesky decomposition for A_{n-1} . Suppose we have Cholesky decomposition results, $A_{n-1} = L_{n-1}D_{n-1}L_{n-1}^T$. In order to factor the matrix A based on L_{n-1} and D_{n-1} , we only

need to calculate the column vector w and diagonal element d_n . It is easy to verify that:

$$L_{n-1}D_{n-1}w = t \quad (6.3)$$

and,

$$d_n = g - w^T D_{n-1} w = g - \sum_{i=1}^{n-1} w_i^2 d_i \quad (6.4)$$

The column vector w can be calculated by solving the lower triangular linear equations in Eq. (6.3). Therefore, based on the factored matrix $A_{n-1} = L_{n-1}D_{n-1}L_{n-1}^T$ we know that matrix A can be decomposed. However, to obtain the Cholesky decomposition for matrix A_{n-1} we must know the decomposition $A_{n-2} = L_{n-2}D_{n-2}L_{n-2}^T$. Now imagine that this computation begins with the upper left element in the matrix A . We denote that one element matrix $A_1 = a_{11}$ and $L_1 = l_{11}$. Similarly, a 2 by 2 sub-matrix $A_2 = \{a_{ij}\}_{i,j=1}^2$ and $L_2 = \{l_{ij}\}_{i,j=1}^2$.

Fig. 6.2 demonstrates the iterative Cholesky decomposition on FPGAs. Cholesky decomposition begins with the upper left most element, i.e., $A_1 = \{a_{11}\}$. $A_1 = L_1 D_1 L_1^T = 1 \times a_{11} \times 1$, where $L_1 = \{l_{11}\}$. In the next step, we obtain matrix L_2 and D_2 by solving w and d from Eq. (6.3) and Eq. (6.4). In a similar manner, the matrices $L_3, D_3, \dots, L_{n-1}, D_{n-1}, L_n$, and D_n are computed sequentially by solving Eq. (6.3) and Eq. (6.4). Recursively, therefore, this is an iterative algorithm for computing Cholesky decomposition that only needs to solve the lower triangular equations for updating vectors and the diagonal elements.

It is not hard to design dedicated hardware for solving Eq. (6.4); however, solving Eq. (6.3) is essentially a serial computational procedure with heavy data dependency. Therefore a dedicated triangular equation solver for Eq. (6.3) must be designed and optimized for good performance. Note this triangular hardware solver is also utilized for solving linear equation systems after the Cholesky decomposition, which will be discussed in the next section.

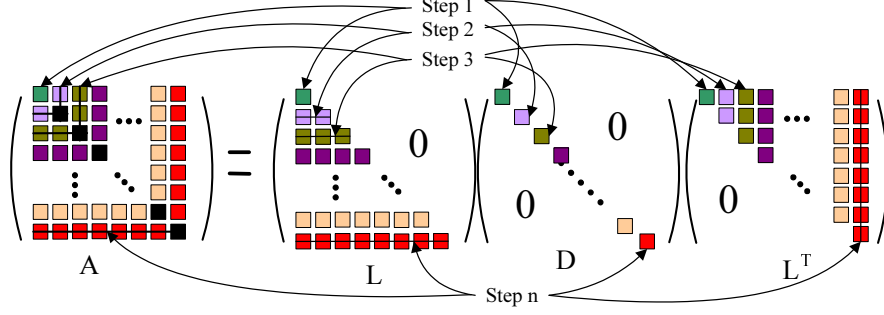


Figure 6.2: Cholesky decomposition computational procedure on FPGAs

6.3.1 Pipelined Triangular Linear Solver on FPGAs

We present a dedicated triangular linear solver for solving Cholesky decomposition and linear equation systems in this section. In order to further demonstrate why we adopt LDL^T Cholesky decomposition, we first show how to solve the linear equation system, $Ax = b$, based on standard U^TU Cholesky decomposition on FPGAs. Then we design a pipelined triangular solver on FPGAs for solving Cholesky decomposition and linear equation systems.

Assume matrix A is factored to $A = U^TU$ using standard Cholesky decomposition. In order to solve the linear equation system, $Ax = b$, we need to solve $U^TUx = b$, which is equivalent to two steps: (1) solve $U^Tr = b$ for the vector r using forward substitutions; (2) solve $Ux = r$ to get x using backward substitutions. The first step is expanded as:

$$\begin{aligned}
 r_1 &= b_1/u_{11} \\
 r_2 &= (b_2 - u_{12}r_1)/u_{22} \\
 &\vdots \\
 r_n &= (b_n - \sum_{i=1}^{n-1} u_{in}r_i)/u_{nn}
 \end{aligned} \tag{6.5}$$

The second step for solving $Ux = r$ is written as:

$$\begin{aligned}
x_n &= b_n/u_{nn} \\
x_{n-1} &= (b_{n-1} - u_{n-1n-2}r_{n-1})/u_{n-1n-1} \\
&\vdots \\
x_1 &= (b_1 - \sum_{i=1}^{n-1} u_{1i}r_i)/u_{11}
\end{aligned} \tag{6.6}$$

Obviously, it is observed that divisions must be computed for each component due to non-unit diagonal elements, u_{ii} , in the triangular matrix U . It is clear that the long latency of the divider (determined by the bit width of the dividend and divisor)[\[71\]](#), will adversely hurt the performance because the next input data depends on previous division results. Therefore, we adopt the LDL^T Cholesky decomposition to remedy this problem.

The LDL^T Cholesky decomposition can separate the divider dependency by introducing a diagonal matrix D . In this way introduction of the triangular matrices D make the diagonal components in the matrix L become all unit elements. Equivalently, in order to solve $LDL^Tx = b$, we need three steps: $Lz = b$, $Dr = z$ and $L^Tx = r$, which are:

$$\begin{aligned}
z_1 &= b_1 \\
z_2 &= (b_2 - l_{21}z_1) \\
z_3 &= (b_3 - l_{31}z_1 - l_{32}z_2) \\
&\vdots \\
z_n &= (b_n - \sum_{i=1}^{n-1} l_{ni}z_i)
\end{aligned} \tag{6.7}$$

Note that divider operations can be separately performed in parallel:

$$r_n = z_n/d_n \tag{6.8}$$

where d_n is the diagonal element in the diagonal matrix D . Finally the solution vector x is obtained similarly by solving the triangular linear system:

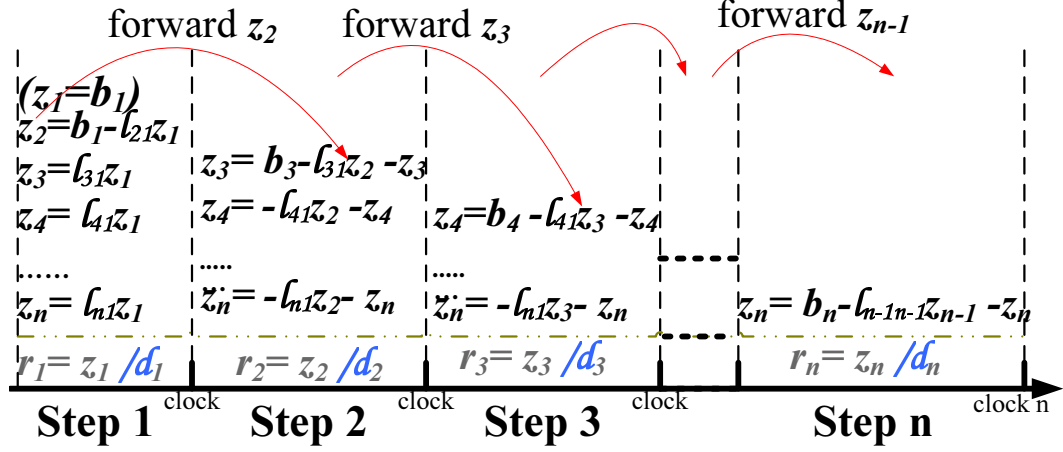


Figure 6.3: Computational procedure for solving triangular equations on FPGAs

$$\begin{aligned}
 x_n &= b_n \\
 x_{n-1} &= b_{n-1} - u_{n-1,n-2}r_{n-1} \\
 &\vdots \\
 x_1 &= b_1 - \sum_{i=1}^{n-1} u_{in}r_i
 \end{aligned} \tag{6.9}$$

Fig. 6.3 demonstrates the pipelined computation order for solving the linear triangular equations, Eq. (6.7) and Eq. (6.8). This computational order can maximize the potential computation parallelism to optimize the design for good performance. Note that $z_1, z_2, \dots, z_n$ are registers which save results for the accumulator. Also note that initially z_1 is provided then input to the hardware solver for calculating z_2 . After being computed, z_2 is then fed back for z_3 . The time delay between two clocks is determined by the hardware latency. At the same time the separated division also operates in parallel using an individual divider.

Fig. 6.4 illustrates a dedicated pipelined hardware architecture for implementing the triangular linear solver on FPGAs. This triangular linear equation solver consists of multiple PEs, result control logic, a divider, and BRAM modules. Inside a PE, one of the columns in the lower triangular matrix L (such as $l_{21}, l_{31}, \dots, l_{n1}$) is fed into the PEs, multiplied with z_j . The product is accumulated as the basic computation, demonstrated in each step in Fig. 6.3. The result z_{j+1} is further fed back through

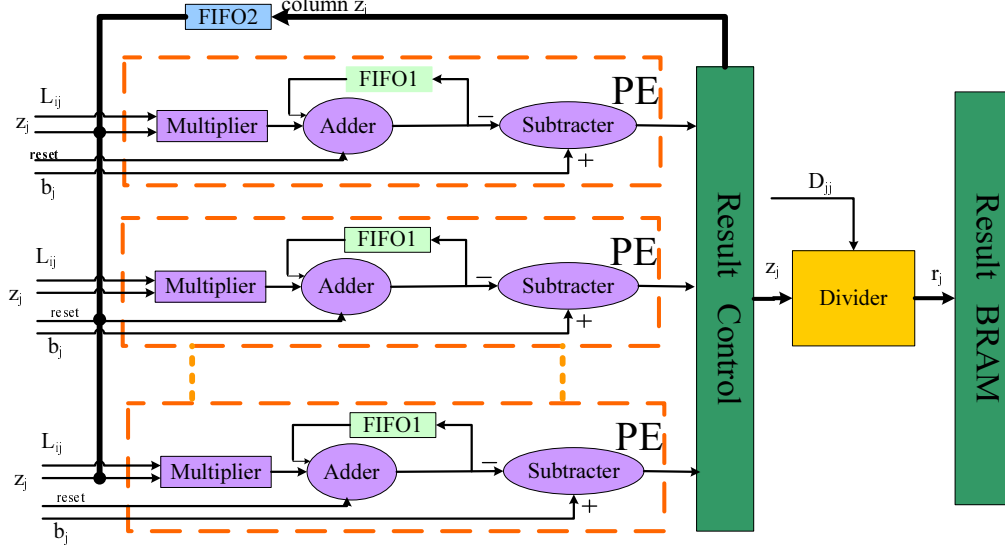


Figure 6.4: Architecture for solving triangular linear systems

FIFO2 outside of the PEs for computing the next z_{j+2} . The latency of the PE determines the time delay from z_j to z_{j+1} . Note that the divider is separated, operated in pipelined mode for calculating $(r_1, r_2, \dots, r_n)$ based on the $(z_1, z_2, \dots, z_n)$ output from the PEs, and controlled by the result control logic module.

Note that the FIFO in the PE is used for solving large linear equation systems based on a limited number of PEs, which improves the hardware usage efficiency. Without FIFO1, for example, in order to solve a 128×128 lower triangular linear system we have to use 127 PEs (z_1 is known) with one clock latency, according to Fig. 6.3, to process 127 pipelined multiplication-addition operations in Step 1. In Step 2, it only requires 126 PEs. In Step 3, 125 PEs are needed, and so on. Obviously, this design wastes hardware resources because the requirement of PEs decreases in later steps. The design of the short PE latency will decrease system clock frequency. So we propose a novel method to solve this problem by introducing one FIFO1. For example, assume that we have only 8 PEs with clock latency 8 for pipelined processing. In the first step it requires 127 multiplications for computing the product of the vector $\{l_{11}, l_{21}, l_{31}, \dots, l_{128,1}\}$ and number z_1 . So we have to run the hardware solver 16 times or clocks ($127/8$) based on 8 PEs. Thus, the depth of the FIFO1 is set

to 16, which is used to save the registers $\{z_1, z_2, \dots, z_{16}\}$ for the following accumulator or adder. The depth of the FIFO1 is controllable, which could be changed for the next steps. For instance, the depth of FIFO1 is adjusted to 15 if it requires 120 multiplications ($120/8=15$). Due to the limited depth, this kind of FIFO can be customized by designing the Finite State Machine (FSM) control logic plus separate RAM or registers in VHDL.

The BRAM is the end of the hardware solver for storing the solution vector. The memory size is determined by the input matrix size. To feed the hardware solver it has a memory block storing the elements of the triangular matrix L and column vector b inside or outside of the FPGA, depending on the matrix size and FPGA memory limitation. For a small matrix, elements can be stored inside the FPGA but for a large matrix, data must be stored in memory outside the FPGA. The data communication bandwidth decided by the number of PEs and the matrix size can be alleviated by using several independent memory blocks outside the FPGA.

The proposed triangular equation hardware solver for solving Eq. (6.7) and Eq. (6.8) can also be utilized for solving Eq. (6.3) for Cholesky decomposition. Note that the computation for updating matrix D using Eq. (6.4), which is not shown in Fig. 6.4, can be easily designed and implemented by using a pipelined multiplier-accumulator[73]. Therefore, the whole Cholesky decomposition of the linear equation system can be solved using only one dedicated triangular equation solver. Similarly, only one pipelined triangular equation solver is needed to solve least square problems[14]. Our design can simplify system complexity, reduce circuit delay, and improve performance. Furthermore, our proposed iterative Cholesky decomposition can be utilized for accelerating CS signal reconstruction algorithms.

