APPLICATIONS OF SPARSE SIGNAL RECOVERY: 2D-PATTERN MATCHING AND SPARSE WALSH-HADAMARD TRANSFORM COMPUTATION

A Thesis

by

JIAHUI GAO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Krishna Narayanan
Committee Members,	Anxiao Jiang
	Jean-Francois Chamberland
	Nick Duffield
Head of Department,	Miroslav Begovic

December 2020

Major Subject: Electrical Engineering

ABSTRACT

We study two problems related to sparse signal recovery. The first problem considered is querying a sub-image of size $M \times M$ in a large image database of size $N \times N$ to determine all the locations where sub-image appears. We use sparse graph based codes Fourier transform computation to compute the peaks in the 2-D correlation to determine the matching positions in a computationally efficient manner. We then design a 2-D pattern that can facilitate vision based positioning by enabling the use of our algorithm for fast pattern matching. The second problem studied is the computation of sparse Walsh-Hadamard transform for binary data. We consider signals that are sparse in Walsh-Hadamard tranform domain where the non-zero coefficients are all ones. A possible application of this algorithm is learning an undirected unweighted graph by using a sub-sample version of its evaluation. We design an adaptive algorithm for sparse WHT computation. Adaptivity provides an opportunity to recover more than one non-zero coefficient aliased together in each iteration so that a faster recovery can be expected given the same amount of sub-samples. It is shown that with the same amount sample, the probability of error of our proposed algorithm is lower compared to the earlier work.

DEDICATION

To my family.

ACKNOWLEDGMENTS

I would like to thank both of my parents, Ming Gao and Zhaoxia Guo, for their support and love. I would like to thank my professor, Dr. Krishna Narayanan for his guidance and encouragement. I would also like to thank Dr. Anxiao Jiang, Dr. Jean-Francois Chamberland, and Dr. Nick Duffield. for there willingness to be on my committee.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Krishna Narayanan [advisor] and Professor Jean-Francois Chamberland and Professor Nick Duffield of the Department of Electrical and Computer Engineering [Home Department] and Professor Anxiao Jiang of the Department of Computer Science and Engineering [Outside Department].

All the work conducted for the thesis was completed by the student independently under the supervision of Professor Krishna Narayanan.

Funding Sources

No outside funding was received in the research and writing of this document.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES ix

1. INTRODUCTION 1

    1.1 Motivation 1
    1.2 Applications of Sparse Signal Recovery 1
    1.3 Thesis Organization 2

2. 2-D PATTERN MATCHING USING SPARSE FOURIER TRANSFORM APPROACH* . 3

    2.1 Retated Work 3
        2.1.1 2-D FFAST 3
        2.1.2 Sub-string/Pattern Matching in Sub-linear Time Using a Sparse Fourier Transform Approach 6
        2.1.3 Proposed Work 9
    2.2 2D-Pattern Matching 9
        2.2.1 System Model and Notations 9
        2.2.2 2-D Pattern Matching Algorithm 10
            2.2.2.1 Bin Classification and Decoding 13
            2.2.2.2 Peeling Process 15
            2.2.2.3 Sketch of y' 15
        2.2.3 Sample and Computation Complexity 15
            2.2.3.1 Sample Complexity 15
            2.2.3.2 Computation Complexity 16
        2.2.4 Design of 2D Patterns for Matching 17
        2.2.5 Simulation Results 18
            2.2.5.1 Performance for Binary Randomly Generated Data 18
            2.2.5.2 Performance with designed patterns 21

3. IMPROVED SPARSE WALSH-HADAMARD TRANSFORM FOR BINARY DATA 25

3.1 Introduction 25
3.1.1 Walsh-Hadamard Transform 25
3.1.2 Graph learning problems and cut function 26
3.2 Retated Work 27
3.2.1 Fast Hadamard Transform with Sublinear Sparsity in the Transform Domain 27
3.2.2 Improved Sparse Walsh-Hadamard Transform 28
3.3 Improved Sparse Walsh-Hadamard Transform for binary data 30
3.3.1 Choice of shift 32
3.3.2 Bin Classification 33
3.3.3 Position Identification by Look up table 33
3.3.4 Peeling Decoder 34
3.3.5 Sample and Computation Complexity 35
3.3.5.1 Sample Complexity 35
3.3.5.2 Computation Complexity 35
3.3.6 Simulation 35
3.3.6.1 Random generated binary data 35
3.3.6.2 Random generated graph data 36

4. CONCLUSIONS 38

4.1 Further work 38
4.1.1 2-D Pattern Matching 38
4.1.2 Improved sparse Walsh-Hadamard transform computation for binary data ... 39
4.1.2.1 Graph Learning 39
4.1.2.2 Adaptivity of the proposed algorithm 39

REFERENCES 40

LIST OF FIGURES

FIGURE Page

2.1 2D-FFAST Architecture with d = 2. This figure is from [1] with permission. 4

2.2 Bi-partite Graph of the non-zero coefficient and the bin observation. This figure is from [1] with permission. 5

2.3 Flow chart of the algorithm from [2] 7

2.4 Sparse signal recovery algorithm from [2]. 7

2.5 Bi-partite graph for RSIDFT. 9

2.6 Flowchart of the proposed algorithm. 11

2.7 Architecture of the pattern matching algorithm. 13

2.8 2 patterns with immunity to scaling and rotation. 17

2.9 Flowchart of the 2-D pattern matching algorithm. 18

2.10 Sub-image and database 19

2.11 Cross-correlation signal. 20

2.12 Performance with different sparsity. 21

2.13 Pattern with real image. 22

2.14 Correlation of the patterns. 23

2.15 Performance when image is corrupted with Gaussian noise. 24

3.1 Schematic of improved sparse Walsh-Hadamard algorithm 29

3.2 Performance of the proposed algorithms on random generated binary data. 36

3.3 Performance of the proposed algorithms on graph data. 37

LIST OF TABLES

TABLE Page

2.1 Parameters involved in describing the algorithm 10

3.1 Parameters involved in describing the algorithm. 31

1. INTRODUCTION

1.1 Motivation

The discrete Fourier Transform (DFT) and the Walsh-Hadamard transform (WHT) are powerful tools for analyzing signals that arise in a variety of applications in engineering and sciences. Algorithms for efficiently computing the DFT or WHT have been studied for several years. The Fast Fourier Transform (FFT) is the most well-known algorithm to compute the DFT with the computation complexity of $O(N \log N)$ given data of length N. Similarly, algorithms that compute the WHT with a complexity of $O(N \log N \log \log N)$ exist.

