SFT-BASED DOA DETECTION ON CO-PRIME ARRAY

BY GUANJIE HUANG

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Athina Petropulu

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2015

ABSTRACT OF THE THESIS

SFT-BASED DOA DETECTION ON CO-PRIME ARRAY

by Guanjie Huang

Thesis Director: Athina Petropulu

In the thesis, we propose a direction of arrival (DOA) estimation method with co-prime sensor arrays based on the Sparse Fourier Transform (SFT). A co-prime array is composed of two uniform linear arrays (ULAs) whose inter-sensor spacings are $M\lambda/2$ and $N\lambda/2$, respectively, where M and N are co-prime integers and the λ represents the signal wavelength. The co-prime array is adopted here because it can extend the degrees of freedom thus benefiting DOA detection. Assuming that there are not many targets in the array far field, the array snapshot is sparse in the spatial frequency domain. Since the spatial frequencies of the array snapshot contain DOA information, the DFT is traditionally used to obtain DOA information. In the thesis, the SFT is employed to estimate the DOA instead of the DFT. Compared to the DFT, the SFT only needs a small subset of snapshot samples to estimate the significant coefficients. Owing to this advantage, we design a new method to estimate DOA, using a subset of sensors. Both analytical and computer simulations confirm the validity of the proposed approach.

Acknowledgements

I would never have been able to finish my thesis without the guidance of my committee members, the inspiration of previous publications, and the help from friends.

I would like to express my deepest gratitude to my advisor, Prof. Athina Petropulu, for her excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I would also like to thank Bo Li for guiding my research for the past one year and helping me to develop my background in radar system.

And many thanks to Shunqiao Sun, Valerie Yang, who was always willing to help and give best suggestions. It would have been a lonely lab without them.

Finally, I would like to thank my parents. They were always supporting me and encouraging me with their best wishes.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Discrete Fourier Transform (DFT) can convert the discrete time signal samples from the original domain (often time or position along a line) to the sampled frequency domain. The DFT is obtained by decomposing a list of values into components of different frequencies. The DFT coefficients represent magnitude and phase of the different frequency components. This operation is useful in many applications but computing it directly from the definition is often too slow to be practical. The FFT is a algorithm to compute the same result more quickly and efficiently; computing the DFT of N points using the definition takes $\mathcal{O}(N^2)$ complex multiplications, while an FFT can compute the same DFT in only $\mathcal{O}(\frac{N}{2}\log_2 N)$ complex multiplications. The difference in speed can be enormous, especially for long data sets where N may be in the thousands or millions.

Recently, a new tool known as the Sparse Fourier Transform (SFT), has been developed by [1, 2, 3] to compute the frequency content of signals which are sparse in the frequency domain. The SFT has a ability to use partial time domain samples to estimate the significant Fourier coefficients, as opposed to computing all Fourier coefficient and then determining the largest values. So this algorithm can greatly reduce the computation time and the required storage as compared with FFT.

In this thesis, the SFT is applied to detect direction of arrival (DOA) on a sensor array. In the DFT of the array snapshot (spatial frequency domain), the location of the peaks are directly related to DOAs of the targets. An array snapshot contains the array measurements at a specific

time. If there is a small number of targets as compared to the number of receive sensors, the array snapshot will be sparse in the spatial frequency domain. Thus, the SFT can be used on the snapshot to detect the spatial frequencies corresponding to targets. Since the SFT only needs to use partial samples, just a subset of the sensors are need. That is to say, we can save some sensors by using SFT-based DOA detection.

The co-prime array consists of two ULAs which have N and $2M-1$ sensors, respectively. Their inter-sensor spacing are $M\lambda/2$ and $N\lambda/2$, where M and N are co-prime integers and λ represents the transmit signal wavelength. The co-prime array can improve resolution by achieving higher degrees of freedom (DOFs). There are two kinds of approaches: sum co-prime array and difference co-prime array. The difference co-prime array can achieve $2MN+1$ DOFs by using $N+2M-1$ physical sensors. The sum co-prime array only achieves $2MN-2M-N$ DOFs [4]. In this thesis, the difference co-prime array is employed to increase the resolution of the DOA detection. Since the samples for DOA estimation are randomly generated, the saving sensors are different in each estimation. Hence, we try to implement the SFT-based DOA estimation with pseudorandom samples to make the saving sensors fixed.

In the next chapter, we review the basic concepts and theorems which are necessary to understand the how the SFT benefits us. In Chapter 3, we introduce the SFT algorithm to show how it works and how fast it is. Chapter 4 introduces the sensor array and the implementation of the SFT-based DOA detection using a co-prime array. Then we introduce the SFT with pseudorandom samples in chapter 5. Chapter 6 concludes the paper.

Chapter 2

Basic Concepts of Discrete Fourier transform

In this chapter, we introduce the useful concepts and theorems required to understand what the Discrete Fourier Transform and Sparse Fourier Transform aim to do. Also, we will briefly speak about the Fast Fourier Transform, which is an algorithm to compute the DFT and inverse DFT.

2.1 The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) converts the discrete time signal samples from their original domain to the sampled frequency domain. The DFT coefficients represent magnitude and phase of the different frequency components.

The Discrete Fourier Transform of a signal x is defined by

$$X[k] \triangleq \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-2\pi i k n/N}, \qquad k = 0, 1, 2, 3, \ldots, N-1, \tag{2.1}$$

where $\triangleq$ means "is defined as" or "equals by definition", k is used to represent the frequency domain ordinal, and n is used to denote the time domain ordinal. The N is the length of the sequence to be transformed. $X[k]$ is the Fourier coefficient of $x[n]$ at frequency k.

The inverse DFT is given by

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{2\pi i k n/N}, \qquad n = 0, 1, 2, 3, \ldots, N-1. \tag{2.2}$$

It can be shown that there is a one-to-one relationship between the two kind of representations.

We write

$$X[k] \overset{\mathscr{F}}{\longleftrightarrow} x[n]. \tag{2.3}$$

Also, the DFT is almost an isometric map:

$$\mathbb{C}^N \overset{\mathscr{F}}{\longleftrightarrow} \mathbb{C}^N, \tag{2.4}$$

enjoying several properties:

1. Time Shifting

$$x[n - n_0] \overset{\mathscr{F}}{\longleftrightarrow} X[k]e^{-2\pi ikn_0/N}; \tag{2.5}$$

2. Time Scaling

$$x[an] \overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{|a|}X[a^{-1}k]; \tag{2.6}$$

2.2 The Fast Fourier Transform

The FFT is a popular algorithm to compute the DFT and its inverse. The traditional DFT implementation takes $\mathcal{O}(N^2)$ arithmetical operations to compute the N coefficients, while the FFT can compute the same coefficients in only $\mathcal{O}(\frac{N}{2}\log N)$. Obviously, the FFT can make a big difference when the data is very long.