6.4 Compressed Sensing on FPGAs and GPUs

6.4.1 Signal Reconstruction Algorithm

In CS theory, a signal is not directly acquired using Nyquist sampling theory but indirectly reconstructed from a few measurements in order to significantly reduce the sampling rate, which is described by:

$$\mathbf{y} = \Phi \mathbf{s} + \epsilon \quad (6.10)$$

where $\mathbf{s}$ is the sparse signal vector, which is needed to be reconstructed from the measurement vector $\mathbf{y}$; the matrix Φ , ($\Phi \in \mathbb{R}^{M \times N}$) is the known projection matrix with $M \ll N$, and ϵ is additive noise. Even though the length of vector $\mathbf{y}$ is much less than the length of signal vector $\mathbf{s}$ (i.e., $M \ll N$), CS theory shows that the signal $\mathbf{s}$ can be exactly or approximately reconstructed based on $\mathbf{y}$ and Φ .

However, CS signal reconstruction algorithms utilize linear programming, convex optimization, and Bayesian inference methods, which are very computationally expensive. A family of algorithms for signal reconstruction are through ℓ_2 minimization, where the key computation is associated with solving least square problems and Cholesky decomposition. A family of signal reconstruction algorithms is summarized in Procedure 4.

The most computationally expensive step in Procedure 4 is to estimate the signal vector at each iteration. To estimate the signal vector, one has to solve problem (6.11) or problem (6.12). A family of CS signal reconstruction algorithms, such as orthogonal matching pursuit (OMP)[35], STOMP[10], SOMP[23], FOMP[12], and so on, all need to iteratively solve Eq. (6.13) for signal reconstruction. Solving problem (6.11) is equivalent to solving the least square problem, which is equivalent to computing:

Procedure 4 A family of CS signal reconstruction algorithms

- 1: Input: the projection matrix Φ and measurement vector $\mathbf{y}$, where ϕ_i is the i -th column vector in the matrix Φ , y_i is the i -th element in the vector.
Output: the estimated signal vector S .
- 2: Initialize the iteration counter $t = 1$, the residual $r_t = y$, the candidate set $\Omega_t := \emptyset$, and the matrix $\Phi_t := \emptyset$.
- 3: **repeat**
- 4: Find the candidate index. In OMP, the candidate index can be found by projecting the measurement vector y onto the matrix Φ through matrix-vector multiplication, which is equivalent to solve a optimization problem
$$\lambda_t = \arg \max_{i=1, \dots, n} | \langle \phi_i, y \rangle |$$
- 5: Augment the index set $\Omega_t = \Omega_{t-1} \cup \{\lambda_t\}$ and the chosen matrix $\Phi_t := \{\Phi_{t-1}, \phi_{\lambda_t}\}$.
- 6: Reconstruct the signal vector by minimizing norm-2 to estimate the signal vector:

$$S_t = \arg \min_s \|\Phi_t - y\|_2 \quad (6.11)$$

Considering noise effects, add a constraint:

$$\begin{aligned} S_t &= \arg \min_s \|\Phi_t - y\|_2 \\ \text{Subject to : } &\arg \min \|S\|_0 \end{aligned} \quad (6.12)$$

- 7: Calculate the new residual and judge whether it converges or not. The residual can be calculated as:

$$r_t = \|\Phi_t S_t - y\|_2$$

- 8: Increment t , then return to Step 2.
 - 9: **until** converged
 - 10: Return the reconstructed signal vector.
-

$$\hat{\mathbf{s}} = ((\Phi_t)^T \Phi_t)^{-1} (\Phi_t)^T \mathbf{y} \quad (6.13)$$

where Φ_t is a set of selected column vectors from Φ , and $\hat{\mathbf{s}}$ is the reconstructed signal. In computing Eq. (6.13), Cholesky decomposition typically is adopted and is the most expensive step in calculating $\hat{\mathbf{s}}$ each iteration. Note that each iteration, the matrix

is augmented and the Cholesky decomposition operates on previous results for good performance.

Other well-known CS signal reconstruction algorithms solve problem (6.11) by introducing a regulation to combat noise for signal reconstruction. Those CS algorithms include regularized OMP[9], Bayesian compressed sensing (BCS)[59][39][53], TBCS[15], and MTBCS[61]. To solve problem (6.11), one needs to compute:

$$\hat{\mathbf{s}} = ((\Phi_t)^T \Phi_t + \mathbf{A})^{-1} (\Phi_t)^T \mathbf{y} \quad (6.14)$$

where matrix $\mathbf{A}$ is the regulation term for combating noise. Note that computing Eq. (6.14) is the most computationally expensive step in estimating the signal each iteration[15]. Cholesky decomposition should be utilized to iteratively calculate the signal vector. Also note that Cholesky decomposition can be performed by using previous results to save time.

Therefore, the iterative CS signal reconstruction algorithms are computationally expensive. Cholesky decomposition is the key computation step, used to iteratively factor the augmented matrix for signal reconstruction at each iteration. Other computation for CS signal reconstruction algorithms can be easily realized using CUBLAS subroutines[42][58]. We only focus on accelerating the most computationally expensive step, the iterative Cholesky decomposition. Our proposed high performance Cholesky decomposition for GPUs and FPGAs is not only for general matrix decomposition but also optimized for accelerating CS signal reconstruction algorithms.

6.4.2 Compressed Sensing on FPGAs

Fig. 6.5 shows implementation of the OMP algorithm for CS signal reconstruction on FPGAs. The proposed structure consists of several units:

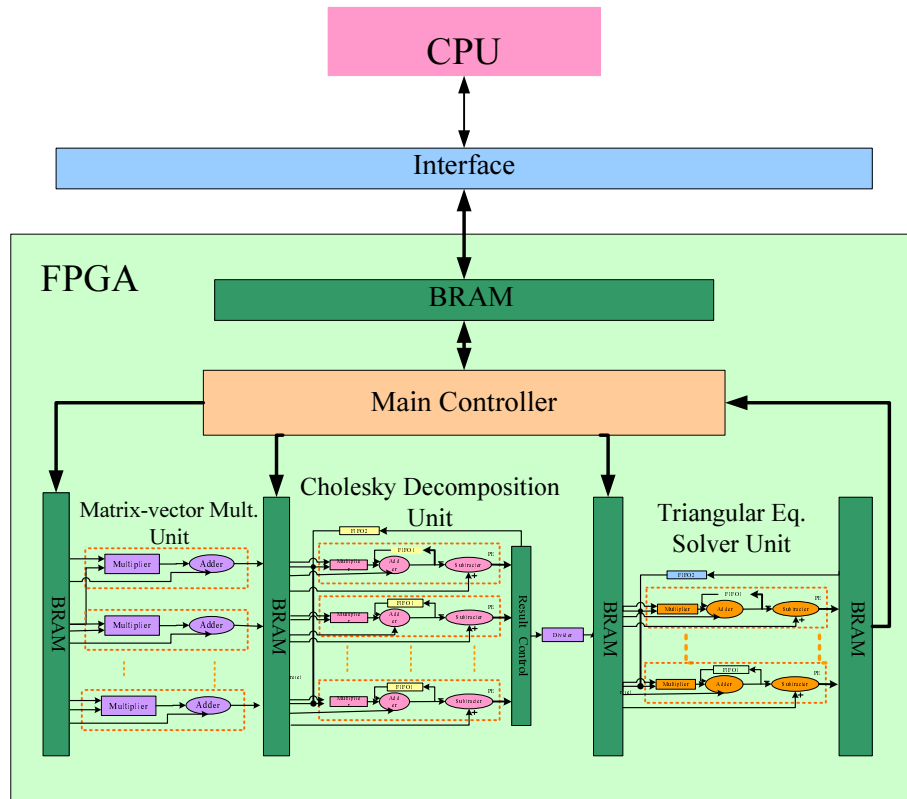


Figure 6.5: Compressed sensing on FPGAs

1. A matrix-vector multiplication unit for computing the inner product $\Phi_t y$ to select the candidate index.
2. A Cholesky decomposition unit for factoring the target matrix Φ_t^+ . Since $\Phi_t^+ = (\Phi_t^T \Phi_t)^{-1} \Phi_t$, a matrix-matrix multiplication unit can be designed to calculate the target matrix. As the matrix Φ is known, the target matrix Φ_t^+ can also be obtained by simply using a look-up table. Computation of the target matrix Φ_t^+ is not shown in the structure.
3. A linear triangular solver for solving linear equations and estimating the signal vector. It computes $S_t = \Phi_t^+ y$, where Cholesky decomposition is performed for $\Phi_t^T \Phi_t = LDL^T$. Note that the residual computation can be performed using the matrix-vector multiplication unit.
4. A main logic controller. The logic controller can be realized by using an embedded CPU, such as a PowerPC or Microblaze[72] on Xilinx FPGAs. It controls the data communication and schedules the working units for the system.

Note the proposed hardware structure is pipelined and a dual port BRAM is utilized between two units and stages for good performance. The data between two stages can be buffered and reorganized for data reuse in the next stage. Initialization data and results can be fetched and updated to the CPU from the FPGA through an interface controlled by the main controller. Also note that the proposed structure can be easily modified for implementing other CS signal reconstruction algorithms. At each iteration, Cholesky decomposition operates based on previous results. So only the new selected index and vectors must be transferred for Cholesky decomposition on the FPGA.

6.4.3 Compressed Sensing on GPUs

Fig. 6.6 shows the computational procedure of CS signal reconstruction algorithms on a GPU. The GPU implementation for CS signal reconstruction include several

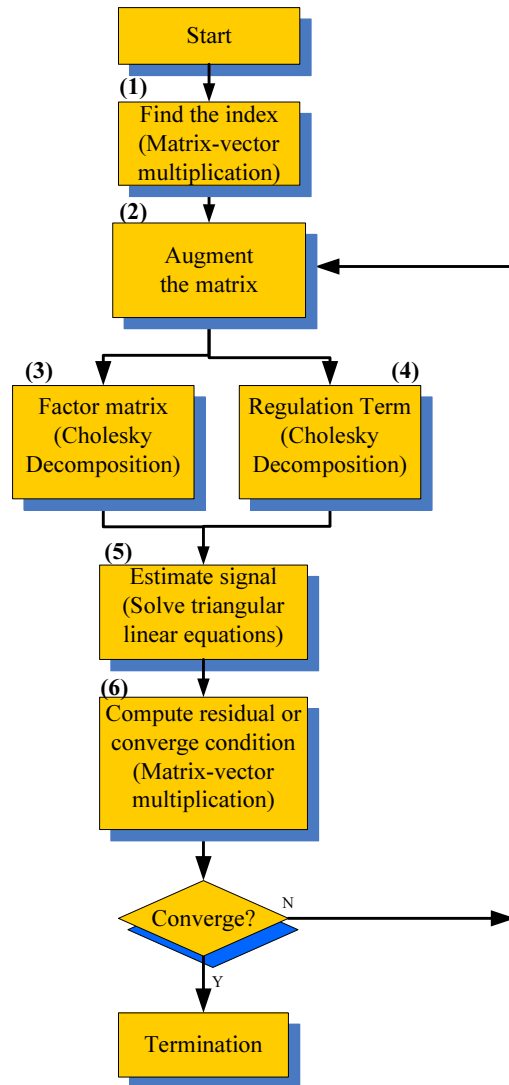


Figure 6.6: Compressed sensing on GPUs

sequential steps as described in Procedure. 4. The program starts with data transfer from CPU to GPU. In step (1), the candidate indices are chosen by doing matrix-vector multiplication, i.e., $\Phi_t y$. In step (2), the target matrix is calculated, where $\Phi_t = \{\Phi_t, \phi_t\}$ and $\Phi_t^+ = (\Phi_t^T \Phi_t)^{-1} \Phi_t$. Steps (3) and (4) perform matrix decomposition for solving linear equations. Steps (5) and (6) compute the signal vector and the residual for judging convergence. These computation steps are iteratively performed until the algorithm converges and the signal is reconstructed. The key computation in signal reconstruction is to compute Cholesky decomposition iteratively for the augmented matrix each iteration. Cholesky decomposition dominates computation time. Note that each iteration, the target augmented matrix, Φ_t^+ , needing to be factored using Cholesky decomposition can be conveniently solved based on the previous target matrix, Φ_{t-1}^+ , due to $\Phi_t = \{\Phi_{t-1}, \lambda_t\}$. Rather than calling a CUBLAS level 3 subroutine for Cholesky decomposition (i.e., `sport.cl` for single precision and `dportf.cll` for double precision) to recalculate the whole matrix, we utilize our proposed Cholesky decomposition, which can take advantage of previous results to save computation time and improve performance[17]. For other computation steps, the CUBLAS libraries can be exploited[49]. Note that Level 3 sub-routines about matrix-matrix inner product can be utilized in step (2). Level 2 sub-routines in CUBLAS for matrix-vector inner products can be exploited in step (1), and Level 1 sub-routines for scalar vector-vector computation can be used for calculating the regulation term in Steps (1) or (4) and other steps according to the CS signal reconstruction algorithm.

6.5 Performance Comparison of FPGA and GPU

In parallel computing, FPGAs and GPUs are two important technologies for application acceleration. However, FPGAs are essentially different from GPUs from many perspectives. A GPU has a fixed SIMD hardware architecture, which can provide massively parallel execution resources and high memory bandwidth. GPUs

are optimized for streaming, floating point calculations. FPGAs provide basic logic units, function blocks, lookup tables, and routing resources which are highly customizable for fine grained parallelism. Thus, FPGAs can offer better performance, flexibility, and low overhead because the hardware architecture can be fully optimized for the specific application.