Even when the amount of data is huge, many signals that are commonly encountered in engineering and science are often sparse in the discrete transform domain, which means many of the coefficients in that domain are zeros or they are so small that can be treated like zeros. For example, discrete cosine transform (DCT) of images usually contains many zeros or very small coefficients, audio signals are usually sparse in frequency domain. When the sparsity assumption is valid, algorithms can be designed to compute the discrete transform of the signal of large size with lower sample and computational complexity (particularly, in sub-linear time). Recently, there has been significant progress in the design and analysis of sub-linear time algorithms for efficient computation of sparse discrete Fourier transform (DFT) and sparse Walsh-Hadamard transform (WHT) based on graph-based codes [3, 4, 5].

1.2 Applications of Sparse Signal Recovery

Sparse transform computation has found applications in several areas of engineering, mathematics and computer science [6]. In this thesis, we are interested in the following two problems. The first one is pattern matching. Pattern matching is an important problem that has potential applications in bioinformatics, global positioning system (GPS), data querying and so on. Very recently sparse DFT computation algorithms have been used for 1-D pattern matching in [2, 7]. In this thesis, we first expand the scope of the results in [2] to 2-D pattern matching algorithm based

on 2-D sparse DFT and discuss its advantages for finding image patterns in image data sets.

The other application considered is sparse Walsh-Hadamard transform computation for signals whose WHTs are sparse and the non-zero coefficients are ones. This is motivated by applications in learning sparse set functions of graphs such as what is considered in in [8]. Here we extend the results of [5] to design an adaptive algorithm to compute the Walsh-Hadamard transform of signals whose WHT is sparse and the non-zero coefficients are ones.

1.3 Thesis Organization

The rest of the thesis is organized as follow. In the chapter two, we first introduce two works that are related to our proposed 2-D pattern matching algorithm. Our work is introduced in the following section. Simulation results are also included in that section with a demonstration of the performance both for binary data and real image implementation. In chapter three, we discuss the sparse Walsh-Hadamard transform computation with the application in graph learning. The introduction of Walsh-Hadamard transform and the graph learning problem is given in the first section. The following section describes the scheme of our proposed adaptive algorithm. The simulation is given in the same section to show the performance or improvement of the proposed work on binary data. Finally, we give a conclusion and proposed some potential future works and thoughts on related topics.

2. 2-D PATTERN MATCHING USING SPARSE FOURIER TRANSFORM APPROACH*

In this chapter, we present our work on 2-D pattern matching by using a sparse signal recovery strategy. Pattern matching is a fundamental data science problem aiming to find the location of a small scale data with some special pattern in a large data set. Pattern matching is often studied under the names of querying, sub-string matching, shift finding. The application of pattern matching can be found everywhere. Some such applications include audio/image signal matching, bioinformatics data matching, text matching in large database, etc. Several algorithms have been designed in the literature for various versions of the pattern matching problem. In this thesis, we focus on pattern matching based on sparse transform computation. In the next section, we briefly introduce two important prior works and discuss the relation of these works and ours.

2.1 Retated Work

In this section, we present two related works: the Fast Sparse 2-D DFT Computation using Aliasing-based Sparse Transform(2D-FFAST) algorithm by Ong[1] and the Sub-string/Pattern Matching in Sub-linear Time Using a Sparse Fourier Transform Approach from Janakiraman [2]. Then, we introduce the main idea behind our work on the 2D-Pattern Matching problem.

2.1.1 2-D FFAST

[1]The problem considered is computing the 2D-DFT of a 2D signal of size $N_x \times N_y$ when the transform is sparse. If N_x and N_y are co-prime, the problem can be reduced to computing the DFT of a corresponding 1-D signal. And this can be done by using 1-D FFAST algorithm[3] which also uses a very similar computation scheme. When N_x and N_y are not co-prime, algorithm require a factorization of $N = N_x \times N_y = \prod_{i=0}^{i=d-1} Q_i$ where d is a well chosen constant.

The idea of the 2D-FFAST algorithm is illustrated by the following example. A 2-D signal x of size 6×6 is considered. The 2-D DFT of this signal is X of sparsity $k = 4$ with

[1]Reprinted with permission from "Fast sparse 2-D DFT computation using sparse-graph alias codes" by Frank Ong; Sameer Pawar; Kannan Ramchandran, 2016. IEEE International Conference on Acoustics, Speech and Signal Process-ing (ICASSP), pp. 4059–4063, Copyright [2016] by IEEE.

$X[1][3], X[2][0], X[2][3], X[4][0] \neq 0$.

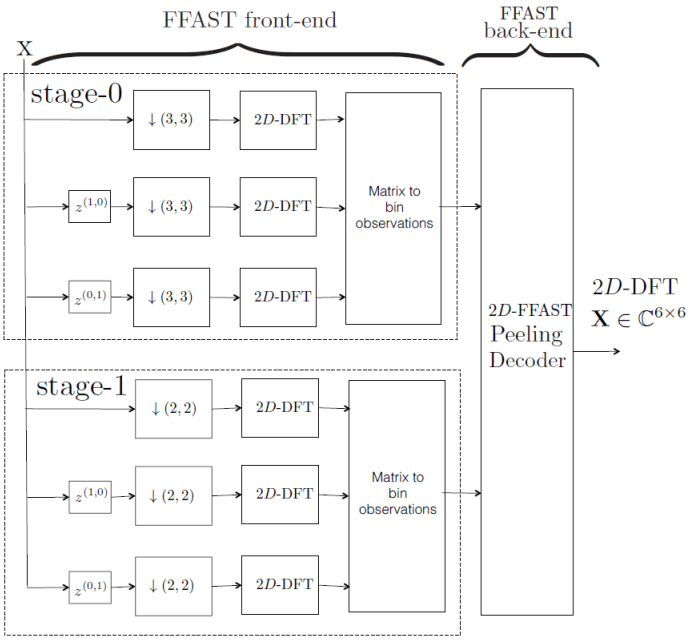

Figure 2.1: 2D-FFAST Architecture with d = 2. This figure is from [1] with permission.

As shown above, the 2-D FFAST algorithm architecture consists of two main parts, the front end that performs down sampling and a peeling decoder at the back end. There are two down-sample stages with different down-sample rates. The two dimensions of the input signal do not necessarily have to be the same but to keep demonstration simple, we assume they are the same. In each stage the signal has three shifts: one zero shift and two non-zero shifts along each dimension. Then the smaller size 2-D DFT is computed. The output of the front-end are bins for observation, which are vectors of DFT coefficients. For example, the bin $y_{0,3} = (X[1][3], e^{2\pi i 1/6}X[1][3], e^{2\pi i 3/6}X[1][3])$. The value of the coefficient is the first entry of this vector and the position information is contained in the phase of the other entries.

The relationship between the non-zero DFT coefficient and the bin observation vectors is given in 2.2. The bin is classified as a zeroton if there is no contribution from any of the nodes on the left. It is called a singleton if it is connected to one non-zero coefficient. And it is called a multiton

if it is connected to more than one non-zero coefficient. A bin can be classified into these three classes by doing the ratio-test[3]. Coefficients can be detected and located if it falls in a singleton.