There are many algorithmic variants of the FFT. The widespread one is the Cooley-Tukey (CT) Algorithm. And the most common form of the CT algorithm is the radix-2 decimation-in-time (DIT) FFT. For Equation (2.1), the radix-2 DIT FFT firstly computes the DFT of the even-indexed data ($x[2m] = x[0], x[2], \ldots, x[N-2], m \in \mathbb{N}^+$) and of the odd-indexed data ($x[2m+1] = x[1], x[2], \ldots, x[N-1], m \in \mathbb{N}^+$), respectively. Then combining the two results,

we obtain the DFT of the whole data, as follow:

$$X(k) = \sum_{m=0}^{N/2-1} x[2m]e^{-2\pi i(2m)k/N} + \sum_{m=0}^{N/2-1} x[2m+1]e^{-2\pi i(2m+1)k/N} \tag{2.7a}$$

$$= \sum_{m=0}^{N/2-1} x[2m]e^{\frac{-2\pi imk}{N/2}} + e^{\frac{-2\pi ik}{N}} \sum_{m=0}^{N/2-1} x[2m+1]e^{\frac{-2\pi imk}{N/2}} \tag{2.7b}$$

$$= \mathscr{F}_{even}[k] + e^{\frac{-2\pi ik}{N}} \mathscr{F}_{odd}[k], \qquad k = 0, 1, \ldots, N-1. \tag{2.7c}$$

Here $\mathscr{F}_{even}[k]$ and $\mathscr{F}_{odd}[k]$ represent the Fourier transform of the even-indexed and odd-indexed part of $x[n]$, respectively. Since $\mathscr{F}_{even}[k]$ and $\mathscr{F}_{odd}[k]$ are periodic with period $N/2$, we have $\mathscr{F}[k + N/2] = \mathscr{F}[k]$. In addition, the factor $e^{\frac{-2\pi i(k+N/2)}{N}} = -e^{\frac{-2\pi ik}{N}}$. So Equation (2.7) can be written as

$$X[k] = \mathscr{F}_{even}[k] + e^{\frac{-2\pi ik}{N}} \mathscr{F}_{odd}[k], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{2.8}$$

$$X[k + N/2] = \mathscr{F}_{even}[k] - e^{\frac{-2\pi ik}{N}} \mathscr{F}_{odd}[k], \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{2.9}$$

Now we only need to take $2(N/2)^2 + N/2$ complex multiplications instead of N^2 by splitting N-points DFT into two $N/2$-points DFTs. Then doing the split recursively until we have the 2-points DFTs. So the number of complex multiplications is:

$$M(N) = 2M(\frac{N}{2}) + \frac{N}{2} \tag{2.10a}$$

$$= 2(2M(\frac{N}{4}) + \frac{N}{4}) + \frac{N}{2} \tag{2.10b}$$

$$= 4M(N/4) + \frac{N}{2} \tag{2.10c}$$

$$\ldots \tag{2.10d}$$

$$= NM(1) + \frac{N}{2}\log_2 N \tag{2.10e}$$

$$= \frac{N}{2}\log_2 N, \tag{2.10f}$$

Where $M[N]$ denotes the the number of complex multiplications needed for N-points DFT. For each 2-points DFT no multiplication is required, i.e., $M(1) = 0$. Hence, we can achieve $\mathcal{O}(\frac{N}{2}\log N)$ at last. As shown in Figure 2.1, FFT only takes few operations compared with DFT. Already for $N = 2^{10} = 1024$, FFT makes a big difference.

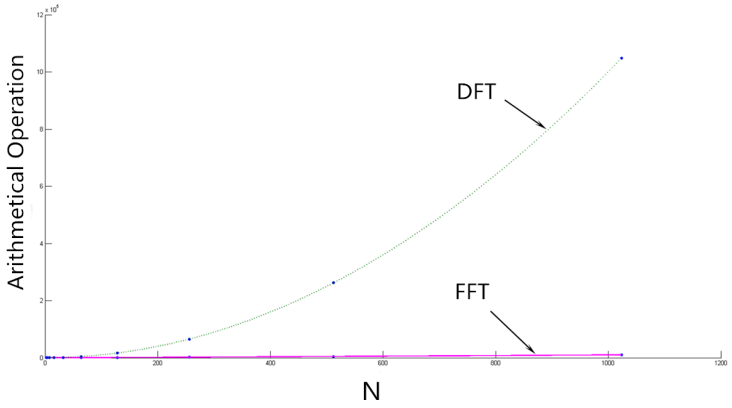

Figure 2.1: The number of arithmetical operation required for DFT and FFT

Fewer complex multiplications always translates to less computation time. So FFT is much faster than traditional DFT. Currently, it is broadly applied to various fields, such as signal processing and image/audio/video compression. Hence, if there exists one algorithm, which is even faster than the FFT, it will cause a significant impact in applications. The next chapter will introduce such an algorithm, named Sparse Fourier Transform.

Chapter 3

Sparse Fourier Transform

As we described before, the Discrete Fourier Transform can be written as Equation (2.1). However, in many cases, most of these Fourier coefficients are zero. To obtain the significant coefficients, we have to compute all Fourier coefficients. For computing each coefficient, all the input data will be used. Even when using the FFT, this wastes time. Hence, the FFT is not fast enough in the sparse Fourier coefficients case, especially for the big data applications where the data size can easily go beyond terabytes.

The Sparse Fourier Transform (SFT), to be discribed next, applies to signals that are sparse in the frequency domain. It uses a small fraction of the data to estimate the Fourier coefficients of the non-zero frequencies. A simple method, proposed by [3], to implement SFT, is presented next. The method consists of 4 steps:

1) Implement random binning and identify the non-zero frequencies ω in each bin.
2) Estimate the Fourier coefficient, c_ω, of each frequency, identified in the first step.
3) Out of all pairs (ω, c_ω), and retain the m pairs whose coefficients have the greatest magnitude. (m is the number of frequency components in the signal, assumed to be know apriori.)
4) Compute the residual data and go to step 1.

Obviously, the most significant advantage of SFT is that it is very fast for processing big data, which are sparse in the frequency domain. This is because it uses only a small number of data samples and estimates the non-zero Fourier coefficients only. That is to say, the SFT does not waste time to compute the zero coefficients.

3.1 Single Frequency Identification

Assuming that the input data is a vector of the following form:

$$x[t] = c_\omega e^{2\pi i \omega t/N}, \quad t = 0, 1, 2, 3, \ldots, N-1, \tag{3.1}$$

where c_ω represents the Fourier coefficients. Then considering a single frequency case, we fix a frequency $\omega \in \{0, 1, \ldots, N-1\}$.