GPUs tend to be much easier to programmers and have less hardware controllability compared with FPGAs. The programming language for GPUs are CUDA[48] or OpenCL[51], each extensions of C with an associated API. By using CUDA[48], a GPU programmer does not need to know as many hardware details. In CUDA, computational tasks are performed by thousands of threads in 3-dimensions, which are further organized as thread blocks and grids. CUDA provides a friendly interface for programmers by hiding hardware architecture details. However, for best performance, the CUDA program for Cholesky decomposition acceleration still needs to be tuned and optimized according to the characteristics of the application and GPU hardware limits, such as arithmetic operation order, memory access pattern, and communication. FPGA designers have to not only think about programming but also consider the low level design and implementation, such as pipelines, delays, and architecture details. FPGA designers utilize a hardware design language (such as VHDL or Verilog HDL) to have full control on the low level logic circuits, programmable resources, and hardware architecture. In addition, IP CoreGen[70] provides plenty of reusable and optimized IP cores, such as arithmetic units for scientific computations, which can save designers' time. However, the dedicated hardware architecture for a particular application is still needed to be specially designed for best performance. All hardware details, such as pipeline depth, latency, throughput, and memory bandwidth should be fully investigated and considered by FPGA designers.

GPUs have fixed memory architecture while FPGAs provide customizable memory hardware. The fixed memory architecture, such as cache, global memory, and shared memory, along with their associated bandwidths, may slow down the performance

for the Cholesky decomposition. For example, for the NVIDIA GeForce GTX480 GPU[50], the peak performance is 1.35TFLOPS in single precision with 1400MHz frequency. However, due to memory bandwidth this peak performance is very difficult to achieve for Cholesky decomposition. For FPGAs, a dedicated triangular linear equation solver is designed by using several pipelined PEs. The dedicated scalable architecture can be fully customizable and optimized, but on-chip memory is very limited. For example, the newer Xilinx XC6VSX475T FPGA in the Virtex 6 family has only 38,304 KB total BRAM memory which can be fully customized, so the limited memory resource is not suitable for large matrix operations. The memory communication bus can be fully customized for Cholesky decomposition in order to achieve the best performance. GPUs are more suitable for large size matrix operation while FPGAs are the best for small size matrix due to limited but customizable on chip hardware resources.

GPUs can support single and double precision floating point. When computation is associated with some operations, such as division, inversion, and square root, results from GPUs lose some precision. Due to limited hardware resources, the peak performance for double precision is worse than that for single precision. For instance, the GTX480 GPU has double precision performance only one half of the single precision performance [50], and older GPUs have one eighth the performance of single precision (or no support for double precision at all). On the other hand, FPGAs allow customizing the precision. For example, IP CoreGen provides IEEE 754 [14] arithmetic units for integer, fixed point, and single and double precision floating point. Moreover, the precision of those arithmetic units in the PE can be fully customized, such as adjusting mantissa and exponential bits. Lower precision arithmetic units on FPGAs require significantly less hardware resources than higher precision units, leading to higher frequency and performance for FPGAs.

For Cholesky decomposition, both GPUs and FPGAs can be utilized for high performance parallel computing; however, design and optimization for GPU and FPGA implementations are dramatically different. In order to achieve higher

frequency on FPGAs, one should reorder and simplify Cholesky decomposition for a pipelined architecture, minimize circuit delay, and maximize memory usage efficiency. Arbitrary memory access patterns in FPGAs may be supported but memory resources are limited. Memory outside the FPGA may be large but the communication interface may be a bottleneck. So we designed one pipelined triangular equation solver for Cholesky decomposition, where several identical Processing Elements (PEs) can operate in parallel. The BRAM memory can be organized to facilitate data fetch and store for reuse in the next iteration. In contrast to FPGAs, the proposed Cholesky decomposition on GPUs can take advantage of the SIMD architecture, which is totally different from FPGAs. The proposed algorithm for GPUs can facilitate optimizing memory access patterns to reduce data communication for good performance.

6.6 Hardware Results

The proposed Cholesky decomposition and CS signal reconstruction algorithms are implemented on the latest representative commercial products of GPUs and FPGAs. For performance comparison purpose, we also perform Cholesky decomposition and signal reconstruction algorithms on a CPU using LAPACK[37] and a hybrid system with CPU and GPU using MAGMA[41]. The test platform has an NVIDIA Fermi GTX480 GPU card[50] running at 1400MHz. The CPU is an Intel Quad Core i7 2.67GHz with 12 GB RAM. The software compilers consist of CUDA[48], OpenCL[51] version 3.0 and gcc version 4.3.3 for the GPU and CPU implementation. For the FPGA implementation, we adopt a Xilinx XC5VSX95T-2 FPGA in the Virtex 5 family. We utilize VHDL and Xilinx ISE 11.4 compiler, where IP CoreGen[70] can provide plenty of basic arithmetic units with customizable IEEE 754 standard floating point.

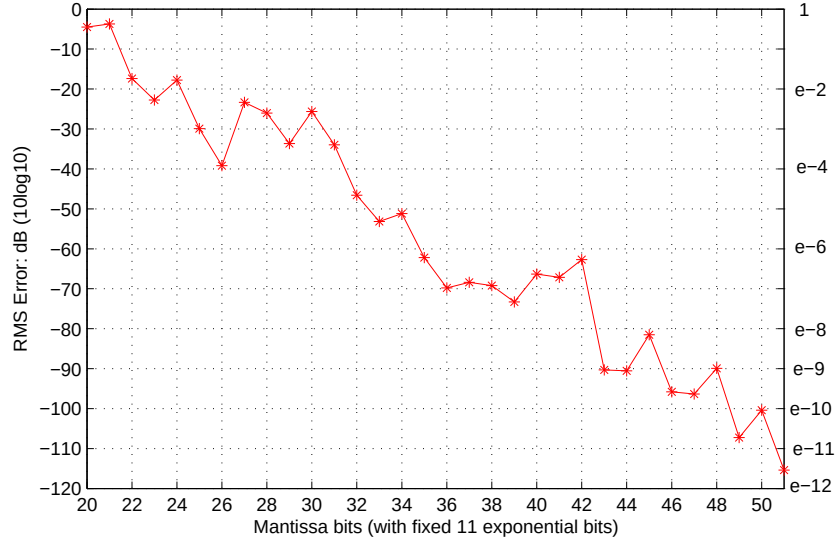


Figure 6.7: Accuracy using customized mantissa bits on FPGAs

6.6.1 Customizable Precision on FPGAs

FPGA implementation of Cholesky decomposition can fully customize the precision by designing fixed point, floating point, or other formats. In scientific computing, IEEE 754 floating point [14] is the most widely used format which has been proven to be suitable for most situations. Cholesky decomposition on FPGAs can specify the mantissa and exponential bits. High accuracy requirements imply high precisions leading to higher memory bandwidth, and more hardware resources, but lower achievable frequency. Customizing precision can save limited hardware resources, improve the achievable frequency, and maximize performance. There is a balance between computation speed and accuracy using different precisions with customized mantissa bits. Table 1 shows the resource usage for Cholesky decomposition for different precisions. We implemented 16 parallel pipelined Processing Elements (PEs) on the FPGA for Cholesky decomposition. Note that we utilized DSP48 modules in building multipliers for best performance. For notational simplicity, "s52e11" represents 52 mantissa bits and 11 exponent bits, which is double precision, and "s23e8" represents single precision with 23 mantissa bits and 8 exponent bits. The

number of exponent bits determines the dynamic range of representable values, so we vary the mantissa bits from 20 to 52 bits and fix the exponent to 11 bits.

Table 6.1: Performance and hardware resources usage on FPGA

FPGA Design	s20e8	s23e8 (single)	s32e11	s46e11	s52e11 (double)
Frequency	265 MHz	255 MHz	220 MHz	206 MHz	182 MHz
Slices	19%	24%	34%	62%	70%
DSP48	7%	22%	24%	35%	50%

Fig. 6.7 depicts the root mean square (RMS) error for the Cholesky decomposition of a 256x256 matrix with random elements. The RMS error, E_r , is defined as,

$$E_r = 10 \log_{10} \frac{1}{N} \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\tilde{L}_{ij} - L_{ij})^2 + \sum_{i=1}^n (\tilde{D}_{ii} - D_{ii})^2} \quad (6.15)$$

where the elements L_{ij} , ($L_{ij} \in L$) and D_{ii} , ($D_{ii} \in D$) are the reference results using double precision (s52e11), and the matrix $\tilde{L}$ and $\tilde{D}$ are the results with customized mantissa bits by factoring the truncated matrix. Since the error will be affected by the condition number, we test 1000 iid matrices for Cholesky decomposition with different mantissa size. Obviously, the RMS error is exponentially decreased with increasing mantissa size. Taking advantage of customized precision on FPGAs, designer could balance the accuracy, precision, and hardware resources according to application requirements.

6.6.2 Cholesky Decomposition Performance

Fig.6.8 and Fig.6.9 show the performance of Cholesky decomposition on GPUs, FPGAs, CPUs using LAPACK[37], and a hybrid system using MAGMA[41] in single and double precision. The FPGA is running at 180MHz for both single and double precision. The FPGA has around 20GFLOPS performance. For a small 256x256 matrix, the FPGA can outperform GPU, CPU, LAPACK, and MAGMA in single and

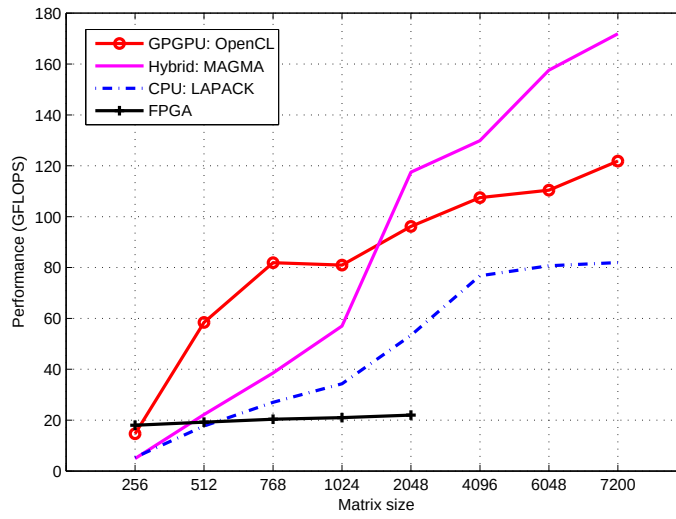


Figure 6.8: Performance comparison for Cholesky decomposition in single precision

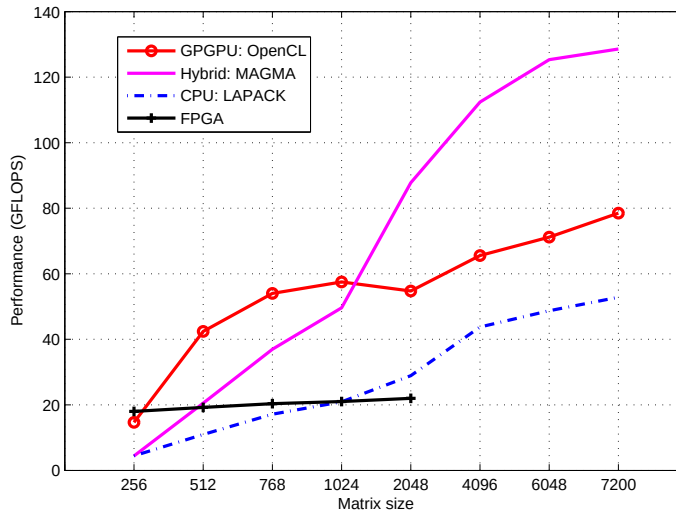


Figure 6.9: Performance comparison for Cholesky decomposition in double precision

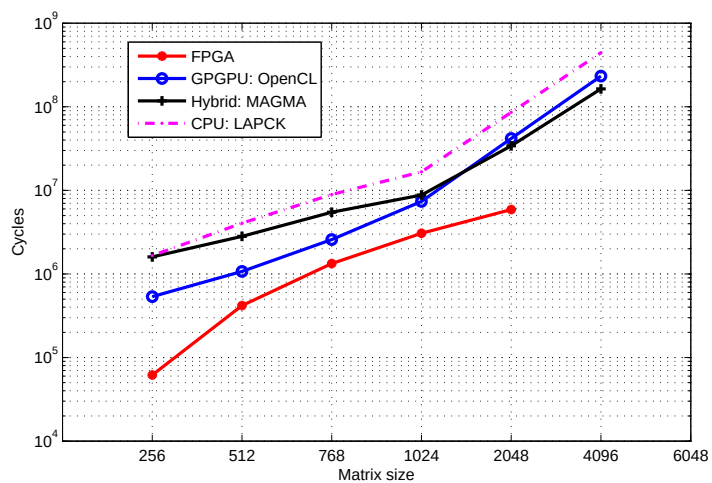


Figure 6.10: Computation cycles for Cholesky decomposition in single precision

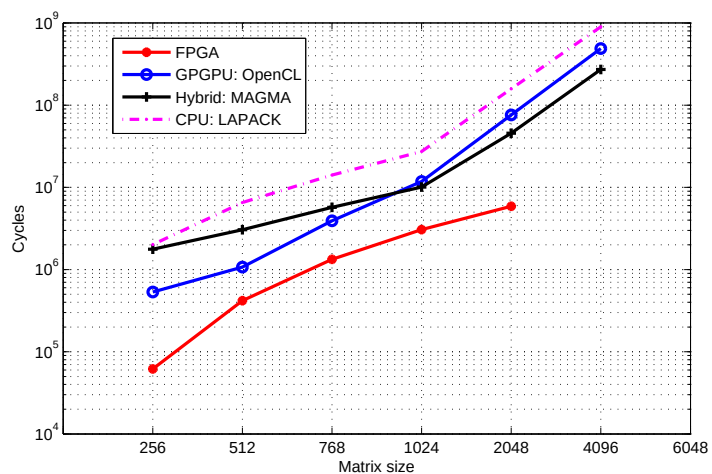


Figure 6.11: Computation cycles for Cholesky decomposition in double precision

double precision. The GPU has much better performance than the FPGA, LAPACK, and MAGMA when the matrix is smaller than 2048x2048. MAGMA exhibits much better performance than LAPACK for large matrices. With growth of the matrix size, the performance gap of MAGMA and LAPACK is increasing. Therefore, FPGAs and GPUs in single and double precision are suitable for small matrix Cholesky decomposition, outperforming LAPACK and MAGMA, while MAGMA shows advantage for large matrices because both the CPU and GPU are working simultaneously.