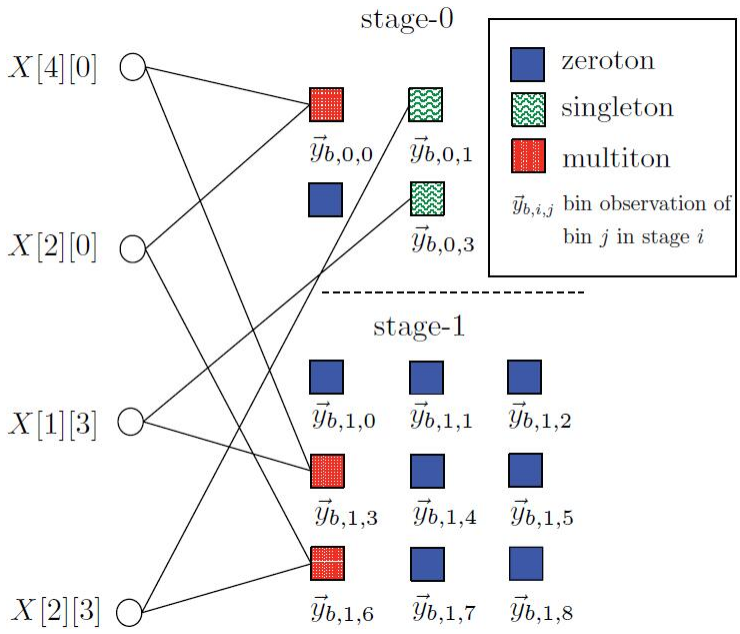

Figure 2.2: Bi-partite Graph of the non-zero coefficient and the bin observation. This figure is from [1] with permission.

The back-end peeling process is an iterative algorithm. In each iteration, a singleton is detected by doing the ratio-test, the value and the index of the coefficient corresponding to it can be found. Then the contribution from this coefficient is removed. Accordingly, the edges connected to this coefficient are deleted in the bi-partite graph. By doing so, the degree of some bins connected to it will drop and some multiton will turn into singleton which can be decoded. The peeling decoder algorithm will stop when all the coefficients are found. The high probability of success is achieved by downsampling the signal with well chosen factors according to Chinese Reminder Theory (CRT).

2.1.2 Sub-string/Pattern Matching in Sub-linear Time Using a Sparse Fourier Transform Approach

In this section, we briefly introduce how to convert a pattern matching problem into a sparse signal recovery problems. Here we only give a brief introduction of exact pattern matching. $\vec{x} = (x[0], x[1], ..., x[N-1])$ is a N-point length signal which represents the data base. $\vec{y} = (y[0], y[1], ..., y[M-1])$ is a shorter signal of length M. The question is whether $\vec{y}$ is a sub-string of $\vec{x}$ and where are the locations of these sub-strings $\vec{\tau} = (\tau(1), \tau(2), ..., \tau(t))$. If there can be more than one sub-strings to match and all of them have to be found. Binary cases is considered in this work.

The idea is that the correlation signal $\vec{r} = \vec{x} \otimes \vec{y}$ can be used to find the matching position. Generally, the correlation signal $\vec{r}$ will have a large value at the matched position and small value at non-matched positions.

$$r[m] = \begin{cases} M, & if m \in \tau \\ n_m, & m \in [N] - \tau \end{cases} \tag{2.1}$$

This is a true in many applications and a well known one is Global Positioning System (GPS). In GPS synchronization problem, the satellite transmit CDMA code to the receiver. Cross-correlation signal is computed locally at the receiver, a peak will appear at the matched position so that the receiver can infer the delay of the propagation[7]. The cross-correlation can be computed by using the DFT of x and y as follows:

$$\vec{r} = \mathcal{F}_N^{-1}\{\mathcal{F}_N\{\vec{x}\} \odot \mathcal{F}_N\{\vec{y}\}\} \tag{2.2}$$

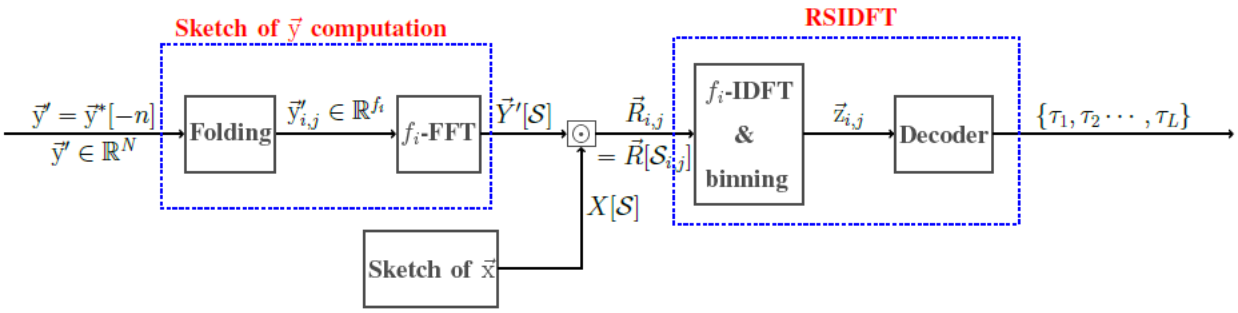

Figure 2.3: Flow chart of the algorithm from [2]

.

where $\mathcal{F}_N$ represents the N-point Discrete Fourier Transform and $\mathcal{F}_N^{-1}$ is the N-point inverse Discrete Fourier Transform (IDFT). Since the data base $\vec{x}$ is known in advance, one time DFT of it can be computed and stored locally. There is also an efficient method to compute the N-point DFT of the $\vec{y}$ by leveraging the fact that it contains many zeros.

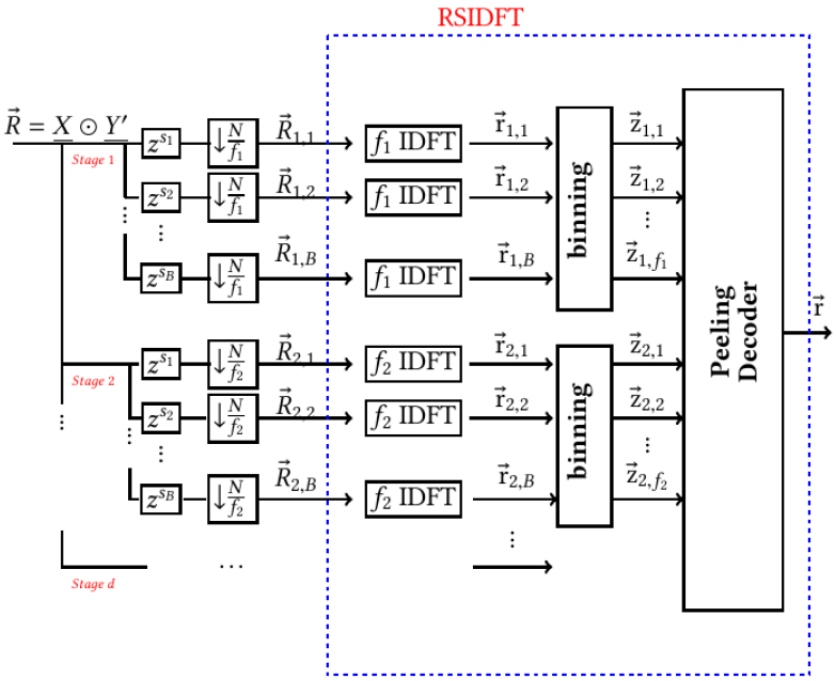

Figure 2.4: Sparse signal recovery algorithm from [2].