The frequency position can be represented in binary format and its length is $log_2(N)$. We employ the bit-testing algorithm to find the frequencies bit by bit. To estimate the bit value of the frequency, we apply the frequency mask filters [3]:

$$g_b^{even}[n] = \frac{1}{2}(\delta[n] + \delta[n - N/2^{b+1}]), \tag{3.2}$$

$$g_b^{odd}[n] = \frac{1}{2}(\delta[n] - \delta[n - N/2^{b+1}]), \tag{3.3}$$

where $n = 0, 1, \ldots, N-1$ and $\delta[n]$ is the unit impulse

$$\delta[n] = \begin{cases} 0 & \text{n}\neq 0 \\ 1 & \text{n}=0 \end{cases} \tag{3.4}$$

and b represents the digit of the binary number. The filter g_b^{even} only allows the even frequencies to pass. Similarly, the filter g_b^{odd} only allows the odd frequencies to pass. Since the representation of the frequency position is binary, it can only be 0 or 1 for each bit. Assuming the magnitude of the output of filter g_b^{odd} and filter g_b^{even} are H_1 and H_0 respectively, then our decision rule can be expressed as

$$|H_1| \underset{\varepsilon=0}{\overset{\varepsilon=1}{\gtrless}} |H_0|, \tag{3.5}$$

where the ε assigns the bit value. Hence, if the output of the filter g_b^{odd} is larger than g_b^{even}, we can decide that this digit is 1 (i.e., $\varepsilon = 1$), and vice versa. Next, we demodulate the signal with

$e^{-2\pi i 2^b \varepsilon t/N}$

$$x[t] = c_\omega e^{2\pi i \omega t/N} \times e^{-2\pi i 2^b \varepsilon t/N} \tag{3.6}$$

$$= c_\omega e^{2\pi i(\omega - 2^b \varepsilon)t/N}. \tag{3.7}$$

The frequency of the demodulated signal is $\omega - 2^b\varepsilon$. Then input the demodulated signal to the frequency mask filters to do the next bit-test. For example, the frequency is 13 Hz, can be written as 1101 in binary. When $b = 0$, we plug $b = 0$ into Equation (3.2) and (3.3). Then we have

$$g_b^{even}[n] = \frac{1}{2}(\delta[n] + \delta[n - N/2]) \xleftrightarrow{\mathscr{F}} G_b^{even}(k) = \begin{cases} 1 & \text{k=0,2,4,...,N-2,} \\ 0 & \text{k=1,3,5,...,N-1,} \end{cases} \tag{3.8}$$

$$g_b^{odd}[n] = \frac{1}{2}(\delta[n] - \delta[n - N/2]) \xleftrightarrow{\mathscr{F}} G_b^{odd}(k) = \begin{cases} 0 & \text{k=0,2,4,...,N-2,} \\ 1 & \text{k=1,3,5,...,N-1,} \end{cases} \tag{3.9}$$

where G_b^{even} and G_b^{odd} are the Fourier transform of the g_b^{even} and g_b^{odd}, respectively. k denotes the frequency domain ordinal. In the frequency domain, we can easily find that G_b^{even} only passes the even frequency and G_b^{odd} only passes the odd frequency. By convolving the input data with the filters, we have $|H_1| > |H_0|$ (i.e., $\varepsilon = 1$). Because 13 is a odd number. Then demodulate the signal with $e^{-2\pi i 2^0 \varepsilon t/N}$. Applying next bit-testing to the demodulated signal, we have the $|H_1| < |H_0|$ (i.e., $\varepsilon = 0$) when $b = 1$. Similarly, we have $|H_1| > |H_0|$ (i.e., $\varepsilon = 1$) when $b = 2$ and $b = 3$. So the frequency value is equal to $2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 13$.

Note that the estimated results of bit-testing probably also contain the frequencies of noise and interference. So, later, we will talk about how to decide the true frequencies in Section 3.3 and Section 3.4.

3.2 Randomly Binning Frequencies

3.2.1 Random Sampling and Random Permute

If the input data have multiple frequency components, we need to filter the data to isolate the frequencies. Before filtering, we make the frequencies well separated by random sampling and random permuting the data. The easiest way to do random sampling and random permuting is to utilize the basic properties of Fourier transform (see Equation (2.5) and (2.6)) and create

$$x[(\sigma(k-1)+t) \mod N], \tag{3.10}$$

where $\sigma \in \{1, 3, 5, \ldots, N-1\}$, $\tau \in \{0, 1, 2, \ldots, N-1\}$. k represents the sample ordinal and $k = 1, 2, \ldots, K$, where K is the size of the sample set. Here, σ is used to make the frequency components well separated. t is used to decide the sample locations randomly.

3.2.2 Filtering to Isolated Frequencies

After random sampling and random permuting, we apply a sub-band decomposition filter bank (e.g., boxcar filter: $h[t] = \frac{\sqrt{N}}{K}, \quad t = 0, 1, 2, \ldots, K-1$) to create K signals that each carries a chunk of the permuted spectrum. The K is a constant and represents the number of taps of the filter bank. It is recommended in [3] to take K equal to 7 times the number of the frequency components. Each component only contains one signifucant frequency with high probability. Then each isolated frequency can be identified by the approach described in Section 3.1.

3.3 Estimate Coefficients

The simplest approach to estimate the coefficients is [1]

$$\hat{c}_\omega = \frac{1}{K} \sum_{k=1}^{K} x[t_k] e^{-2\pi i \omega t_k / N} \tag{3.11}$$

where K is the size of the random sample set, with $K \ll N$. $x[t_k]$ is the random sample which is defined in Section 3.2.1, and the ω is the frequencies which are identified by bit-testing. Note

that only K samples are used to estimate the coefficients $\hat{c}_\omega$, instead of using all N samples. Then we compare all the $\hat{c}_\omega$ and retain the m coefficients with the largest magnitude.

3.4 Compute the Residual and Repeat

Assume we have the ω and $\hat{c}_\omega$, then we compute the residual by

$$x[t] - \sum_{all\ \omega\ and\ \hat{c}_\omega} \hat{c}_\omega e^{2\pi i \omega t/N} \tag{3.12}$$

The residual will be used in subsequent repetition. One primary purpose of repetition is to avoid mistakenly identifying insignificant frequencies as being significant [1]. The spurious Fourier coefficients can be removed in subsequent repetition. Hence, the repetition process can correct the error estimation efficiently. In the end of each repetition loop, we should compare the old m coefficients with the new $\hat{c}_\omega$ and find the new m largest coefficients.

Chapter 4

SFT-based DOA Detection on Sensor Array

As we said before, we hope that the Sparse Fourier Transform can save physical antennas, and achieve the same target estimation performance as the DFT. First, we tested the SFT on a phased array and found that the resolution was not good enough with a small number of antennas. So we tried to employ the coprime array to obtain higher resolution.

4.1 Phased Array Based on SFT

4.1.1 Phased Array model

As shown in Figure 4.1, N is the number of sensors in a phased array. d represents the distance between two antennas, and is here taken as half of signal wavelength $\lambda/2$. θ is the direction of arrival (DOA). The target is in the far field of the phased array. First, we suppose there is one target. The case of multiple targets can be easily obtained as an extension. The main assumption here is that the number of targets is much smaller than N.