Fig.6.10 and Fig.6.11 compare efficiency and the computation cycles of Cholesky decomposition implemented on GPUs, FPGAs, CPUs using LAPACK[37], and a hybrid system using MAGMA[41] in single and double precision. The FPGA clock cycles are calculated based on the operations needed for a certain size matrix decomposition. For the GPU and CPU implementation, we obtain the computation cycles through the division of the measured execution time and frequency clocks (GPU:1400MHz, CPU: 2670MHz). Obviously, the number of cycles needed for Cholesky decomposition on FPGA, GPU, and CPU all increase with the matrix size due to $O(\frac{1}{3}n^3)$ computational complexity. The FPGA needs the minimum number of cycles because the hardware architecture is fully customized and optimized for the Cholesky decomposition. For single and double precision, our GPU implementation needs much fewer cycles than LAPACK and MAGMA, indicating more efficiency and less execution time depending on clock rate. When the matrix size is larger than 512×512 , that performance gap decreases.

6.6.3 Compressed Sensing Performance

Fig. 6.12 and Fig. 6.13 show performance comparison of the CS signal reconstruction algorithm implemented on GPUs and FPGAs using the proposed iterative Cholesky decomposition, on CPU using LAPACK, and in a hybrid mode using MAGMA in single and double precision. We implement the BCS algorithm[59] for signal

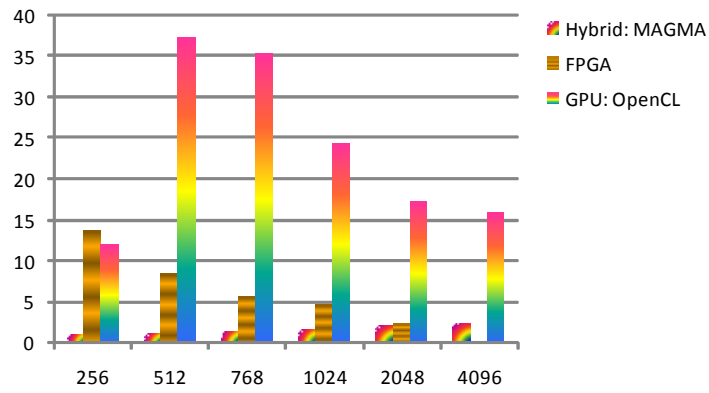


Figure 6.12: Hardware compressed sensing speedup of GPU, FPGA and MAGMA in single precision with respect to CPU execution time

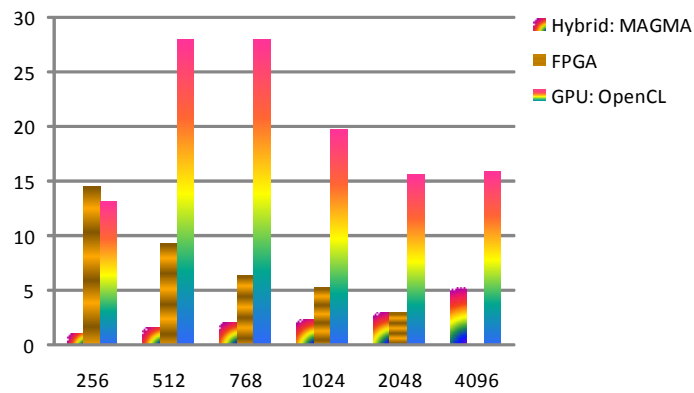


Figure 6.13: Hardware compressed sensing speedup of GPU, FPGA and MAGMA in double precision with respect to CPU execution time

reconstruction based on the proposed Cholesky decomposition for GPUs and FPGAs. The BCS algorithm is a typical de-noising CS signal reconstruction algorithm which has been widely utilized[61][15][21]. We compare the execution time of the CS algorithm on FPGA, GPU, and on a hybrid mode using MAGMA with respect to the execution time on CPU using LAPACK. We test different scale signal reconstruction tasks, i.e., the length of the target signal vector, which is shown in the the x-axis. Note that the length of the signal vector also represents the matrix size which must be iteratively factored using Cholesky decomposition. Due to limited on-chip memory, we only test the FPGA for computing a 1×2048 signal vector. In order to compare GPU, CPU, and hybrid CPU/GPU with FPGA, the performance comparison is only performed for reconstructing from 1×256 to 1×4096 signal vectors.

Fig. 6.12 compares the speedup of different implementations in single precision. The speedup is defined as: $S_r = \frac{T_c}{T_p}$, where T_c is the execution time of the BCS algorithm on CPU using LAPACK, T_p is the execution time of the BCS algorithm on FPGA, GPU, or a hybrid mode with MAGMA. It is observed that our GPU and FPGA implementations outperform the CPU using LAPACK and the hybrid system using MAGMA. The FPGA has the best performance compared with the GPU, CPU with LAPACK, and hybrid GPU/CPU with MAGMA in recovering a length of 1×256 signal vector, which is 13.6 times faster than the CPU using LAPACK and 12.9 times faster than MAGMA. Note that the FPGA has the best performance in Cholesky decomposition for a 256×256 matrix as shown in Fig. 6.8. With an increase of the problem scale, the speedup of FPGAs decreases but is still better than the CPU and MAGMA. GPUs can achieve a 37.6x speedup in reconstructing a 1×512 signal vector, and a 35.1x speedup for a 1×768 signal vector, which are much faster than MAGMA and FPGAs. This is because in computing the Cholesky decomposition for small matrices, the GPU and FPGA are better than the CPU using LAPACK and the hybrid system with MAGMA. This advantage is greatly enlarged in computing the iterative Cholesky decomposition required in CS signal reconstruction algorithms. The key point is that our proposed Cholesky decomposition on FPGAs

and GPUs for CS signal reconstruction can utilize previous results while MAGMA and LAPACK have to recalculate the whole matrix each iteration, so that good performance on FPGAs and GPUs is achieved. In recovering a small 1×128 signal vector, GPUs do not show good performance due to overhead and data communication for small matrices. With growth of the problem scale, GPU speedup is decreasing and the performance of CPUs using LAPACK and the hybrid mode using MAGMA are relatively increasing. MAGMA exhibits much better performance for large scale problems than LAPACK due to both GPUs and CPUs working together.

Fig. 6.13 compares the speedup of different implementations in double precision. Compared with the performance in single precision, our GPU and FPGA performance in double precision are still much better than the CPU using LAPACK and the hybrid CPU and GPU using MAGMA. In solving the 1×256 signal vector, the FPGA still has the best performance, which is 14.6 times faster than the CPU using LAPACK and hybrid mode using MAGMA. The GPU can achieve about 27.9x speedup compared with the CPU using LAPACK in computing the 1×512 signal vector. With problem size increasing, MAGMA in double precision shows much better performance than that in single precision. Therefore, FPGAs and GPUs are suitable for small-scale CS signal reconstruction problems. The GPU implementation can achieve best performance for recovering a 1×512 signal vector. The FPGA is the best in reconstructing the 1×256 signal vector. For large scale signal reconstruction problems, implementation on the CPU using LAPACK and the both CPU and GPU using MAGMA may be a good choice since the FPGA and GPU implementations for very large matrices are less effective.

In this chapter, we explore the hardware implementation of the BCS algorithm to accelerate the computation for fast signal reconstruction. The CS signal reconstruction algorithms in previous chapters can also be implemented on hardware such as FPGAs and GPUs for computational acceleration. In the next chapter, we conclude the dissertation and discuss the future work.

Chapter 7

Conclusions and Future Work

7.1 Contributions

In this PhD dissertation, we have made contributions to CS signal reconstruction algorithms and their hardware implementation. The main contributions include:

1. A novel *orthogonal pruning pursuit* (OPP) hard decision algorithm is proposed and developed for CS signal reconstruction. Based on our feedback structure, the OPP algorithm can reconstruct the signal much faster and require fewer measurements than other traditional hard decision CS algorithms. In contrast to other greedy CS algorithms, the key idea of OPP is to prune indices of zero elements based on a given index set for finding true nonzero indices. OPP can reduce a substantial amount of measurements. OPP can also make the signal as sparse as possible and thus improves the capability of defeating noise and interference.
2. A novel Turbo Bayesian Compressed Sensing (TBCS) algorithm is proposed to provide an efficient approach to transfer and incorporate this redundant information for joint sparse signal reconstruction. A space-time TBCS structure is developed for exploiting and incorporating the spatial and temporal prior information, both or independent, for space-time signal reconstruction. Based

on the BCS algorithm, we develop an iterative mechanism for information exchange among different reconstruction processes, motivated by the Turbo decoding structure, which is denoted *Turbo BCS*. To the author's best knowledge, there has not been any work applying the Turbo scheme in the BCS framework. A key contribution is the space-time structure to exploit and utilize the temporal and spatial redundancies. A mathematically elegant framework is proposed to impose an exponentially distributed hyperparameter on the existing hyperparameter α of the signal elements. This exponential distribution for the hyperparameter provides an approach to generate and fuse prior information with measurements in the signal reconstruction procedure. An incremental method [39] is developed to find the limited nonzero signal elements, which reduces the computational complexity compared with the expectation maximization (EM) method.

3. We propose a novel front-end scheme for a high precision CS-based UWB positioning system. Focusing on the properties of the UWB positioning system, we applied the TBCS algorithm into the UWB positioning system. The joint signal reconstruction algorithm is tightly integrated with the TDOA algorithm to develop a new algorithm for fast tag tracking, which is named Space-Time Bayesian Compressed Sensing (STBCS). The key idea in the proposed scheme is to *utilize the spatial and temporal redundancies existing in received UWB signals among base stations*. In one base station, the received UWB echo signals are similar in time because the tag moves very slowly compared with the pulse repetition frequency, which results in the temporal redundancy. Among different base stations, the received UWB echo signals are also similar, which yields in the spatial redundancy. The STBCS algorithm is in a pipelined mode to process the TBCS and TDOA signal processing algorithms for fast tag tracking. Both algorithms are realized in a modified incremental method.

The BCS algorithms are computationally expensive based on the BCS signal reconstruction algorithm framework. Hence, we develop a high performance iterative Cholesky decomposition and CS signal reconstruction implementation on FPGAs and GPUs to speed up the computation for fast signal processing. Contributions on hardware implementation in this dissertation include:

- Cholesky decomposition implementation on GPUs. We present a novel algorithm for Cholesky decomposition on GPUs through several optimized kernel calls to exploit potential parallelism, minimize copying operations, and optimize memory access for best performance.
- Cholesky decomposition implementation on FPGAs. We develop an augmented Cholesky decomposition on FPGAs, in which only one pipelined triangular linear equation solver is needed for solving the whole Cholesky decomposition and associated linear equation systems. A dedicated pipelined Processing Element (PE) is designed and optimized for realizing the triangular linear equation solver on a single FPGA. The proposed approach can avoid division dependencies and remove expensive square root operations so that hardware complexity is minimized and performance is greatly improved.
- CS signal reconstruction on GPUs and FPGAs. We analyze and exploit parallelism in signal reconstruction algorithms. We propose a high performance structure for computational acceleration for CS signal reconstruction on both GPUs and FPGAs.
- Performance comparison of parallel Cholesky decomposition and CS on different devices. We compare GPU and FPGA implementation from many perspectives, such as hardware architecture, software programming, optimization techniques, and precision.

7.2 Conclusions

In this dissertation, we have made contributions to CS signal reconstruction algorithms and their hardware implementation on GPUs and FPGAs. We proposed the OPP algorithm based on the hard decision CS signal reconstruction algorithm framework. A novel joint BCS algorithm is developed to integrating the redundant information for good performance. For the UWB positioning system, we integrate the CS signal reconstruction algorithm and the TDOA algorithm together to develop a new algorithm for fast signal processing. Finally, we implement the CS algorithm on GPUs and FPGAs for high performance computation.

For the hard decision CS signal reconstruction algorithm, we develop a novel OPP algorithm. OPP is based on a feedback structure to exploit the prior knowledge in consecutive signal frames. The OPP algorithm can significantly reduce the number of measurements and improve the capability to defeat noise and interference. The numerical simulation shows that our proposed scheme can achieve good performance using very few measurements, while traditional CS algorithms like OMP and BP cannot successfully reconstruct the UWB signal. It is also demonstrated that our proposed structure and algorithm achieve good BER performance under a few measurements in the proposed UWB communication system.

Then, a joint signal reconstruction algorithm is developed to exploit and integrate the spatial and temporal prior information existing in sparse signals, e.g. UWB pulses. The turbo BCS algorithm has been designed to fully exploit prior information from both space and time. Numerical simulation results have shown that the proposed TBCS outperforms the MBCS and traditional BCS, in terms of the robustness to noise and reduction of the required amount of samples.