Once the production of these two N-point DFT coefficients are obtained, the Robust Sparse Inverse Discrete Fourier Transform (RSIDFT) will be performed to locate the peak of the cross-correlation signal. The input signal is $\vec{R} = \mathcal{F}_N\{\vec{r}\}$. N can be factorized as $N = \prod_{i=0}^{i=d-1} f_i$ where f_is are relatively prime. There are d stages to down-sample the $\vec{R}$ and compute the IDFT of the shorter signals. In the ith stage, the down-sample rate is N/f_i and there are B shifts $\vec{s} = (s_1, s_2, ..., s_B)$. The IDFT of these shifted and down-sampled signals is viewed as some bin observations. $\vec{r}_{i,j}$ is the f_i-point IDFT of $\vec{R}_{i,j}$ which is the shifted and down-sampled signal in the jth branch of ith stage and $\vec{z}$ are given as follow:

$$\vec{z}_{i,k} = [r_{i,1}[k], r_{i,2}[k], ..., r_{i,B}[k]]^T \tag{2.3}$$

and due to the shift and down-sample property of fourier transform, $\vec{z}$ can also be given as:

$$\vec{z}_{i,k} = W_{i,k} \times [r[k], r[k+f_i], ..., r[k+(g_i-1)f_i]]^T \tag{2.4}$$

where $g_i = N/f_i$ and $W_{i,k}$ is defined as

$$W_{i,k} = [\vec{w}^k, \vec{w}^{k+f_i}, ..., \vec{w}^{k+(g_i-1)f_i}] \tag{2.5}$$

where $\vec{w}^k = [e^{j2\pi k s_1/N}, e^{j2\pi k s_2/N}, ..., e^{j2\pi k s_B/N}]$.

Unlike to the 2D-FFAST work, the ratio-test can't be used here because the value of non-matched position is not strictly zero. So thresholds can be set for bin classification. Only when the value of first entry fall into $(0.5M, 1.5M)$, a bin can be classified as a singleton. A similar bi-partite graph is given below and the peeling decoder algorithm can be used for the recover of the signal.

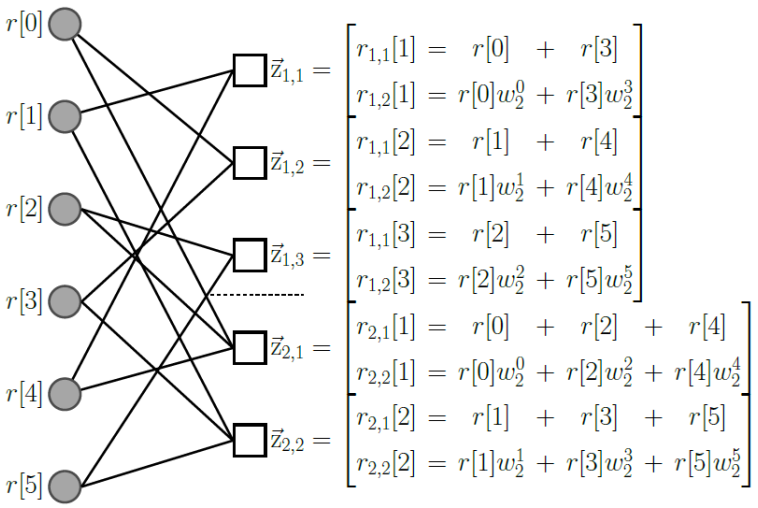

Figure 2.5: Bi-partite graph for RSIDFT.

2.1.3 Proposed Work

We design a 2-D pattern matching algorithm which is build on these two recent works. The work in [2] shows that the location of matched pattern can be given by the cross-correlation signal of the pattern and the database. The major difference of our work and [2] is the shape of the data. Work in [1] gives a hint how to handle the data of more than one dimension.

In the next section, we presents the main result of our 2D-pattern matching algorithm.

2.2 2D-Pattern Matching

In this section, we describe our work of 2D-pattern Matching algorithm.

2.2.1 System Model and Notations

We consider the problem of locating the position of a small 2D signal in a large 2D data base. One or more patterns can be found in this data base. Unlike the 1D problem, the shape of the data is different. A novel algorithm is designed to find the pattern with low sample and computational complexity.

In this thesis we represent the input signal as x of size N by N, a large image in general. The pattern is a sub-image y of size M by M. It is not required the input has the same size for both

dimension but it is easy to describe the algorithm when the size is the same in both dimensions. There is one or more patterns in the input signal and the proposed algorithm is designed to find out the position of these patterns by using only a sub-sampled version of x. To searching for sub-image y, the cross-correlation of x and y has to be computed and it's denoted as $r = x * y'$, and the coordinate of the maximum of r gives the position of the pattern. We use binary data (For each pixel, the value is given by a uniformly distribution over $\{-1, +1\}$.) to do the simulation and the result will be given in the following sections.

Symbol	Meaning
x	database
y	sub-image
N	Size of the database
$M = O(N^{\mu})$	Size of the sub-image
$L = O(N^{\lambda})$	Number of matches
d	Number of stages
$f_i \approx N^{\alpha}$	Size of small point 2D-IDFT at each stage-i
B	Number of shifts in each stage

Table 2.1: Parameters involved in describing the algorithm

To describe the algorithm, we use lowercase letters to refer time domain signals and use uppercase letters to refer frequency domain signals. Some important parameters are shown in the table 2.1.

2.2.2 2-D Pattern Matching Algorithm

In this section we describe the query algorithm for locating a small 2-D block in a relatively large image. The main idea is that the cross correlation signal r is very sparse since it only have

large coefficients of value M^2 (for the exact pattern matching) at the matching positions $P = \{P_1, P_2, P_3, ..., P_L\}$ and behaves like noise close to zero where it is not matched. The index of the location can be known as long as the peak in the cross-correlation signal is found. So this can be treated as a sparse signal recovery problem.