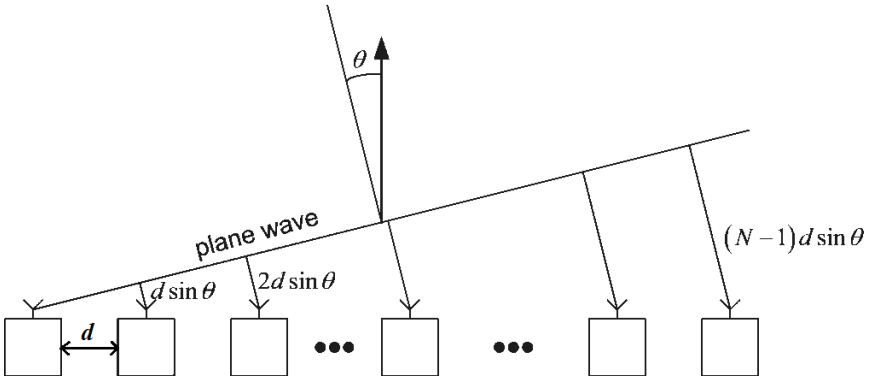

Figure 4.1: Model for phased array

The received signal of the qth sensor can be expressed as

$$y_q(t) = m(t)e^{2\pi i \frac{q \cdot d \sin(\theta)}{\lambda}}, \quad q = 0, 1, \ldots, N-1, \tag{4.1}$$

where $m(t)$ represents a narrow-band signal and λ is the signal wavelength. t denotes time. The array snapshot at time t is

$$\mathbf{y}(t) = [y_0(t), y_1(t), \ldots, y_{N-1}(t)]^{\mathbf{T}} \tag{4.2}$$

$$= m(t)\mathbf{a}(\theta), \tag{4.3}$$

where

$$\mathbf{a}(\theta) = [e^{2\pi i \frac{0 \cdot d \sin(\theta)}{\lambda}} \; e^{2\pi i \frac{1 \cdot d \sin(\theta)}{\lambda}} \; \ldots \; e^{2\pi i \frac{(N-1) \cdot d \sin(\theta)}{\lambda}}]^{\mathbf{T}} \tag{4.4}$$

is the steering vector of the array corresponding to DOA θ.

4.1.2 Applying SFT to DOA Detection

Let us compare Equations (3.1) and (4.1), and draw the following equivalences:

$$(4.1) \longleftrightarrow (3.1) \tag{4.5}$$

$$s(t) \longleftrightarrow c_\omega \tag{4.6}$$

$$l \longleftrightarrow t \tag{4.7}$$

$$(\text{spatial frequency})\frac{d\sin(\theta)}{\lambda} \longleftrightarrow \frac{\omega}{N} \tag{4.8}$$

By taking the SFT of the array snapshot, we can identify the $\frac{d\sin(\theta)}{\lambda}$ corresponding to the target, thus obtaining DOA information. For the traditional DFT, all the snapshot samples should be used to compute the DFT coefficients. For the SFT, only a subset of samples need to be used and only the significant coefficients will be computed. Hence, we only need to use a subset of the antennas to detect the DOA by implementing SFT.

4.1.3 Performance of the SFT-based DOA Detection on Phased Array

Assume that we employ 128 sensors in a 3 target senario. Then the SFT output is shown in Figure 4.2 and the detection result is shown in Figure 4.3.

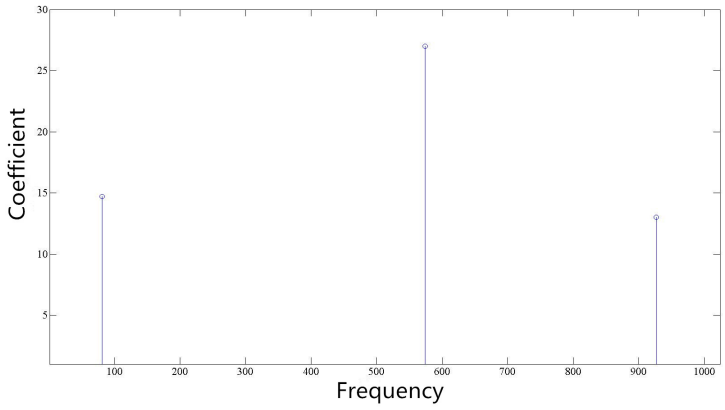

Figure 4.2: The SFT output when we have 3 targets

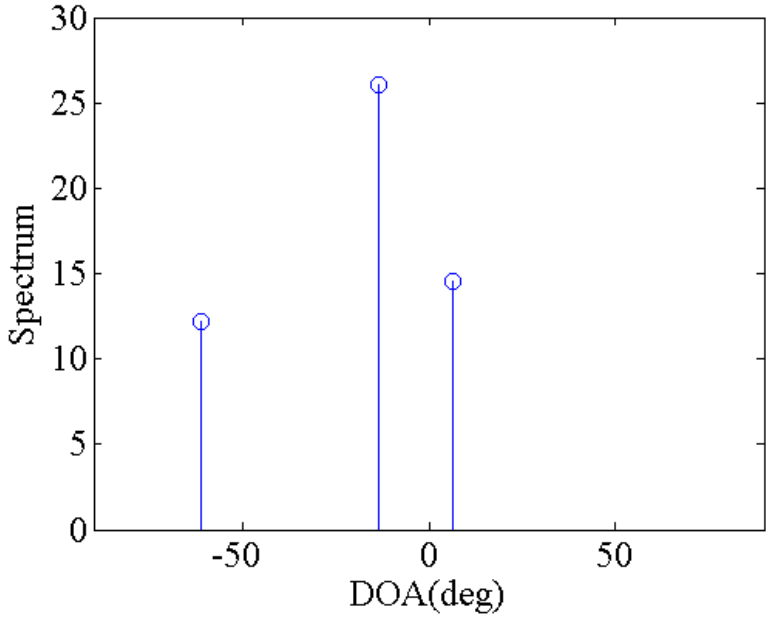

Figure 4.3: The result of DOA detection when we have 3 targets

Figure 4.2 shows the output of the SFT. By drawing the equivalences, we can obtain the Figure 4.3, which shows the result of SFT-based DOA detection. The DOAs are identified. In this result, we only use 83 antennas to detect the 3 targets. Hence, we can turn off the other unused sensors to save power.