For the UWB positioning system, a front-end TBCS-TDOA algorithm is developed to apply the CS theory into UWB positioning systems for achieving ultra-high accuracy. The STBCS algorithm is able to exploit space-time prior information in joint UWB signal reconstruction to dramatically reduce the number of measurements

while thus the sampling rate of ADCs. Simulation results demonstrate that our proposed STBCS-TDOA algorithm in the CS-based UWB positioning system can achieve a sub-mm accuracy while using low sampling rate ADCs. The results using STBCS-TDOA are not sensitive to the tag location relative to base stations, so this approach avoids the geometrical dilution of accuracy using the traditional sequential sampling method.

In order to alleviate the computational cost, we propose a novel Cholesky decomposition implementation on both GPUs and FPGAs for computational acceleration. Cholesky decomposition on a GPU is optimized with minimum memory access for good performance. Cholesky decomposition on FPGAs needs only one pipelined and dedicated triangular linear equation solver. CS signal reconstruction algorithms are both implemented based on the proposed iterative Cholesky decomposition, which greatly improve performance. Results shows that Cholesky decomposition on FPGAs and GPUs have much better performance than the CPU using LAPACK and the hybrid system using MAGMA for small matrices. In terms of computing cycles, the FPGA implementation shows the highest efficiency. We can achieve around 15x speedup for CS signal reconstruction algorithm on FPGAs and around 38x speedup on GPUs compared with the implementation with LAPACK and MAGMA.

7.3 Future Work

One of the main problem in CS is how to deal with the noise in the sparse signal vector. The basic assumption in CS is that the original signal vector should be sparse: most of elements in the vector are zero and only a few of elements are nonzero; however, the additive noise in the signal breaks the sparse assumption, which brings a fundamental problem to reconstruct the noise-free signal.

One of the trends in the research of CS is to deal with the noise in the signal vector. Bayesian compressed sensing theory develops a way to reducing the noise. However,

if the additive noise is large, the signal reconstruction effect is not acceptable, which has similar performance as the hard decision CS algorithms.

Based on our previous research work on BCS theory, our future is to exploit new ways to combat the noise. In particular, in many applications, we have prior knowledge about the noise strength, or statistical properties, or other information. All the useful information can be integrated into the algorithm for better combating the noise to improve the performance of signal reconstruction. This research work is significantly meaningful and will impact many applications, such as signal acquisition, signal compressing, and signal sensing.

Bibliography

Bibliography

- [1] A. F. Molisch, “IEEE 802.15.4a channel model - final report,” [online], IEEE 2004.
www.ieee802.org/.../channel-model-final-report-r1.pdf 29, 36, 54, 60, 68, 69, 85
- [2] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, 2nd edition, CRC Press, 2003. 45
- [3] A. G. Dimakis, P. O. Vontobel, “LP decoding meets LP decoding: a connection between channel coding and compressed sensing,” in *Proceedings of the 47th annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, 2009. 3, 14, 15
- [4] A. Septimus and R. Steinberg, “Compressive sampling hardware reconstruction,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, July, 2010. 20
- [5] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM Journal on Computing*, vol. 24, no. 2, pp. 137-147, 1995. 19, 96
- [6] Cholesky factorization algorithm, *Parallel Linear Algebra Package*, The University of Texas at Austin, [Online], 2007.
<http://userweb.cs.utexas.edu/~lapack/> 19

- [7] C. Hegde and R. G. Baraniuk, “Compressive sensing of a superposition of pulses,” in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, March, 2010. [18](#)
- [8] C. Zhang, M. Kuhn, B. Merkl, A. Fathy, and M. Mahfouz, “Real-time noncoherent UWB positioning radar with millimeter range accuracy: theory and experiment,” in *IEEE Trans. on Microwave Theory and Techniques*, vol. 58, pp. 9-20, 2009. [17](#), [18](#), [91](#), [92](#)
- [9] D. Needell and R. Vershynin, “Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit,” in *IEEE Journal of Selected Topics in Signal Processing*, pp.310-316, 2010. [14](#), [110](#)
- [10] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, “Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit,” Technology Report, Mar., 2006. [14](#), [108](#)
- [11] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inform. Theory*, vol.52, no.4, pp.1289-1306, April, 2006. [1](#), [3](#), [4](#), [13](#), [96](#)
- [12] D. Yang, H. Li, and G. D. Peterson, “Feedback orthogonal pruning pursuit for pulse acquisition in UWB communications,” in *the 20th Personal, Indoor and Mobile Radio Communications Symposium (PIMRC)*, Tokyo, Sept. 2009. [18](#), [33](#), [67](#), [108](#)
- [13] D. Yang, A. Fathy, H. Li, G. D. Peterson, and M. Mahfouze, “Millimeter accuracy UWB positioning system using sequential sub-sampler and time difference estimation algorithm,” in *IEEE Radio and Wireless Symposium (RWS)*, New Orlean, Jan. 2010. [17](#), [18](#), [51](#), [52](#), [66](#), [67](#), [69](#), [70](#), [71](#), [91](#), [92](#), [93](#)

- [14] D. Yang, G. D. Peterson, H. Li and J. Sun, “An FPGA implementation for solving least square problem,” in *the Seventeenth IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa, California, April., 2009. [20](#), [47](#), [52](#), [100](#), [107](#), [116](#), [118](#)
- [15] D. Yang, Husheng Li, and G.D. Peterson, “Space-time Turbo Bayesian Compressed Sensing for UWB system,” in *IEEE International Conference on Communications (ICC)*, Kapton, May, 2010. [18](#), [67](#), [110](#), [124](#)
- [16] D. Yang, G. Peterson, and H. Li, “High performance reconfigurable computing for Cholesky decomposition,” in *Symposium on Application Accelerators in High Performance Computing(SAAHPC)*, UIUC, Urbana, July, 2009. [20](#), [23](#), [27](#), [47](#)
- [17] D. Yang, G. Liang, D. Jenkins, G. D. Peterson, and H. Li, “High performance relevance vector machine on GPUs,” *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*, Knoxville, TN, July, 2010. [20](#), [114](#)
- [18] D. Yang, J. Sun, J. Lee, G. Liang, D. D. Jenkins, G. D. Peterson, and H. Li, “Performance comparison of Cholesky decomposition on GPUs and FPGAs,” *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*, Knoxville, TN, July, 2010. [20](#)
- [19] D. Yang, H. Li, G. D. Peterson, and A. Fathy, “UWB signal acquisition in positioning systems: Bayesian compressed sensing with redundancy,” in *Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, March, 2009. [5](#), [17](#), [18](#), [19](#), [23](#), [67](#), [97](#)
- [20] D. Yang, H. Li, G. D. Peterson, and A. E. Fathy, “Compressed sensing based UWB receiver: hardware compressing and FPGA reconstruction,” in *Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, March, 2009. [5](#), [18](#), [19](#), [20](#), [24](#), [33](#), [34](#), [52](#), [67](#), [70](#), [97](#)

- [21] D. Yang, H. Li, and G.D. Peterson, “Decentralized Turbo Bayesian compressed sensing with application to UWB systems,” in *EURASIP Journal on Advances in Signal Processing*, March, 2011. 19, 96, 97, 124
- [22] D. Baron, M. B. Wakin, M. F. Duarte, S. Sarvotham, and R. G. Baraniuk, “Distributed compressed sensing,” [online], 2005.
<http://arxiv.org/abs/0901.3403> 15, 19
- [23] D. Leviatan and V. N. Temlyakov, “Simultaneous approximation by greedy algorithms,” IMI Report 2003, University of South Carolina at Columbia, 2003.
14, 15, 19, 108
- [24] D. Baron, S. Sarvotham, and R. G. Baraniuk, “Bayesian compressive sensing via belief propagation,” *IEEE Transactions on Signal Processing*, vol. 58, Issue 1, pp.269-275, 2009. 3, 14, 15
- [25] D. M. Pozar, *Microwave Engineering*, Wiley, 1998. 6
- [26] E. Berg, and M. P. Friedlander, “Joint-sparse recovery from multiple measurements,” *IEEE Transactions on Information Theory*, vol.56, no.5, pp.2516-2527, 2010 15, 19
- [27] E. J. Candès, J. Romberg and T. Tao, “Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information,” *IEEE Trans. Inform. Theory*, vol.52, no.2, pp.489-509, Feb. 2006. 1, 4, 13, 18, 24, 26, 67, 96
- [28] F. Zhang and Henry D. Pfister, “On the iterative decoding of high-rate LDPC codes with applications in compressed sensing,” submitted to *IEEE Trans. on Inform. Theory*, 2011. 14, 15
- [29] H. Zayyani, M. Babaie-Zadeh, and C. Jutten, “Bayesian pursuit algorithm for sparse representation” in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, April, 2009 14, 17

- [30] J. L. Paredes, G. R. Arce, and Z. Wang, "Ultra-Wideband compressed sensing: channel estimation," *IEEE Journal of Selected Topics in Signal Processing*. vol. 1, no. 3, pp.383-395, 2007. [18](#), [67](#)
- [31] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *IEEE Trans. Signal Processing*, vol. 86, pp. 572-588, Apr. 2006. [14](#), [15](#), [19](#)
- [32] J. Haupt and R. Nowak, "A generalized restricted isometry property," University of Wisconsin Madison Technical Report, May. 2007. [13](#)
- [33] J. A. Tropp, "Algorithms for simultaneous sparse approximation. Part II: Convex relaxation," *IEEE Trans. Signal Processing*, vol. 86, pp. 589-602, Apr. 2006. [14](#), [19](#)
- [34] J. Jung, D. OLeary, "Cholesky decomposition and linear programming on a gpu," in *ACM/IEEE Conference on Supercomputing*, Reno, NV, November, 2007. [19](#)
- [35] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 53, no. 12, pp. 4655-4666, 2007 [13](#), [14](#), [26](#), [108](#)
- [36] L. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, 1997. [25](#), [26](#), [98](#)
- [37] Linear Algebra PACKage (LAPACK), Version 3.2.1, [Online], 2010.
<http://www.netlib.org/lapack/> [20](#), [117](#), [119](#), [122](#)
- [38] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol.1, pp.211-244, 2001. [3](#), [14](#), [17](#), [38](#), [40](#), [42](#), [43](#), [46](#), [47](#), [54](#), [149](#)

- [39] M. E. Tipping and A. C. Faul, “Fast marginal likelihood maximisation for sparse Bayesian models,” in *Proc. of the International Conference on Artificial Intelligence and Statistics*, Keywest, Fl., June, 2003. [14](#), [35](#), [38](#), [43](#), [48](#), [49](#), [73](#), [78](#), [110](#), [127](#)
- [40] M. Davenport and M. Wakin, “Analysis of orthogonal matching pursuit using the restricted isometry property,” *IEEE Transactions on Information Theory*, Vol. 56, issue 9, pp. 53-64, Jan. 2010. [13](#)
- [41] Matrix Algebra on GPU and Multicore Architectures (MAGMA) user guide, [Online], 2009.
<http://icl.cs.utk.edu/magma/> [20](#), [117](#), [119](#), [122](#)
- [42] M. Andrecut, “Fast GPU implementation of sparse signal recovery from random projections,” [online], 2008.
<http://arxiv.org/arxiv/pdf/0809/0809.1833v1.pdf> [20](#), [110](#)
- [43] M. Andrieu, L. Rebollo-Neira, and E. Sagonas, “Backward-optimized orthogonal matching pursuit approach,” *IEEE Signal Processing Letters*, Vol. 11, pp.705-708, Sept. 2004. [13](#), [26](#)
- [44] M. Kuhn, C. Zhang, B. Merkl, D. Yang, Y. Wang, M. Mahfouz, and A. Fathy, “High accuracy UWB localization in dense indoor environments,” in *IEEE International Conference on Ultra-Wideband (ICUWB)*, Germany, Sep. 2008. [91](#)
- [45] M. Mahfouz, C. Zhang, B. Merkl, M. Kuhn, and A. E. Fathy. “Investigation of high-accuracy indoor 3D positioning using UWB technology,” *IEEE Trans. on Microwave Theory and Technology*, no. 6, pp. 88-96, 2008. [17](#), [18](#), [66](#), [70](#), [71](#), [91](#), [92](#)

- [46] N. Vaswani and W. Lu, “Modified-CS: Modifying compressive sensing for problems with partially known support,” *IEEE International Symposium on Information Theory (ISIT)*, Seoul, Jun. 2009. 15, 19, 23
- [47] N. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, pp. 258, 2002. 144, 157
- [48] NVIDIA CUDA programming guide, [Online], 2007.
http://www.nvidia.com/object/cuda_home.html 115, 117
- [49] NVIDIA CUBLAS library, [Online], 2008.
http://developer.download.nvidia.com/compute/CUBLAS_Library 114
- [50] NVIDIA Tesla GTX480 computing processor, [Online], 2010.
http://www.nvidia.com/object/product_geforce_GTX480_us.html 116, 117
- [51] OpenCL programming guide, [Online], 2009.
<http://developer.download.nvidia.com/.../ProgrammingGuide.pdf> 115, 117
- [52] O. Maslennikov, V. Lepekha, A. Sergiyenko, A. Tomas, and R. Wyrzykowski, “Parallel implementation of Cholesky LLT algorithm in FPGA-based processor,” *Parallel Processing and Applied Mathematics*, pp.137-141, 2008. 19
- [53] P. Schniter, L. C. Potter, and J. Ziniel, “Fast Bayesian matching pursuit,” *Workshop on Information Theory and Applications (ITA)*, La Jolla, CA, Jan. 2008. 17, 110
- [54] P. Zhang, Z. Hu, R. C. Qiu, and B. M. Sadler, “Compressive sensing based Ultra-wideband communication system,” in *IEEE International Conference on Communications (ICC)*, Dresden, Germany, June, 2009. 33