To computed the cross-correlation efficiently, the algorithm is built on the fact that the DFT of r can be given as the product of Fourier transform of x and y', i.e., $y'[i,j] = y^*[-i,-j]$. The cross-correlation signal r is given by computing the IDFT of that product:

$$r = F_{N\times N}^{-1}\left\{F_{N\times N}(x) \odot F_{N\times N}(y')\right\} \tag{2.6}$$

where $F_{N\times N}\{\cdot\}$ denotes the 2-D discrete fast fourier transform of size $N \times N$. The 2D-DFT of the data base x can be one-time computed and stored locally and there would be an efficient way to computed the 2D-DFT of y' of size $N \times N$ and it will be introduced after the description of the recovery algorithm. The flow chart is given below.

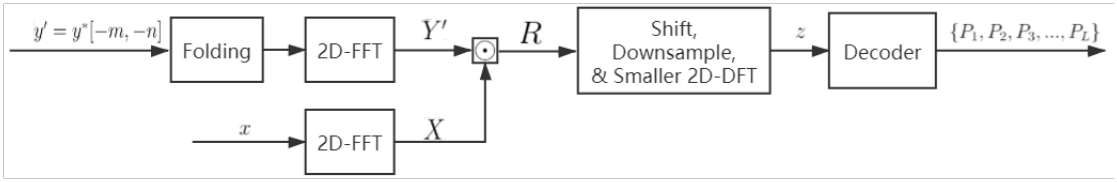

Figure 2.6: Flowchart of the proposed algorithm.

The 2D-DFT of the cross-correlation signal r is obtained by the production of the 2D-DFT of x and y'. It is denoted as R. The shape of R is $N \times N$. A description of the algorithm follows. There are d shift and down-sample stages. Firstly, in each stage, the signal will be shifted in two groups for each dimension according to a series shifts $\{s_1, s_2, ..., s_B\}$. $s_1 = 0$ and the rest of the shift can be any integers in $[1 : N-1]$. In each group of shifts, the cross-correlation signal is shifted along one dimension and remind unmoved for the other. In the ith stage, each branch of the shifted

signal is down-sampled by a factor of f_i. We choose d factors $\{f_1, f_2, ..., f_d\}$ that $N = \prod_i f_i$. The shifted cross-correlation signal is downsampled into the shape of $f_i \times f_i$, and the corresponding down-sample rate is $g_i = \frac{N}{f_i}$. When the small size shifted and downsampled signal is obtained, a 2D-IDFT of size $f_i \times f_i$ is computed in jth branch of ith stage to get $R_{i,j}$ which is an aliased observation of the original cross-correlation signal r. The aliased signal in stage i with shift s_j is give as follow:

$$r_{i,j}[m,n] = \sum_{p \mod f_i = m, q \mod f_i = n} r[p,q] \times \omega^{ps_j} \tag{2.7}$$

if the shift is along the first dimension, where $\omega = e^{j\frac{2\pi}{N}}$. If the shift is along the second dimension, the power over ω should change to qs_j respectively.

This is from the property of discrete Fourier transform that sub-sampling in the frequency domain will lead to an aliasing in time domain. By having B aliased signals, bin observation z_i is given as follow:

$$z_i[m,n] = [r_{i,1}[m,n], r_{i,2}[m,n], ...r_{i,B}[m,n]]^T \tag{2.8}$$

and this gives a total number of $f_i \times f_i$ bin observation in each group of stage i. The architecture of the algorithm is shown below.

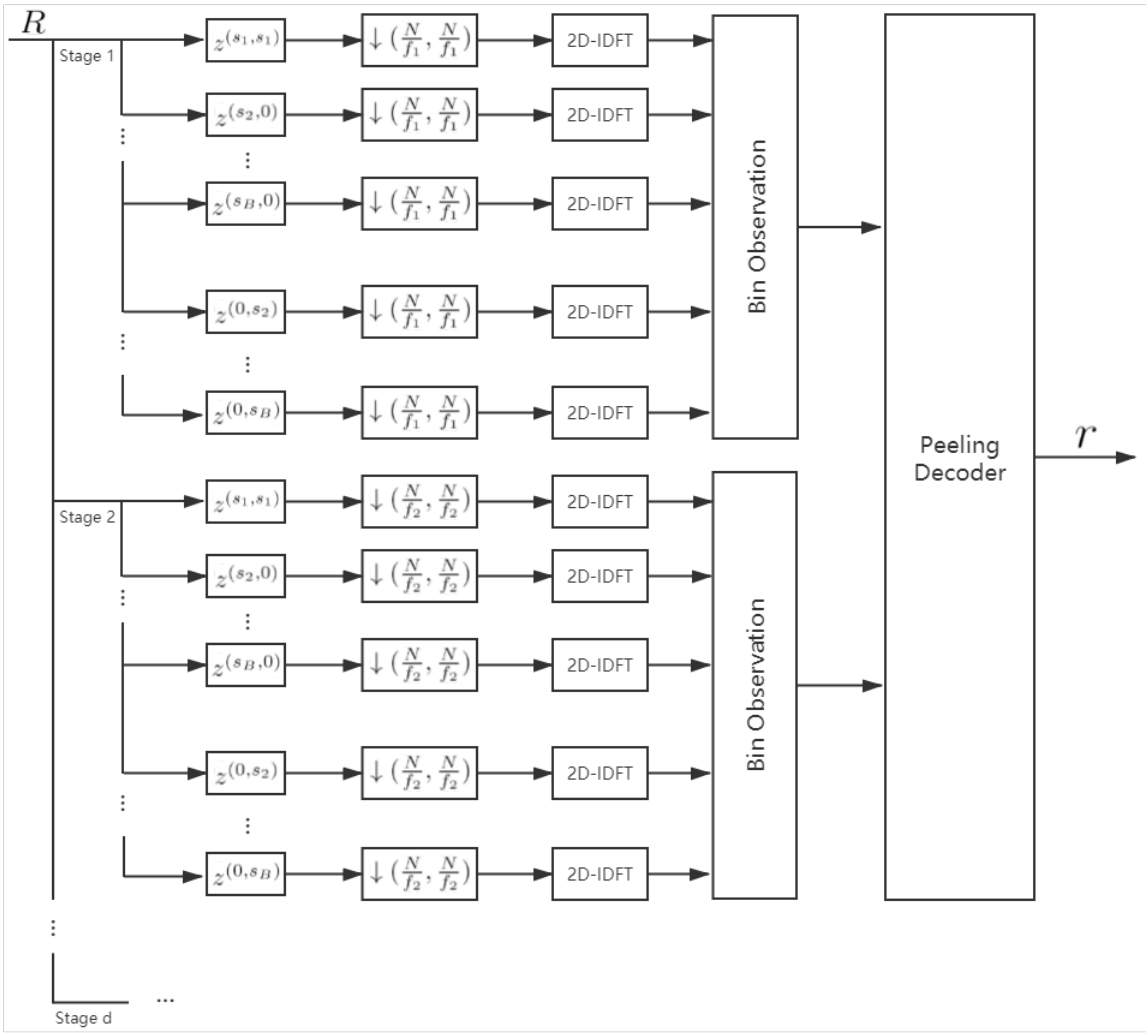

Figure 2.7: Architecture of the pattern matching algorithm.