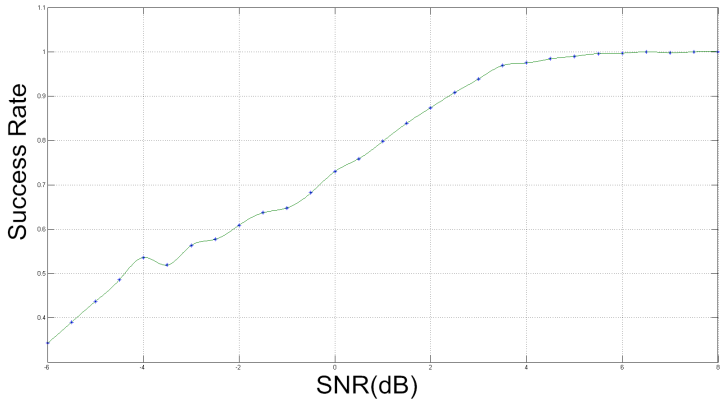

Figure 4.4: The success rate in different SNR

As the Figure 4.4 shows, the success rate is almost 100% if the SNR is larger than 5 dB. Each SNR was tested for 1000 times. The SFT-based DOA detection can keep high success rate when the SNR is larger than 5 dB, even if only part of the sensors is used. The success rate is shown in Figure 4.4. However, the resolution of $\sin(\theta)$ is proportion of $\frac{\lambda}{Nd}$ with λ and d being constant. Hence, if we want to increase the resolution, we have to use more sensors. However, the cost would increase in that case.

4.2 Coprime Array Based on SFT

Since we want higher resolution with using few sensors, we employ the coprime array, introduced in [5], to greatly extend the degrees of freedom (DOFs).

4.2.1 The Co-prime Array Model

Figure 4.5 shows that the co-prime array contains two uniform linear arrays (ULAs) with N and $2M$ sensors respectively. d is half of the wavelength of the signal. M and N are coprime integers. The two ULAs share the first sensor.

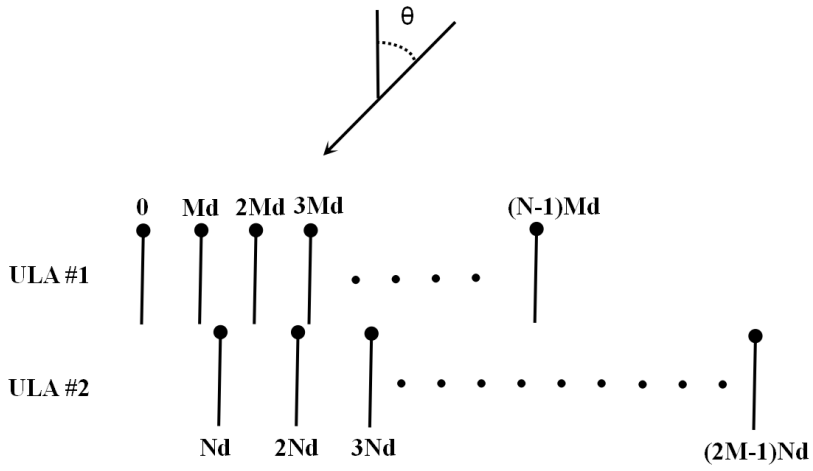

Figure 4.5: Model for co-prime arrays

That is, the locations of the sensors are given by the set [5]

$$S = S_{ULA1} \cup S_{ULA2} \tag{4.9}$$

$$S_{ULA1} = \{Mnd, 0 \leq n \leq N-1\} \tag{4.10}$$

$$S_{ULA2} = \{Nmd, 1 \leq m \leq 2M-1\} \tag{4.11}$$

The steering vector of the sensor located at q corresponding to one particular DOA, θ, is:

$$a(q,\theta) = e^{j\frac{2\pi}{\lambda}q\sin(\theta)},\ q \in S,\ \theta \in [-\frac{\pi}{2}, \frac{\pi}{2}). \tag{4.12}$$

Then the steering vector of the array corresponding to one particular DOA is

$$\mathbf{a}(\theta) = [a(0Md,\theta), \ldots, a((N-1)Md,\theta), a(1Nd,\theta), \ldots, a((2M-1)Nd,\theta)]^{\mathbf{T}} \tag{4.13}$$

$$= [e^{2\pi i\frac{0Md\sin(\theta)}{\lambda}}, \ldots, e^{2\pi i\frac{(N-1)Md\sin(\theta)}{\lambda}}, e^{2\pi i\frac{Nd\sin(\theta)}{\lambda}}, \ldots, e^{2\pi i\frac{(2M-1)Nd\sin(\theta)}{\lambda}}]^{\mathbf{T}} \tag{4.14}$$

Next, assuming we have L targets, the received signal can be written as

$$\mathbf{y}[t] = \mathbf{V}\mathbf{s}[t], \tag{4.15}$$

where $\mathbf{V} = [\mathbf{a}(\theta_1), \mathbf{a}(\theta_2), \ldots, \mathbf{a}(\theta_L)]$ represents the $(N+2M-1) \times L$ steering matrix , t denotes the tth time, and $\mathbf{s}[t] = [s_1[t], s_2[t], \ldots, s_L[t]]^{\mathbf{T}}$ represents the $L \times 1$ narrow-band signal vector. Hence, $\mathbf{y}[t]$ is a $(N+2M-1) \times 1$ vector.

4.2.2 The Mathematical Theory of Co-prime Arrays

The reason why we choose to use N and $2M$ sensors respectively can be found as follows [6]:

Theorem 1. *Assume M and N are coprime numbers with $M < N$. Given an integer k in the range $0 \leq k \leq MN$, there exist integers n and m in the ranges $0 \leq n \leq N-1$ and $0 \leq m \leq 2M-1$ such that $k = Nm - Mn$. The corresponding k is produced by the same choice of m and n by considering the negative of this difference.*

Proof.

$\because k = Nm - Mn$

$\therefore Nm = k + Mn$

$\because 0 \leq n \leq N-1 \;\; and \;\; 0 \leq k \leq MN$

$\therefore 0 \leq Nm \leq 2MN - M$

$\therefore 0 \leq m \leq 2M - \frac{M}{N}$

$\because M < N, \;\; i.e., \frac{M}{N} < 1 \;\; and \; m \; is \; integer$

$\therefore 0 \leq m \leq 2M - 1$ □

Hence, Theorem 1 ensures that all integers from 1 to MN are generated consecutively as follows:

$$\{1, 2, \ldots, MN\} \in \{Nn - Mm,\ 0 \leq n \leq N-1,\ 0 \leq m \leq 2M-1\}. \tag{4.16}$$

For example, $m = 3$ and $n = 4$. Then we can find $\{-12, -11, -10, \ldots, -2, -1,$

$1, 2, \ldots, 10, 11, 12\} \in \{\pm(4n - 3m),\ 0 \leq n \leq 3,\ 0 \leq m \leq 5\}$.