- [55] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation* pp. 253-263, December 2008. [13](#)
- [56] R. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, pp. 118-121, July, 2007. [13](#)
- [57] R. Harrison, G. Fann, T. Yanai, Z. Gan, G. Beylkin, "Multiresolution quantum chemistry: Basic theory and initial applications," *J. Chem. Phys.*, pp. 102-121, 2004. [19](#), [97](#)
- [58] S. Lee and S. J. Wright, "Implementing algorithms for signal and image reconstruction on graphical processing units," Technical Report, [online], 2008. <http://www.optimization-online.org/2008/11/2131.pdf> [20](#), [110](#)
- [59] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Processing*, vol. 56, no. 6, June 2008. [3](#), [14](#), [17](#), [33](#), [38](#), [40](#), [48](#), [54](#), [73](#), [78](#), [110](#), [122](#)
- [60] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Processing*, vol. 53, no. 7, pp. 2477-2488, July 2005. [14](#), [15](#), [19](#)
- [61] S. Ji, D. Dunson, and L. Carin, "Multitask compressive sensing," *IEEE Trans. Signal Processing*, vol. 57, issue 1, pp. 92-106, 2009. [17](#), [34](#), [41](#), [44](#), [54](#), [110](#), [124](#)
- [62] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk, "Analog-to-Information conversion via random demodulation," *Proc. IEEE Dallas Circuits and Systems Workshop (DCAS)*, 2006. [24](#)
- [63] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Bayesian compressive sensing using laplace priors," *IEEE Transactions on Image Processing*, Vol. 19, issue 1, pp. 53-64, Jan. 2010. [17](#), [43](#)

- [64] T. Do, L. Gan, N. Nguyen and T. Tran, “Sparsity adaptive matching pursuit algorithm for practical compressed sensing,” *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, Oct. 2008. 14, 29
- [65] V. Volkov, J. Demmel LU, “LU, QR and Cholesky factorizations using vector capabilities of GPUs,” Technical Report, UCB/EECS-2008-49, EECS Department, University of California, Berkeley, May 2008. 20
- [66] V. Cevher, P. Boufounos, R. G. Baraniuk, A. C. Gilbert, and M. J. Strauss, “Near optimal bayesian localization via incoherence and sparsity,” *IEEE/ACM Information Processing in Sensor Networks (IPSN)*, San Francisco, CA, April 2009. 18
- [67] W. Wang, M. Garofalakis, and K. Ramchandran, “Distributed sparse random projections for refinable approximation,” in *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, Cambridge, Massachusetts, April 2007 15, 19
- [68] W. S. Anglin, and J. Lambek, *The Heritage of Thales*, Springer, 1993. 152, 160
- [69] X. Shen, M. Guizani, R. C. Qiu, and T. LeNgoc, *Ultra-Wideband Wireless Communications and Networks*, John Wiley, 2006. 4, 19, 69, 96
- [70] Xilinx IP Coregen, [online], 2006.
http://www.xilinx.com/ise/products/coregen_overview.pdf 115, 117
- [71] Xilinx Logicore, “Xilinx pipelined divider v3.0,” [online], 2008.
<http://www.xilinx.com/ipcenter/.../sdivider.pdf> 104
- [72] Xilinx PowerPC and MicroBlaze development kit, [online], 2006.
<http://www.xilinx.com/publications/EmbeddedKit.pdf> 112

- [73] Y. Dou, S. Vassiliadis, G. K. Kuzmanov, G. N. Gaydadjiev, “64bit floating point FPGA matrix multiplicatio,” in *13th International Symposium on Field Programmable Gate Arrays (FPGA)*, Monterey, CA., Feb., 2005. [107](#)
- [74] Y. Qi, D. Liu, D. Dunson, and L. Carin, “Bayesian multi-task compressive sensing with dirichlet process priors,” in *the 25th International Conference on Machine Learning*, Helsinki, Finland, July, 2008. [17](#)
- [75] Y. Zhu, J. D. Zuegel, J. R. Marciante, and H. Wu, “A reconfigurable, multi-Gigahertz pulse shaping circuit based on distributed transversal filters ”, IEEE ISCAS, 2006. [6](#), [7](#)
- [76] Z. Wang, G. R. Arce, B. M. Sadler, J. L. Paredes, S. Hoyos and Z. Yu, “Compressed UWB signal detection with narrowband interference mitigation,” in *IEEE International Conference on Ultra-Wideband, ICUWB*, Singapore, Sept., 2008. [18](#), [33](#), [67](#)

Appendix

Appendix A

Proof of Equations

A.1 Proof of Lemma 3.0.2

Proof. By deleting the j th column vector ϕ_j from Φ_I , the j th coefficient $\hat{\mathbf{x}}_j^I$ in signal $\hat{\mathbf{x}}$ is then become zero. For convenience, we rearrange the matrix $\Phi_I = (\Phi_{I-1} \ \phi_j)$. The order of coefficients of $\hat{\mathbf{x}}_t$ is rearranged but does not affect the values. For convenience, we write ϕ_j as ϕ . Then we use Φ_{I-1} to express projection $\mathbf{y}_I$:

$$\begin{aligned} \mathbf{y}_I &= \mathbf{y} - \mathbf{R}_I \\ &= \begin{pmatrix} \Phi_{I-1} & \phi \end{pmatrix} \begin{pmatrix} \Phi'_{I-1}\Phi_{I-1} & \Phi'_{I-1}\phi \\ \phi'\Phi_{I-1} & \phi'\phi \end{pmatrix}^{-1} \\ &\quad \times \begin{pmatrix} \Phi'_{I-1} \\ \phi' \end{pmatrix} \mathbf{y} \end{aligned} \tag{A.1}$$

On using the Woodbury identity[47], the inverse matrix could be rewritten as

$$\begin{aligned} &\begin{pmatrix} \Phi'_{I-1}\Phi_{I-1} & \Phi'_{I-1}\phi \\ \phi'\Phi_{I-1} & \phi'\phi \end{pmatrix}^{-1} \\ &= \begin{pmatrix} C + C\Psi'\phi D\phi'\Psi C & -C\Psi'\phi D \\ -D\phi'\Psi C & D \end{pmatrix}, \end{aligned} \tag{A.2}$$

where we define

$$\Psi = \Phi_{I-1}; \quad C = (\Psi' \Psi)^{-1}; \quad (\text{A.3})$$

and

$$D = (\Psi' \Psi - \phi \Psi C \Psi \phi)^{-1}. \quad (\text{A.4})$$

Then (A.1) becomes

$$\begin{aligned} \mathbf{y}_I &= \Phi_I (\Phi_I' \Phi_I)^{-1} \Phi_I' \mathbf{y} = \Phi_I A \mathbf{y} \\ &= \begin{pmatrix} \Psi & \phi \end{pmatrix} \begin{pmatrix} C \Psi' + C \Psi' \phi D \phi' \Psi C \Psi' - C \Psi' \phi D \phi' \\ -D \phi' \Psi C \Psi' + D \phi' \end{pmatrix} \mathbf{y}. \end{aligned} \quad (\text{A.5})$$

Clearly, we also have

$$\begin{aligned} \mathbf{y}_{I-1} &= \mathbf{y} - \mathbf{R}_{I-1} \\ &= \Phi_{I-1} (\Phi_{I-1}' \Phi_{I-1})^{-1} \Phi_{I-1}' \mathbf{y} \\ &= \Psi C \Psi' \mathbf{y}. \end{aligned} \quad (\text{A.6})$$

From Eq. (A.6) and expanding Eq. (A.5), after tedious computation, we have

$$\begin{aligned} \mathbf{y}_{I-1} - \mathbf{y}_I &= \mathbf{R}_I - \mathbf{R}_{I-1} \\ &= \alpha \mathbf{y} (\alpha^T)^{-1}, \end{aligned} \quad (\text{A.7})$$

where $\alpha = D \phi' - D \phi' \Psi C \Psi'$. Clearly, α is just the last row vector of the matrix A .

We note that $\hat{\mathbf{x}}_j^I = \alpha y$, then:

$$\|\mathbf{R}_I - \mathbf{R}_{I-1}\|^2 = \frac{(\hat{\mathbf{x}}_j^I)^2}{\langle \alpha_j, \alpha_j^T \rangle}. \quad (\text{A.8})$$

This concludes the proof. □

A.2 Proof of Eq. (4.10) and (4.11)

We first show the derivation of Eq. (4.10), which is given by:

$$\begin{aligned}
& P(s_j^i | \lambda_j^i) \\
&= \int P(s_j^i | \alpha_j^i) P(\alpha_j^i | \lambda_j^i) d\alpha_j^i \\
&= \int \left(\frac{\alpha_j^i}{2\pi}\right)^{-\frac{1}{2}} \exp\left(-\frac{(s_j^i)^2 \alpha_j^i}{2}\right) \lambda_j^i \exp(-\lambda_j^i \alpha_j^i) d\alpha_j^i \\
&= \frac{\lambda_j^i}{(2\pi)^{\frac{1}{2}}} \int (\alpha_j^i)^{-\frac{1}{2}} \exp\left(-\left(\lambda_j^i + \frac{(s_j^i)^2}{2}\right) \alpha_j^i\right) d\alpha_j^i \\
&\text{Let } t = \left(\lambda_j^i + \frac{(s_j^i)^2}{2}\right) \alpha_j^i \\
&= \frac{\lambda_j^i}{(2\pi)^{\frac{1}{2}}} \int \left(\frac{t}{\lambda_j^i + \frac{(s_j^i)^2}{2}}\right)^{\frac{1}{2}} \exp(-t) \\
&\quad d\left(\frac{t}{\lambda_j^i + \frac{(s_j^i)^2}{2}}\right) \\
&= \frac{\lambda_j^i}{(2\pi)^{\frac{1}{2}}} \left(\lambda_j^i + \frac{(s_j^i)^2}{2}\right)^{-\left(\frac{3}{2}\right)} \int t^{\frac{1}{2}} \exp(-t) dt \\
&= (2\pi)^{-\frac{1}{2}} \Gamma\left(\frac{3}{2}\right) \lambda_j^i \left(\lambda_j^i + \frac{(s_j^i)^2}{2}\right)^{-\left(\frac{3}{2}\right)} \tag{A.9}
\end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. We have $\Gamma(\frac{3}{2}) = \int_0^\infty t^{\frac{1}{2}} e^{-t} dt$. Because both distributions belong to the exponential distribution family, the marginal distribution is still in the same family. It is also observed that the marginal distribution $P(s_j^i | \lambda_j^i)$ is sharply peaked at zero, which encourages the sparsity. Therefore, the chosen exponential prior distribution in the hierarchical Bayesian framework can be recognized and encourage the sparsity of the reconstructed signal.

Based on the assumption $\alpha_j^b = \alpha_j^i$, we have the same derivation:

$$\begin{aligned}
& P(s_j^b | \lambda_j^i) \\
&= \int P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^i) d\alpha_j^i \\
&= \int \left(\frac{\alpha_j^i}{2\pi}\right)^{-\frac{1}{2}} \exp\left(-\frac{(s_j^b)^2 \alpha_j^i}{2}\right) \lambda_j^i \exp(-\lambda_j^i \alpha_j^i) d\alpha_j^i \\
&= (2\pi)^{-\frac{1}{2}} \Gamma\left(\frac{3}{2}\right) \lambda_j^i \left(\lambda_j^i + \frac{(s_j^b)^2}{2}\right)^{-\left(\frac{3}{2}\right)} \tag{A.10}
\end{aligned}$$

In order to obtain Eq. (4.11), we utilize the above equations. Then the derivation of the posterior is given by

$$\begin{aligned}
P(\alpha_j^i | s_j^b, \lambda_j^i) &= \frac{P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{P(s_j^b, \lambda_j^i)} \\
&= \frac{P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{\int P(s_j^b | \alpha_j^i) P(\alpha_j^i | \lambda_j^b) d\alpha_j^i} \\
&= \frac{(\alpha_j^i)^{-1} (2\pi)^{-\frac{1}{2}} \lambda_j^i \exp\left(-\frac{(s_j^b)^2 \alpha_j^i}{2} - \alpha_j^i \lambda_j^i\right)}{(2\pi)^{-\frac{1}{2}} \Gamma\left(\frac{3}{2}\right) \left(\lambda_j^i + \frac{(s_j^b)^2}{2}\right)^{-\frac{3}{2}} \lambda_j^i} \\
&= \frac{\left(\lambda_j^i + \frac{(s_j^b)^2}{2}\right)^{\frac{3}{2}} \exp\left(-\left(\lambda_j^i + \frac{(s_j^b)^2}{2}\right) \alpha_j^i\right)}{\Gamma\left(\frac{3}{2}\right)} \\
&= \frac{\left(\tilde{\lambda}_j^i\right)^{\frac{3}{2}} \exp\left(-\tilde{\lambda}_j^i \alpha_j^i\right)}{\Gamma\left(\frac{3}{2}\right)} \tag{A.11}
\end{aligned}$$

So the parameter λ_j^i is updated to $\tilde{\lambda}_j^i$, which is given by

$$\tilde{\lambda}_j^i = \lambda_j^i + \frac{(s_j^b)^2}{2}. \tag{A.12}$$

For transferred multiply reconstructed signal elements $s_j^{b_1}, s_j^{b_2}, \dots, s_j^{b_n}$, The posterior function also belongs to the exponential distribution family. As shown in the Eq.(4.13), the parameter λ_j^i is updated to:

$$\begin{aligned}
& P(\alpha_j^i | s_j^{b_1}, s_j^{b_2}, \dots, s_j^{b_n}, \lambda_j^i) \\
&= \frac{P(s_j^{b_1} | \alpha_j^i) P(s_j^{b_2} | \alpha_j^i) \dots P(s_j^{b_n} | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{P(s_j^{b_1}, s_j^{b_2}, \dots, s_j^{b_n}, \lambda_j^i)} \\
&= \frac{P(s_j^{b_1} | \alpha_j^i) P(s_j^{b_2} | \alpha_j^i) \dots P(s_j^{b_n} | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{\int P(s_j^{b_1}, \alpha_j^i | \lambda_j^i) P(s_j^{b_2}, \alpha_j^i | \lambda_j^i) \dots P(s_j^{b_n}, \alpha_j^i | \lambda_j^i) d\alpha_j^i} \\
&= \frac{P(s_j^{b_1} | \alpha_j^i) P(s_j^{b_2} | \alpha_j^i) \dots P(s_j^{b_n} | \alpha_j^i) P(\alpha_j^i | \lambda_j^i)}{\int P(s_j^{b_1} | \alpha_j^i, \lambda_j^i) P(s_j^{b_2} | \alpha_j^i, \lambda_j^i) \dots P(s_j^{b_n} | \alpha_j^i, \lambda_j^i) P(\alpha_j^i | \lambda_j^i) d\alpha_j^i} \\
&= \frac{\left(\lambda_j^i + \frac{\sum_{i=1}^n (s_j^{b_i})^2}{2} \right)^{\frac{2n+1}{2}} \exp \left(- \left(\lambda_j^i + \frac{\sum_{i=1}^n (s_j^{b_i})^2}{2} \right) \alpha_j^i \right)}{\Gamma(\frac{2n+1}{2})} \\
&= \frac{(\tilde{\lambda}_j^i)^{\frac{2n+1}{2}} \exp \left(- \tilde{\lambda}_j^i \alpha_j^i \right)}{\Gamma(\frac{2n+1}{2})}.
\end{aligned}$$

The distribution $P(s_j^{b_1} | \alpha_j^i), P(s_j^{b_2} | \alpha_j^i), \dots, P(s_j^{b_n} | \alpha_j^i)$ is conditionally independent from each other. In this case, the parameter is updated to:

$$\tilde{\lambda}_j^i = \lambda_j^i + \frac{\sum_{i=1}^n (s_j^{b_i})^2}{2}, \quad (\text{A.13})$$

where n represents the total number of $s_j^{b_1}, s_j^{b_2}, \dots, s_j^{b_n}$.