2.2.2.1 Bin Classification and Decoding

A classification of bin observation is needed after it's obtained. The bin observation can be defined into these classes:

- $Zeroton$: if there is no peak in the original cross-correlation signal aliased in this bin.
- $Singleton$: if there is one peak in the original cross-correlation signal aliased in this bin.
- $Multiton$:if there is more than one peak in the original cross-correlation signal aliased in

this bin.

We use $C(z_i[m,n])$ to denote the class of bin observation $z_i[m,n]$. And the class of a bin observation $z_i[m,n]$ can be given by a thresholding technique:

$$C(z_i[m,n]) = \begin{cases} Zeroton, & z < \frac{1}{2}M^2 \\ Singleton, & \frac{1}{2}M^2 < z < \frac{3}{2}M^2 \\ Multiton, & z > \frac{3}{2}M^2 \end{cases} \tag{2.9}$$

where z is the first entry of $z_i[m,n]$, and the threshold of M^2 comes from the fact that for binary[2] data, the value of the peak in the cross-correlation signal is the size of the pattern.

Since the $Zeroton$ contains no peak, there is no matching information can be known from it. For the $Multiton$ with more than one peaks added up together with the multiplication of the phase introduced by the shift, things can be complex. So the $Singleton$ is what we are interested in and we present a method to infer the position of the peak at the original cross-correlation r by analysing the bin observation. We consider the $Singleton$ obtained in the stage i of the group shifting along the first dimension. The first index of the peak in r is given by:

$$\hat{p} = \arg \max_{p:p \mod f_i = m, 0 \le p \le N} z_i[m,n]^\dagger w^k \tag{2.10}$$

where † means the conjugate transpose and w^k is defined as:

$$w^k = [\omega^{ks_1}, \omega^{ks_2}, ...\omega^{ks_B}]^T \tag{2.11}$$

The second recovered index $\hat{q}$ is given in the similar way with the bin observation from the group along the other dimension. The value of the recover peak is decoded as:

$$r[\hat{p},\hat{q}] = M^2 \tag{2.12}$$

[2]The binary is referred to $\{-1,+1\}$. For the simulation over real image, the threshold M^2 is subsided to the possible maximum value of correlation signal of the pattern

2.2.2.2 *Peeling Process*

The peeling decoder algorithm is designed to solve the entire sparse recovery problem in an iteration way. In each iteration, once a $singleton$ is detected, the index of the peak in the cross-correlation signal r can be found. Then the peeling process is done by subtracting the contribution of this recovered coefficient in the signal. By doing so, the degree of some $Multiton$ will decrease and turn into $Singleton$ that can be decoded. The algorithm runs iterative until all the peak in r are found.

2.2.2.3 *Sketch of y'*

It should be noticed that the $F_{N\times N}(r)$ can be computed as a point-wise product of $F_{N\times N}(x)$ and $F_{N\times N}(y')$.

$$F_{N\times N}(r) = F_{N\times N}(x) \odot F_{N\times N}(y') \tag{2.13}$$

So only the sub-sampled version of $F_{N\times N}(y')$ is needed to obtain $F_{N\times N}(r)$. Based on the sample property of the Fourier transform, this signal can be obtained by multiplying the time domain signal with a corresponding phase term followed by a folding operation. If the shift s_j happens along the first dimension, the corresponding time domain 2D-signal is given by:

$$y''[m,n] = \sum_{p \mod f_i = m} \; \sum_{q \mod f_i = n} y'[p,q]\omega_{s_j}^{p}. \tag{2.14}$$

The sub-sampled signal of $F_{N\times N}(y')$ is $F_{N\times N}(y'')$. To obtain all the sub-sampled signals in all the $2B$ branches in d stages, $2dB$ times $N^{2\alpha}$ points 2D-DFT is required.

2.2.3 Sample and Computation Complexity

2.2.3.1 *Sample Complexity*

The samples used in the sparse recovery algorithm comes from the shift and down-sample process in each branch of $f_i \times f_i \approx N^{2\alpha}$ points. By adding all samples from $2B = O(\log(N))$ branches, each stage has $N^{2\alpha}O(\log N)$ samples. There are d stages in each iteration, so the sample

complexity is given by

$$O(N^{2\alpha} \log N). \tag{2.15}$$

2.2.3.2 Computation Complexity

There are several operations involved in the proposed pattern matching algorithm.

Operation 1: The 2D-DFT of the database must be pre-computed once. So the computation complexity of that will not be included in the discussion.

Operation 2: the computation of 2D-DFT of y' needed two operations: folding and adding of complexity $O(M^2)$, and small size 2D-DFT of complexity $O(N^{2\alpha} \log N^{2\alpha})$. So the total computation for compute 2D-DFT of y' is given as:

$$O(\max(N^{2\mu} \log N, N^{2(1-\mu)} \log^2 N)). \tag{2.16}$$

Operation 3:The computation in each branch needed for the small-size 2D-DFT is $O(N^{2\alpha} \log N)$. For the $Singleton$ decoding process, there are $\frac{N}{f_i} \approx N^{1-\alpha}$ correlations of length B and the complexity for doing this is $O(N^{1-\alpha} \log N)$. There are at most L matching positions so the total complexity for this operation is $O(N^{1-\alpha+\lambda} \log N)$.

The total computation complexity for the operation 3 is given by:

$$O(\max(N^{2(1-\mu)} \log^2 N, N^{\mu+\lambda} \log N)) \tag{2.17}$$

Overall Computation Complexity: The overall computation complexity is given by:

$$O(\max(N^{2(1-\mu)} \log^2 N, N^{\mu+\lambda} \log N)) \tag{2.18}$$

2.2.4 Design of 2D Patterns for Matching

One possible application of 2D pattern matching is in vision-based positioning systems for robots or drones. The problem can be described as locating the position of a drone by analyzing the image captured by the camera built on it. One or more patterns are placed on the ground as markers and by detecting the position and rotation of these markers, the drone can determined its position.

The marker pattern should have the following characteristics - (i) the pattern should appear pseudo-random such that correlation of the pattern with parts of an image are likely to be small except at the matching locations, i.e., y exhibits high local variation,(ii) pattern matching should be robust to rotation of the images and (iii) pattern matching should be robust to scaling. It is easy to see that traditional patterns like stop signs do not satisfy these characteristics and hence, the pattern has to be carefully designed.

We propose the use of the following two patterns shown in Fig. 2.8. The first pattern is a set of concentric rings with the widths increasing geometrically by a factor θ. This pattern will be immune to rotation and has a periodic invariance - when the pattern is zoomed in and out by a factor of θ, it looks exactly the same as the original one). The second pattern has the similar invariance property with respect to rotation. By doing this, only a few patterns are needed to detect the potential pattern in x.