Table 4.1: The value of $k = \pm(Nm - Mn)$ when $M = 3$ and $N = 4$

m	n	$k = Mn - Nm$	m	n	$k = Nm - Mn$
3	0	-12	1	1	1
5	3	-11	2	2	2
4	2	-10	3	3	3
3	1	-9	1	0	4
2	0	-8	2	1	5
4	3	-7	3	2	6
3	2	-6	4	3	7
2	1	-5	2	0	8
1	0	-4	3	1	9
3	3	-3	4	2	10
2	2	-2	5	3	11
1	1	-1	3	0	12

4.2.3 DOA Detection Using Co-prime Array

To increase the DOFs, we first compute the autocorrelation matrix [5]:

$$\mathbf{R}_{yy} = \mathbb{E}[\mathbf{y}[t]\mathbf{y}^H[t]]. \tag{4.17}$$

Then vectorizing the $\mathbf{R}_{yy}$, we have

$$\mathbf{z} = vec(\mathbf{R}_{yy}). \tag{4.18}$$

The phase part of elements at $(N+m, n)$ and $(n, N+m)$ of $\mathbf{R}_{yy}$ are $e^{-j(Mn-Nm)d\frac{2\pi}{\lambda}\sin(\theta)}$ and $e^{-j(Nm-Mn)d\frac{2\pi}{\lambda}\sin(\theta)}$, respectively, where $n = 0, 1, \ldots, N-1$ and $m = 1, 2, \ldots, 2M-1$. $\pm(Mn - Nm)$ belongs to $\{-MN, -(MN-1), \ldots, (MN-1), MN\}$ due to Theorem 1. Hence, we can build the $\mathbf{z}$ by including the elements at $(N+m, n)$ and $(n, N+m)$ of $\mathbf{R}_{yy}$ sequentially. Then sorting the rows of $\mathbf{z}$ from $-MNd$ to MNd, we can obtain

$$\mathbf{z} = \mathbf{B} \cdot \mathbf{p}, \tag{4.19}$$

where

$$\mathbf{B} = \begin{bmatrix} e^{-jMNd\frac{2\pi}{\lambda}\sin(\theta_1)} & e^{-jMNd\frac{2\pi}{\lambda}\sin(\theta_2)} & \ldots & e^{-jMNd\frac{2\pi}{\lambda}\sin(\theta_L)} \\ e^{-j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_1)} & e^{-j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_2)} & \ldots & e^{-j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_L)} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_1)} & e^{j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_2)} & \ldots & e^{j(MN-1)d\frac{2\pi}{\lambda}\sin(\theta_L)} \\ e^{jMNd\frac{2\pi}{\lambda}\sin(\theta_1)} & e^{jMNd\frac{2\pi}{\lambda}\sin(\theta_2)} & \ldots & e^{jMNd\frac{2\pi}{\lambda}\sin(\theta_L)} \end{bmatrix}, \tag{4.20}$$

$$\mathbf{p} = [s_1^2[t]\ s_2^2[t]\ \ldots\ s_L^2[t]]^T. \tag{4.21}$$

$\mathbf{z}$ is called the virtual array output. $\mathbf{B}$ can be considered as the steering matrix with $2MN+1$ virtual sensors located at qd, $-MN \leq q \leq MN$ and can implement the DOA estimation with this virtual array which evidently gives $2MN+1$ DOFs as opposed to the physical array, which provides only $N + 2M - 1$ DOFs [5]. We can apply the SFT-based DOA detection on $\mathbf{z}$ as the process discribed in the Section 4.1.2, and hence the resolution increases.

We measure the success rate when $n = 9$ and $m = 7$ without noise and let the quantity of targets be equal to 5.

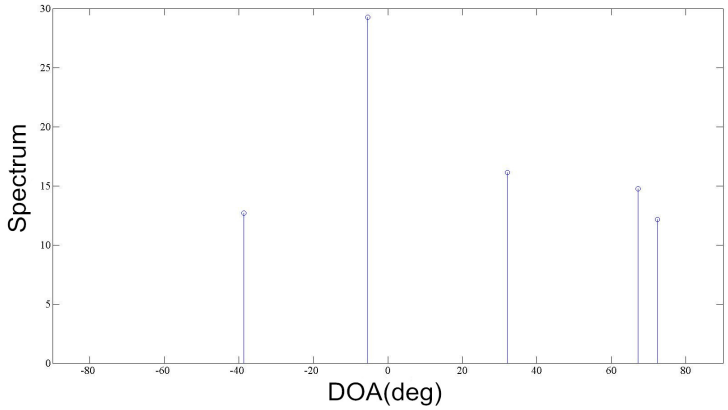

Figure 4.6: The result of DOA detection when we have 5 targets

In Figure 4.6, it shows that the DOAs are identified. In no noise circumstances, we can find that the detected DOAs match to the angles of the targets. Then we run this test for 400 times with noise with $SNR = -4dB$ to $SNR = 10dB$ and compute its accuracy.

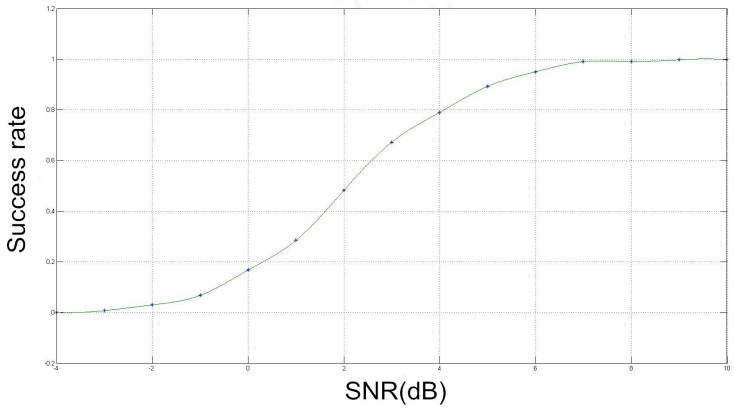

Figure 4.7: The Succedd Rate of the SFT-based DOA Detection Using Coprime Array

In Figure 4.7, it shows that the success rate is close to 100% when the SNR is larger than 7. Then we try to use $9 + 2 \times 7 - 1 = 22$ sensors to detect $2 \times 9 \times 7 + 1 = 127$ targets in order to confirm that the coprime array can give $2MN + 1$ degrees of freedom when the physical array

provides only $N + 2M - 1$ degrees of freedom as shown in Figure 4.8. The detected DOAs match to the angles of the targets. Hence, the resolution greatly increases, because the DOFs increase.

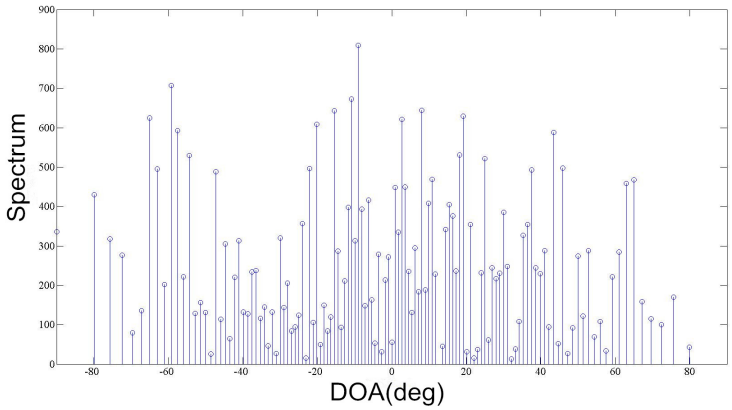

Figure 4.8: The result of DOA detection when we have 127 targets

Since we assume there is a small number of targets, we focus on the 5 targets case which is shown in Figure 4.6. In that case, we only use a subset of virtual sensors. But we have to use all physical sensors. In next chapter, we will introduce how to save physical sensors in coprime arrays.