Therefore, above derivations show that how the single or multiple signal elements $s_j^{b_n}, j = 1, 2, \dots, N, n = 1, 2, \dots$ from the other BCS procedures update the hyperparameters in the i -th BCS signal reconstruction procedure.

A.3 Derivation of Eq. (4.16)

One strategy to maximizing the target log function is to exploit an EM method, treating the $\mathbf{s}^i$ as hidden data and maximize the following expectation:

$$E_{\mathbf{s}^i | \mathbf{y}^i, \boldsymbol{\alpha}^i} [\log P(\mathbf{s}^i | \boldsymbol{\alpha}^i) P(\boldsymbol{\alpha}^i | \boldsymbol{\lambda}^i)]. \quad (\text{A.14})$$

The operator $E_{s^i|y^i, \alpha^i}$ denotes an expectation of the posterior $P(s^i|y^i, \alpha^i, \lambda^i, \beta)$ with respect to the distribution over the s^i given the data and hidden variables. Through differentiation with respect to α^i we get:

$$\begin{aligned}
& \frac{\partial}{\partial \alpha_j^i} E_{s^i|y^i, \alpha^i} [\log P(s^i|\alpha^i, \beta) P(\alpha^i|\lambda^i)] \\
&= E_{s^i|y^i, \alpha^i} \left[\frac{\partial}{\partial \alpha_j^i} (\log P(s^i|\alpha^i, \beta) + \log P(\alpha^i|\lambda^i)) \right] \\
&= -\frac{1}{2} E_{s^i|y^i, \alpha^i} \left[-2(s_j^i)^2 - 4\lambda_j^i + \frac{6}{\alpha_j^i} \right] \\
&= E_{s^i|y^i, \alpha^i} ((s_j^i)^2) + 2\lambda_j^i - \frac{3}{\alpha_j^i}.
\end{aligned} \tag{A.15}$$

According to Eq.(4.3), we have:

$$E_{s^i|y^i, \alpha^i} ((s_j^i)^2) = \Sigma_{jj}^i + (\mu_j^i)^2 \tag{A.16}$$

We set Eq.(A.15) to 0, which yields to an update for α_j^i :

$$\alpha_j^i = \frac{3}{(s_j^i)^2 + \Sigma_{jj}^i + 2\lambda_j^i}. \tag{A.17}$$

A.4 Derivation of Eq. (4.20)

For the $L_1(\alpha)$, as shown in [38], we have:

$$\begin{aligned}
L_1(\alpha) &= -\frac{1}{2} (N \log 2\pi + \log |E| + y^T E^{-1} y) \\
&= L_1(\alpha_{-j}) + l_1(\alpha_j)
\end{aligned} \tag{A.18}$$

where,

$$\begin{aligned}
E &= \sigma^2 I + \Phi A^{-1} \Phi^T \\
&= \sigma^2 I + \sum_{k \neq j} \alpha_k^{-1} \phi_k \phi_k^{-1} + \alpha_j^{-1} \phi_j \phi_j \\
&= E_{-j} + \alpha_j^{-1} \phi_j \phi_j
\end{aligned} \tag{A.19}$$

and

$$L_1(\alpha_{-i}) = -\frac{1}{2}(N \log 2\pi + \log |E_{-j}| + y^T E_{-j}^{-1} y) \tag{A.20}$$

$$l_1(\alpha_j) = \frac{1}{2}(\log \alpha_j - \log(\alpha_j + g_j) + \frac{h_j^2}{\alpha_j + g_j}) \tag{A.21}$$

The quantities g_j , h_j and E_{-j} are given by

$$g_j = \phi_j^T E_{-j}^{-1} \phi_j \tag{A.22}$$

$$h_j = \phi_j^T E_{-j}^{-1} y \tag{A.23}$$

$$E_{-j} = \beta^2 I + \sum_{k \neq j} \alpha_k^{-1} \phi_k \phi_k^{-1} \tag{A.24}$$

where ϕ_j is the j -th column vector of the matrix Φ .

In order to find the critical point, the differentiation of $l_1(\alpha_j)$ is given by:

$$\frac{\partial l_1(\alpha_j)}{\partial \alpha_j} = \frac{\alpha_j^{-1} g_j^2 - (h_j^2) + g_j}{2(\alpha_j + g_j)^2} = 0 \tag{A.25}$$

It is easy to maximize $l_1(\alpha_j)$ with respect to α_j by taking the first and second derivatives. Then the maximum point α_j is given by:

$$\alpha_j^* = \begin{cases} \frac{h_j^2}{g_j^2 - h_j}, & \text{if } g_j^2 > h_j; \\ \alpha, & \text{otherwise.} \end{cases} \tag{A.26}$$

The second derivative is:

$$\frac{\partial^2 l_1(\alpha_j)}{\partial \alpha_j^2} = \frac{-2\alpha_j^2(\alpha_j + g_j)(\alpha_j^{-1}g_j^2 - h_j^2 + g_j) - (\alpha_j + g_j)^2 g_j^2}{2\alpha_j^2(\alpha_j + g_j)^4} \quad (\text{A.27})$$

Taking the critical point α_j^* into the second derivative expression, we have know that:

$$\frac{\partial^2 l_1(\alpha_j = \alpha_j^*)}{\partial \alpha_j^2} = \frac{-g_j^2}{2\alpha_j^2(\alpha_j + g_j)^2} \quad (\text{A.28})$$

Obviously, it is always negative, and therefore function $l_1(\alpha_j)$ achieves the maximum at α_j^* , which is unique.

A.5 Derivations about Eq.(4.27) and Eq.(4.28)

The first derivative of the $l_2(\alpha_j)$ is $l'_2(\alpha_j) = -\lambda_j$. All together the first differentiation of the posterior $l(\alpha_j)$ is given by:

$$\begin{aligned} l'(\alpha_j) &= l'_1(\alpha_j) + l'_2(\alpha_j) \\ &= \left(\frac{g_j}{2\alpha_j(\alpha_j + g_j)} - \frac{h_j^2}{2(\alpha_j + g_j)^2} \right) - \lambda_j \\ &= \frac{1}{2} \left[\frac{1}{\alpha_j} - \frac{1}{\alpha_j + g_j} - \frac{h_j^2}{(\alpha_j + g_j)^2} - 2\lambda_j \right] \end{aligned} \quad (\text{A.29})$$

By setting the Eq.(A.29) to zero, we can find the optimum α_j^* for Eq. (4.28).

The g_j and h_j^2 are not negative based on Eq. (A.22) and (A.23). We have $\alpha_j \geq 0$ and $\lambda_j > 0$ according to the exponential distribution as shown in Eq.(4.8), and $l'(\alpha_j) \rightarrow -2\lambda_j < 0$ as $\alpha_j \rightarrow +\infty$. Then, it has $l'(\alpha_j) > 0$ when $\alpha_j \rightarrow 0$. Therefore, for the function $l'(\alpha_j) = 0$, it has at least one positive root for $\alpha_j > 0$.

We rearrange Eq.(A.29) to

$$\begin{aligned} l'(\alpha_j) &= \frac{1}{2} \left[\frac{1}{\alpha_j} - \frac{1}{\alpha_j + g_j} - \frac{h_j^2}{(\alpha_j + g_j)^2} - 2\lambda_j \right] \\ &= \frac{f(\alpha_j, g_j, h_j, \lambda_j)}{\alpha_j(\alpha_j + g_j)^2} \end{aligned} \quad (\text{A.30})$$

Setting Eq.(A.54) to zero is to let the numerator be zero, i.e., $f(\alpha_j, g_j, h_j, \lambda_j) = 0$. To find the solution, we normalize the equation to reduce one parameter for convenience. Then we need to solve

$$\frac{f(\alpha_j, g_j, h_j, \lambda_j)}{-2\lambda_j} = \alpha_j^3 + B_0\alpha_j^2 + B_1\alpha_j + B_2 = 0 \quad (\text{A.31})$$

The corresponding coefficients are given by [68]:

$$B_0 = 2g_j \quad (\text{A.32})$$

$$B_1 = \frac{g_j - 2\lambda_j g_j^2 - h_j^2}{-2\lambda_j} \quad (\text{A.33})$$

$$B_2 = \frac{g_j^2}{-2\lambda_j} \quad (\text{A.34})$$

To solve the cubic function, we define intermediate components as:

$$U = 2B_0^3 - 9B_0B_1 + 27B_2 \quad (\text{A.35})$$

$$V = (2B_0^3 - 9B_0B_1 + 27B_2)^2 - 4(B_0^2 - 3B_1)^3 \quad (\text{A.36})$$

Then the solutions of the cubic function are given by:

$$x_1 = -\frac{1}{3}(B_0 + \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.37})$$

$$x_2 = -\frac{1}{3}(B_0 + \omega_1 \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \omega_2 \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.38})$$

$$x_3 = -\frac{1}{3}(B_0 + \omega_2 \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \omega_1 \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.39})$$

where,

$$\omega_1 = -\frac{1}{2} + \frac{\sqrt{3}}{2}i \quad (\text{A.40})$$

$$\omega_2 = -\frac{1}{2} - \frac{\sqrt{3}}{2}i \quad (\text{A.41})$$

Therefore, all those three roots x_1 , x_2 and x_3 are critical points of the optimization function shown in Eq.(4.20). We choose the positive root which maximizes the optimization function in Eq.(4.20) as the optimum solution α_j^* for Eq.(4.28).

A.6 Prior Information

Taking the j -th element as an example, we show how the parameters a_j^{ik} and b_j^{ik} are updated based on the signal element u_j^{cd} . First, we prove Eq. (5.7):

$$\begin{aligned}
& P(\mathbf{u}^{cd} | \mathbf{a}^{ik}, \mathbf{b}^{ik}) \\
&= \int P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik}) d\alpha_j \\
\text{Let } u_j^{cd} &= u, \quad a_j^{ik} = a, \quad b_j^{ik} = b; \\
&= \int \frac{b^a (\alpha_j)^{a-1} \exp(-b\alpha_j)}{\Gamma(a)} \left(\frac{\alpha_j}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{u^2 \alpha_j}{2}\right) d\alpha_j \\
&= \frac{b^a}{\Gamma(a)(2\pi)^{\frac{1}{2}}} \int (\alpha_j)^{a-\frac{1}{2}} \exp\left(-\left(b + \frac{u^2}{2}\right)\alpha_j\right) d\alpha_j \\
\text{Let } t &= \left(b + \frac{u^2}{2}\right)\alpha_j \\
&= \frac{b^a}{\Gamma(a)(2\pi)^{\frac{1}{2}}} \times \\
&\quad \int \left(\frac{t}{b + \frac{u^2}{2}}\right)^{(a-\frac{1}{2})} \exp(-t) d\left(\frac{t}{b + \frac{u^2}{2}}\right) \\
&= \frac{b^a}{\Gamma(a)(2\pi)^{\frac{1}{2}}} \int \left(\frac{1}{b + \frac{u^2}{2}}\right)^{(a+\frac{1}{2})} t^{(a-\frac{1}{2})} \exp(-t) dt \\
&= \frac{b^a \left(b + \frac{u^2}{2}\right)^{(-a-\frac{1}{2})}}{\Gamma(a)(2\pi)^{\frac{1}{2}}} \int t^{(a-\frac{1}{2})} \exp(-t) dt \\
&= \frac{b^a \left(b + \frac{u^2}{2}\right)^{(-a-\frac{1}{2})}}{\Gamma(a)(2\pi)^{\frac{1}{2}}} \cdot \Gamma\left(a + \frac{1}{2}\right) \\
&= \frac{(b_j^{ik})^{a_j^{ik}} \Gamma(a_j^{ik} + \frac{1}{2})}{(2\pi)^{\frac{1}{2}} \Gamma(a_j^{ik})} \left(b_j^{ik} + \frac{(u_j^{cd})^2}{2}\right)^{-(a_j^{ik} + \frac{1}{2})}
\end{aligned} \tag{A.42}$$