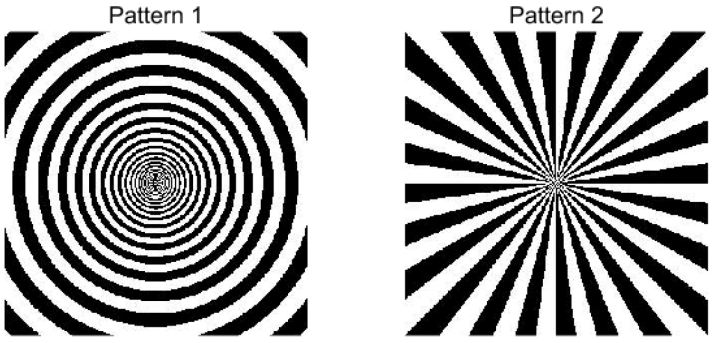

Figure 2.8: 2 patterns with immunity to scaling and rotation.

2.2.5 Simulation Results

We tested the algorithm in two cases: with binary randomly generated data, and when the pattern in 2.8 is embedded within a real image.

2.2.5.1 Performance for Binary Randomly Generated Data

Here, we generated a binary image y where each pixel uniformly chosen to be 0 or 1 and the values are chosen independently. The image y is 'planted' within a larger binary image x at several locations and the rest of the pixels in x are assumed to be generated uniformly at random from 0 or 1 and are chosen independent of each other.

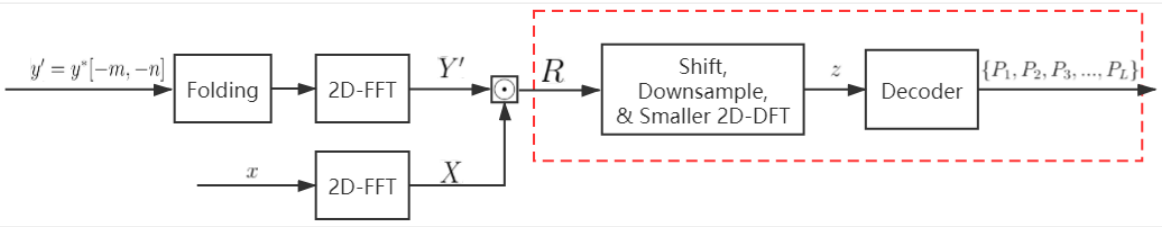

Figure 2.9: Flowchart of the 2-D pattern matching algorithm.

First we want to test the performance of the sparse recovery ability of the proposed algorithm. What in the red block is the testing part of this section. R is the 2-D DFT of the sparse cross-correlation signal r. $\{P_1, P_2, P_3, ..., P_L\}$ are the positions of all large values in r.

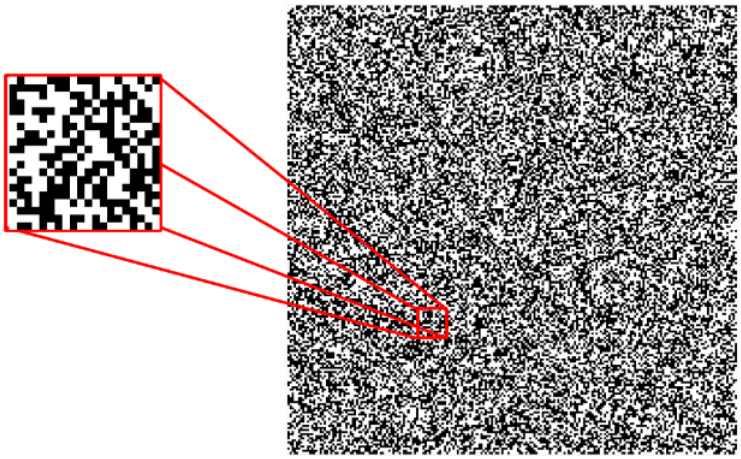

Figure 2.10: Sub-image and database

To generate R, we assume the database and the pattern are both binary of $\{-1, +1\}$ at each pixel so that the value of the cross-correlation signal r is N^2 at matching positions and small value converging to zero at non-matching positions. An example is given as above. As shown in Figure 2.10, The database is a large binary 2-D signal and the pattern is a sub-image located somewhere in the database. An example of cross-correlation signal of the database and the sub-image is given in Figure 2.11.

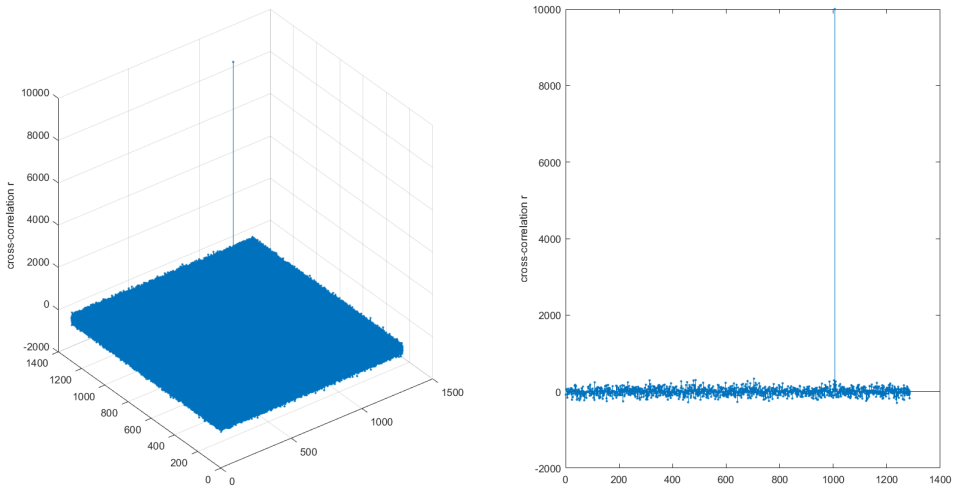

Figure 2.11: Cross-correlation signal.

The figure on the left is an example of a cross-correlation signal. The size of the database is 1287×1287 ($M = 9 \times 11 \times 13$). The size of the sub-image is 100×100 ($N = 100 \times 100$). The pattern appears once in the database at the position $[1117, 1006]$. The figure on the right plots the 1117th row of the 2-D cross-correlation signal, and the value at 1006th position if of value 10000 which is the size of the sub-image.

We simulate the sparse recovery computation to obtain the performance of the algorithm with different sparsity. Sparsity ratio is defined as L/N^2 which represents the sparsity level of the cross-correlation signal. The size of the database is 1287×1287 ($M = 9 \times 11 \times 13$) and the peak value is 10000. The algorithm is carried out with 3 stages, and in each stage, there are 5 shift and down-sample branches. Figure 2.12 gives the result of the simulation. At sparsity ratio of 0.003, around 5000 matching positions can be found successfully with the probability error under 1%.