Chapter 5

DOA Detection Based on Fixed SFT

Since the sample locations are random as we discribed in Equation (3.10), the spared sensors are different in each implementation. This chapter will introduce how to execute the SFT based on a fixed set of spared sensors.

5.1 Pesudorandom Sampling

Due to the advantage of the Sparse Fourier Transform, we are able to only utilize a subset of the elements in virtual array, which is derived from the coprime array, to estimate the DOA. As we discribed in Equation (3.10), the traditional SFT algorithm can identify the frequency (e.g. DOA in radar system) and estimate its Fourier coefficients using random sampling, i.e., using

$$x[(\sigma(k-1)+t) \mod N], \tag{5.1}$$

where $\sigma \in \{1, 3, 5, \ldots, N-1\}$ and $t \in \{0, 1, 2, \ldots, N-1\}$. The sample set does not depend on the input data or the progress of the algorithm. That is, the sample locations can be decided before implementation [3]. Hence, the σ and t can be generated pesudorandomly before execution to make the virtual sample locations fixed.

5.2 Saving Sensors

We can build the sample set by picking the elements pseudorandomly and apply it to detect any signals. The scheme to build the sample set is pseudorandomly picking the virtual samples, which are generated from just part of physical sensors. The virtual samples located at

$\Psi = \{\pm(Ni - Mn)d, 0 \leq n \leq N-1\}$ are generated by the i^{th} physical sensor in the second ULA. Hence, if we want to save the i^{th} physical sensor, we should not use the virtual samples located at Ψ. The steps for the SFT-based DOA estimation using the pseudorandom sample set are:

1) Determine the i^th sensor of the second ULA to spare.

2) Randomly generate the virtual sample set.

3) If none of the samples of the virtual sample set are located in $\Psi = \{\pm(Ni - Mn)d, 0 \leq n \leq N-1\}$, keep the sample set. Otherwise, regenerate it.

4) Apply the Sparse Fourier Transform algorithm to estimate the frequencies "ω".

5) Make the analogies to obtain the estimated DOAs θ.

Similarly, we can save a bunch of physical sensors. The number of spared sensers is affected by the number of source signals and the SNR. We will show this via simulation in the next section. In addition, in Table 4.1, two values of parameter k will be generated if $m = 5$. That is, the last physical sensor always generates the fewest virtual sensors which are located from $-MNd$ to MNd. Hence, we recommend to save the physical sensors sequentially from the end of the array.

For example, we can only use 23+38 sensors of the 23+43 sensors to detect 5 DOAs based on SFT in a coprime array. In other words, we try to spare the last 5 sensors. Then we can adjust the parameters, σ and t, and find a appropriate sample set $\Omega = \{x(\xi) | \xi = (\sigma(k-1) + t) \mod N, \sigma \in \{1, 3, 5, \ldots, N-1\}, t \in \{0, 1, 2, \ldots, N-1\}\}$. Then we should decide if any sample of the sample set is in $\Psi = \{\pm(Ni - Mn)d, 0 \leq n \leq N-1, i = 39, 40, 41, 42, 43\}$. If it is not in Ψ, we can keep the sample set. Otherwise, we should regenerate it. Next, we can do SFT-based DOA detection by using this sample set Ω. Its performance is shown in Figure 5.1. The algorithm works well when $SNR \geqslant 0\ dB$.

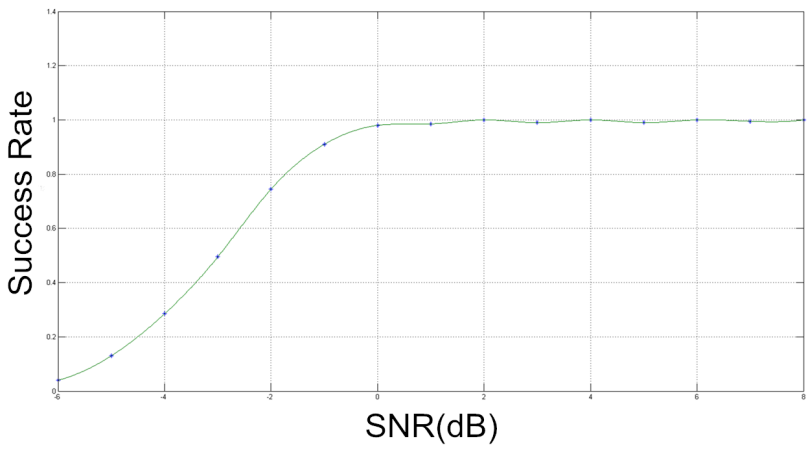

Figure 5.1: The success rate in different SNR using 23+38 sensors.

5.3 Simulation Results

In this part, we provide three examples of our proposed DOA estimation method. The first one is to show the ability of SFT-based DOA estimation and the second one is to show the sensor savings in co-prime arrays. The third example shows the relationship between the percentage of sensor savings and the number of physical sensors. The SFT-based DOA estimation needs the number of targets as prior knowledge. We applied the Akaike information criterion (AIC) to estimate the number of targets before implementing the proposed DOA estimation [7].

Example 1: In this example, we consider the co-prime array consisting of 11 physical sensors. It is composed of two ULAs with $N = 6$ and $M = 5$. That is, we use 6+9 sensors. The sensor locations of the two ULAs are respectively $[0, 5, 10, 15, 20]d$ and $[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]d$. The first sensor is shared by both ULAs and the inter-sensor spacing d is equal to the half wavelength. We assume there are $m = 25$ narrowband source signals with DOAs distributed in $[-\frac{\pi}{2}, \frac{\pi}{2})$. The autocorrelation matrix $\mathbf{R}_{yy}$ is estimated using 500 snapshots. The SNR is chosen as 15 dB. Figure 5.2 shows the spatial spectrum obtained from the SFT-based DOA estimation method. As we can see, all DOAs are identified.

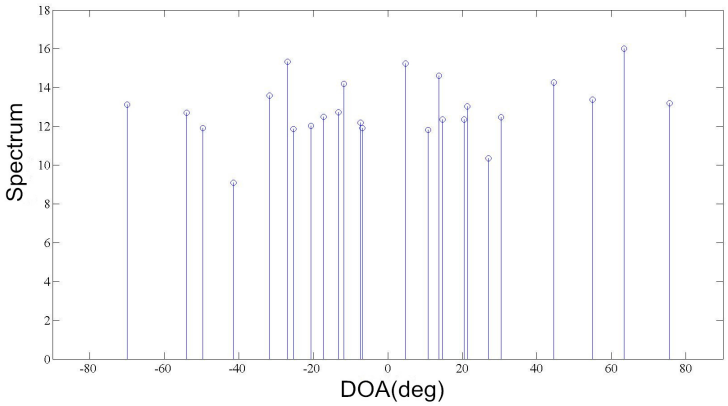

Figure 5.2: Spatial spectrum for SFT-based DOA estimation on co-prime array

This example shows the ability of our proposed method to estimate the DOAs of many more targets than the number of the physical sensors, which is not possible using traditional ULA with $N+2M-1$ physical sensors. As we said before, the co-prime array gives evidently $2MN+1$ DOFs with $N+2M-1$ physical sensors. Hence, it has a ability to achieve very high resolution of DOA estimation.