Then the derivation of the posterior based on Eq. (5.7) is given by

$$\begin{aligned}
& P(\alpha_j | u_j^{cd}, a_j^{ik}, b_j^{ik}) \\
&= \frac{P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{P(u_j^{cd}, a_j^{ik}, b_j^{ik})} \\
&= \frac{P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{\int P(u_j^{cd} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik}) d\alpha_j} \\
&= \left(\frac{\alpha_j}{2\pi}\right)^{1/2} \exp\left(-\frac{(u_j^{cd})^2 \alpha_j}{2}\right) \times \\
&\quad \frac{(b_j^{ik})^{a_j^{ik}} (\alpha_j)^{(a_j^{ik}-1)} \exp(-b_j^{ik} \alpha_j)}{\Gamma(a_j^{ik})} \times \\
&\quad \frac{1}{\frac{(b_j^{ik})^{a_j^{ik}} \Gamma(a_j^{ik} + \frac{1}{2})}{(2\pi)^{\frac{1}{2}} \Gamma(a_j^{ik})} \left(b_j^{ik} + \frac{(u_j^{cd})^2}{2}\right)^{-(a_j^{ik} + \frac{1}{2})}} \\
&= \frac{\left(b_j^{ik} + \frac{(u_j^{cd})^2}{2}\right)^{(a_j^{ik} + \frac{1}{2})} (\alpha_j)^{(a_j^{ik} - \frac{1}{2})} \exp\left(-\left(b_j^{ik} + \frac{(u_j^{cd})^2}{2}\right) \alpha_j\right)}{\Gamma(a_j^{ik} + \frac{1}{2})} \\
&= \frac{\left(\tilde{b}_j^{ik}\right)^{\tilde{a}_j^{ik}} (\alpha_j)^{(\tilde{a}_j^{ik}-1)} \exp\left(-\tilde{b}_j^{ik} \alpha_j\right)}{\Gamma(\tilde{a}_j^{ik})},
\end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function, defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. Obviously, we can see that the posterior is still the gamma distribution with updated parameters by comparing Eq. (A.43) and Eq. (5.6). Because both distributions belong to the exponential distribution family, the joint distribution is still in the same family. So the parameters are updated to,

$$\tilde{a}_j^{ik} = a_j^{ik} + \frac{1}{2} \quad (\text{A.43})$$

$$\tilde{b}_j^{ik} = b_j^{ik} + \frac{(u_j^{cd})^2}{2}. \quad (\text{A.44})$$

For the transferred reconstructed signal elements $u_j^{c_1}, u_j^{c_2}, \dots, u_j^{c_n}$, the posterior function also belongs to the exponential distribution family, which is given by

$$\begin{aligned}
& P(\alpha_j | u_j^{c_1}, u_j^{c_2}, \dots, u_j^{c_n}, a_j^{ik}, b_j^{ik}) \\
&= \frac{P(u_j^{c_1} | \alpha_j) P(u_j^{c_2} | \alpha_j) \dots P(u_j^{c_n} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{P(u_j^{c_1}, u_j^{c_2}, \dots, u_j^{c_n}, a_j^{ik}, b_j^{ik})} \\
&= \frac{P(u_j^{c_1} | \alpha_j) P(u_j^{c_2} | \alpha_j) \dots P(u_j^{c_n} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{\int P(u_j^{c_1}, \alpha_j | a_j^{ik}, b_j^{ik}) \dots P(u_j^{c_n}, \alpha_j | a_j^{ik}, b_j^{ik}) d\alpha_j} \\
&= \frac{P(u_j^{c_1} | \alpha_j) P(u_j^{c_2} | \alpha_j) \dots P(u_j^{c_n} | \alpha_j) P(\alpha_j | a_j^{ik}, b_j^{ik})}{\int P(u_j^{c_1} | \alpha_j, a_j^{ik}, b_j^{ik}) \dots P(u_j^{c_n} | \alpha_j, a_j^{ik}, b_j^{ik}) P(\alpha_j | a_j^{ik}, b_j^{ik}) d\alpha_j} \\
&= \frac{\left(b_j^{ik} + \frac{\sum_{i=1}^n (u_j^{c_i})^2}{2}\right)^{(a_j^{ik} + \frac{n}{2})} (\alpha_j)^{(a_j^{ik} + \frac{n}{2} - 1)}}{\Gamma(a_j^{ik} + \frac{n}{2})} \times \\
&\quad \exp\left(-\left(b_j^{ik} + \frac{\sum_{i=1}^n (u_j^{c_i})^2}{2}\right) \alpha_j\right) \\
&= \frac{\left(\tilde{b}_j^{ik}\right)^{\tilde{a}_j^{ik}} (\alpha_j)^{(\tilde{a}_j^{ik} - 1)} \exp\left(-\tilde{b}_j^{ik} \alpha_j\right)}{\Gamma(\tilde{a}_j^{ik})}.
\end{aligned}$$

The distribution $P(u_j^{c_1} | \alpha_j), P(u_j^{c_2} | \alpha_j), \dots, P(u_j^{c_n} | \alpha_j)$ are conditionally independent of each other. In this case, the parameters are updated to

$$\tilde{a}_j^{ik} = a_j^{ik} + \frac{n}{2} \quad (\text{A.45})$$

$$\tilde{b}_j^{ik} = b_j^{ik} + \frac{\sum_{i=1}^n (u_j^{c_i})^2}{2}. \quad (\text{A.46})$$

where n represents the total number of $u_j^{c_1}, u_j^{c_2}, \dots, u_j^{c_n}$.

A.7 Proof of Proposition 1

Proof. In order to prove Proposition 1, we first expand the marginal log-likelihood function, which is given by

$$L(\alpha) = -\frac{1}{2}(N \log 2\pi + \log |E| + y^T E^{-1} y). \quad (\text{A.47})$$

where,

$$E = \beta^2 I + \Phi A^{-1} \Phi^T, \quad (\text{A.48})$$

The term $\log |E|$ is expanded to

$$\begin{aligned} \log |E| &= \log |\beta^2 I + \Phi A^{-1} \Phi^T| \\ &= \log (|\beta^{-2} I| |A + \beta^2 \Phi^T \Phi|) \\ &= -2N \log \beta + \log |A + \beta^2 \Phi^T \Phi|. \end{aligned} \quad (\text{A.49})$$

Clearly, this term, $\log |E|$, is associated with the projection matrix Φ and the hyperparameter matrix A , which is not directly related to the measurements. Hence, it does not contain any information about measurements and observations. Then in order to expand the term $y^T E^{-1} y$, we utilize the Woodbury Inverse Identity [47] to expand the term E^{-1} , which is given by

$$\begin{aligned} E^{-1} &= (\beta^2 I + \Phi A^{-1} \Phi^T)^{-1} \\ &= \beta^{-2} I - \beta^{-2} \Phi (A + \beta^{-2} \Phi^T \Phi)^{-1} \Phi^T \beta^{-2}. \end{aligned} \quad (\text{A.50})$$

And we have

$$\begin{aligned}
y^T E^{-1} y &= y^T (\beta^2 I + \Phi A^{-1} \Phi^T)^{-1} y \\
&= y^T \beta^{-2} I y - y^T \beta^{-2} \Phi (A + \beta^{-2} \Phi^T \Phi)^{-1} \Phi^T \beta^{-2} y \\
&= \beta^{-2} y^T y - \beta^{-2} y^T \Phi \Sigma \Phi^T \beta^{-2} y \\
&= \beta^{-2} y^T y - \beta^{-2} y^T \Phi \hat{u} \\
&= \beta^{-2} y^T (y - \hat{y}) \\
&= \beta^{-2} (\|y\| - y^T \hat{y}),
\end{aligned} \tag{A.51}$$

where $\hat{u}$ is the estimated signal vector at the current iteration with the current nonzero index set Ω . And correspondingly, $\hat{y}$ is the corresponding measurements at the current iteration. After several iterations, the index set Ω augments until all nonzero signal elements are found and added into Ω . Then the algorithm converges and we have $\hat{y} \rightarrow y$, $\hat{u} \rightarrow u$ and correspondingly the term $\beta^{-2} y^T (y - \hat{y}) \rightarrow 0$.

Based on above analysis, the log-likelihood function Eq. (A.47) can be written as:

$$\begin{aligned}
L(\alpha) &= -\frac{1}{2} [N \log 2\pi + \sum \log(\beta) + \sum \log(\alpha) \\
&\quad + \log |A^{-1} + \beta \Phi^T \Phi| + \beta y^T (y - \Phi \tilde{u})] \\
&\approx C - \frac{1}{2} \beta (\|y\| - y^T \hat{y}) \\
&\propto -(\|y\| - y^T \hat{y}).
\end{aligned} \tag{A.52}$$

where C is the term which is not associated with the measurement vector y .

Obviously, we have

$$\begin{aligned}
[j] &= \arg \min_{\alpha_j} (\|y\| - y^T \hat{y}) \\
&= \arg \max_{\alpha_j} (L(\alpha)).
\end{aligned} \tag{A.53}$$

Therefore, the j -th hyperparameter α_j which can maximize the log-likelihood function can also minimize the "residual" term, i.e., $(\|y\| - y^T \hat{y})$. Hence, the j -th index should be selected and added from the candidate index set Λ to the nonzero index set Ω at that iteration. So Proposition 1 is proved. $\square$

A.8 Maximum Posterior

When external prior information is incorporated, the first order derivative of $l(\alpha_j)$ is given by

$$\begin{aligned} l'(\alpha_j) &= \frac{g_j}{2\alpha_j(\alpha_j + g_j)} - \frac{h_j^2}{2(\alpha_j + g_j)^2} + \frac{a_j}{\alpha_j} - b_j \\ &= \frac{f(\alpha_j, g_j, h_j, a_j, b_j)}{\alpha_j(\alpha_j + g_j)^2}, \end{aligned} \quad (\text{A.54})$$

where $f(\alpha_j, g_j, h_j, a_j, b_j)$ is a cubic function with respect to α_j . By setting Eq. (A.54) to zero, the root is the optimum α_j^* . We know that g_j, h_j^2 are not negative based on their definition. And we also have $\alpha_j \geq 0$ and $a_j > 0, b_j > 0$. Then based on Eq. (A.54), we have $l'(\alpha_j) \rightarrow -b_j < 0$ as $\alpha_j \rightarrow +\infty$. In addition, it has $l'(\alpha_j) > 0$ when $\alpha_j \rightarrow 0$. Therefore, for the function $l'(\alpha_j) = 0$, it has at least one positive root $\alpha_j > 0$.

So we have to solve,

$$\begin{aligned} l'(\alpha_j) &= 0 \\ \Rightarrow f(\alpha_j, g_j, h_j, a_j, b_j) &= A_0\alpha_j^3 + A_1\alpha_j^2 + A_2\alpha_j + A_3 = 0 \\ \Rightarrow \alpha_j^3 + B_0\alpha_j^2 + B_1\alpha_j + B_2 &= 0 \end{aligned}$$

where we normalize the equation to reduce one parameter A_0 for simplicity by replacing $B_0 = \frac{A_1}{A_0}, B_1 = \frac{A_2}{A_0}$, and $B_2 = \frac{A_3}{A_0}$. The corresponding coefficients are given

by

$$B_0 = \frac{(2a_j - 4b_j g_j)}{-2b_j} \quad (\text{A.55})$$

$$B_1 = \frac{g_j + 4a_j g_j - 2b_j g_j^2 - h_j^2}{-2b_j} \quad (\text{A.56})$$

$$B_2 = \frac{(1 + 2a_j)g_j^2}{-2b_j}. \quad (\text{A.57})$$

To solve the cubic function, we define the intermediate components as

$$U = 2B_0^3 - 9B_0B_1 + 27B_2 \quad (\text{A.58})$$

$$V = (2B_0^3 - 9B_0B_1 + 27B_2)^2 - 4(B_0^2 - 3B_1)^3. \quad (\text{A.59})$$

Then the solutions of the cubic function [68] are given by:

$$\alpha_j^{(1)} = -\frac{1}{3}(B_0 + \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.60})$$

$$\alpha_j^{(2)} = -\frac{1}{3}(B_0 + \omega_1 \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \omega_2 \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.61})$$

$$\alpha_j^{(3)} = -\frac{1}{3}(B_0 + \omega_2 \sqrt[3]{\frac{U + \sqrt{V}}{2}} + \omega_1 \sqrt[3]{\frac{U - \sqrt{V}}{2}}) \quad (\text{A.62})$$

with

$$\omega_1 = -\frac{1}{2} + \frac{\sqrt{3}}{2}i \quad (\text{A.63})$$

$$\omega_2 = -\frac{1}{2} - \frac{\sqrt{3}}{2}i. \quad (\text{A.64})$$

Therefore, the three roots $\alpha_j^{(1)}$, $\alpha_j^{(2)}$ and $\alpha_j^{(3)}$ are critical points of the optimization function shown in Eq. (5.20). We choose the positive root which maximizes the optimization function in Eq. (5.20) as the optimum solution α_j^* for Eq. (5.22).

Vita

Depeng Yang was born April 7, 1979 at Yinchuan, Ninxia, China. He received B.S., M.S. degrees in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2006, respectively, and PhD degree in electrical engineering at the University of Tennessee, Knoxville, TN, in 2011. His research interests and experience span a wide range of topics including statistical signal processing, sparse Bayesian learning, super computing on FPGAs and GPUs, and RF/UWB radar systems. He had authored/coauthored more than 25 Journal/conference papers. He is the recipient of 2011 University of Tennessee Chancellors Citation for Extraordinary Professional Promise. He has also served as session chairs of conferences, the invited reviewers for IEEE/EURASIP signal processing journals and other international conferences.