Example 2: Here we consider the co-prime array consisting of 66 physical sensors and $N = 23$ and $M = 22$. Then we suppose there are $m = 5$ narrowband source signals with DOAs distributed in $[-\frac{\pi}{2}, \frac{\pi}{2})$. Our proposed DOA estimation method does not need to use all the samples from all the sensors, we try to reduce the number of sensors and observe what happens.

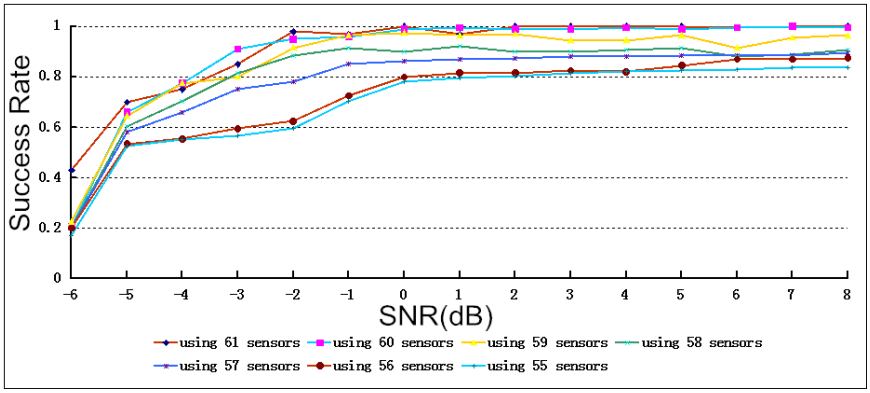

Figure 5.3: The success rate for sensor-saving co-prime array in different SNR.

Figure 5.3 shows that the success rate of DOA estimation decreases when the SNR goes down. The success rate does not drop sharply when we reduce the number of physical sensors. Since we have to provide enough samples to implement the SFT process, we can save 11 sensors at most. In addition, the more DOAs we want to estimate, the more samples we need in order to keep the success rate good enough as shown in Figure 5.4. In the figure, the percentage of saving sensors is obtained when the SNR is -5 dB and success rate is approximately 90%. So we might save less sensors when the number of targets increases.

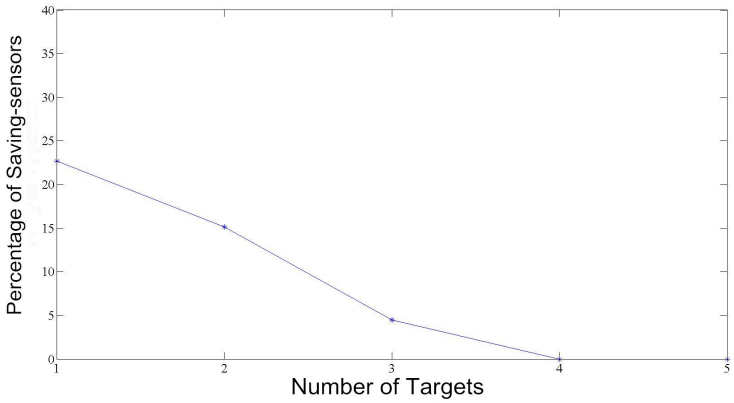

Figure 5.4: The percentage of saving sensors in different number of targets.

Example 3: Now we want to estimate the DOA of 2 source signals and achieve approximately 90% success rate. The SNR is chosen as -5 dB. The autocorrelation matrix is estimated using 500 snapshots. The angle of source signals is distributed in $[-\frac{\pi}{2}, \frac{\pi}{2})$. Here we try to use the co-prime arrays with different number of virtual sensors to estimate the DOAs. The relationship between percentage of physical sensors we can save and the number of virtual sensors is shown in Figure 5.5.

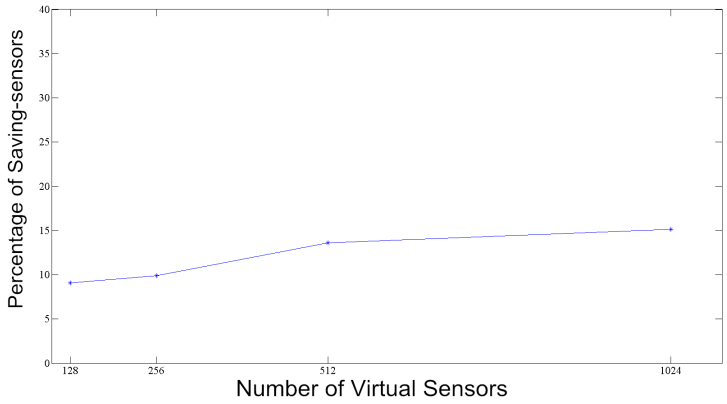

Figure 5.5: The percentage of saving sensors using different number of virtual sensors.

Chapter 6

Conclusion

In this thesis, we studied a new algorithm to compute the DFT, known as Sparse Fourse Transform. The SFT applies to signals which are sparse in the frequency domain, and has the ability to compute the DFT of a signal with only using a subset of the signal samples. The sample set does not depend on the input data or the progress of the algorithm [3]. So the sample set can be fixed before the implementation.

We appiled the SFT on co-prime arrays for fast and high resolution DOA estimation using fewer physical snesor. The high performance of the proposed DOA estimation method was verified using simulation results.

Bibliography

[1] A. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse fourier transform: A compressed fourier transform for big data," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 91–100, 2014.

[2] A. C. Gilbert, S. Muthukrishnan, and M. Strauss, "Improved time bounds for near-optimal sparse fourier representations," in *Optics & Photonics 2005*, 2005, pp. 59 141A–59 141A.

[3] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," *IEEE Signal processing magazine*, vol. 25, no. 2, pp. 57–66, 2008.

[4] P. P. Vaidyanathan and P. Pal, "Sparse sensing with co-prime samplers and arrays," *IEEE Transactions on Signal Processing*, vol. 59, no. 2, pp. 573–586, 2011.

[5] P. Pal and P. P. Vaidyanathan, "Coprime sampling and the music algorithm," in *2011 IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE)*, 2011, pp. 289–294.

[6] P. Vaidyanathan and P. Pal, "Sparse sensing with coprime arrays," in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. IEEE, 2010, pp. 1405–1409.

[7] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 2, pp. 387–392, 1985.