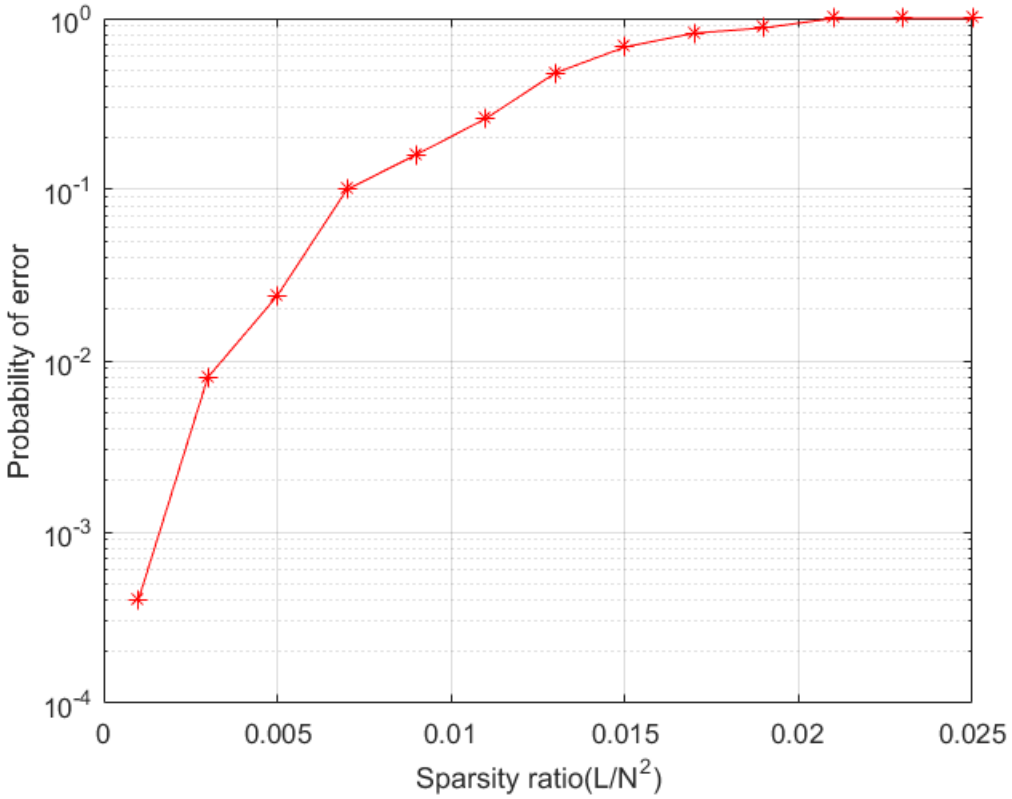

Figure 2.12: Performance with different sparsity.

2.2.5.2 Performance with designed patterns

In this section, we show the performance when our designed patterns are embedded in real images. An example is given below in figure 2.13.

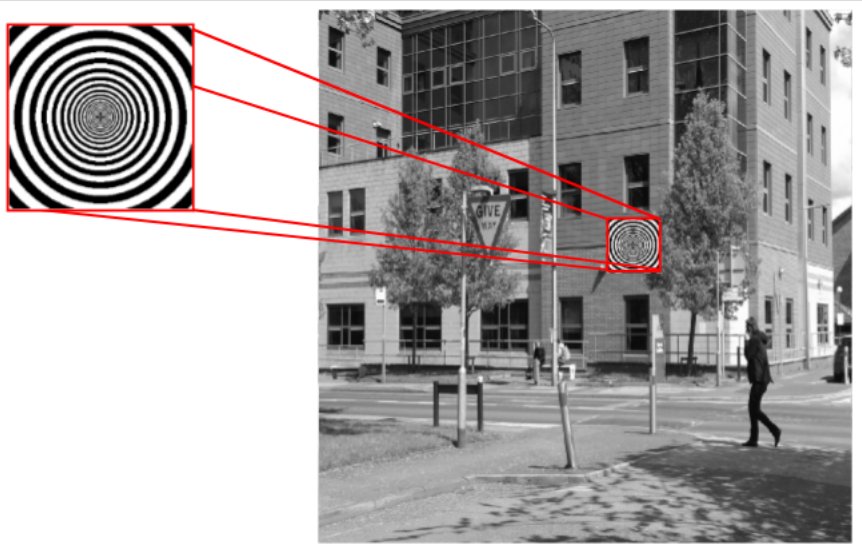

Figure 2.13: Pattern with real image.

The large image of a typical street view is given as the database. The pattern is a series circle with the radius is a geometric sequence. First we test the peak value of a pattern of fixed size correlated with itself of varying sizes. The value periodically reaches the local maximum when the pattern is the same as part of it. The plot is given below. It turns out that as long as the size of the desired pattern in the large image is larger than 40×40 pixel-wise, the algorithm can detect and find the position of it correctly.

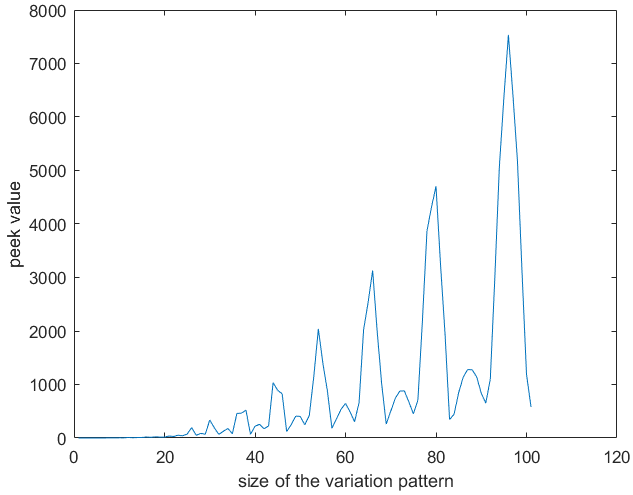

Figure 2.14: Correlation of the patterns.

Since the pattern is built for a vision-based locating system, there will not be too many patterns in the image. It has been shown that the 2-D pattern algorithm has very good performance dealing with lots of patterns in the database. What is tested in this section is the performance of the algorithm against the noise. The data is generated as follow:

1 Three patterns are generated as template. The first pattern is of size $M \times M$ with $M = 100$, the common ratio of the associated geometric sequence is $\theta = 1.1$. The other two patterns are enlarged versions of the first one with sizes of $102 \times 102, 104 \times 104$ respectively.

2 A resized version of the pattern is placed in a random location within the large real image. The size of the pattern is uniformly distributed over $[90, 110]$.

3 Gaussian random noise is generated and added to the image. The performance is tested with the standard deviation σ varying in range of $[0, 260]$.

The detection of any one of the three templates and find the right position is considered as a success.

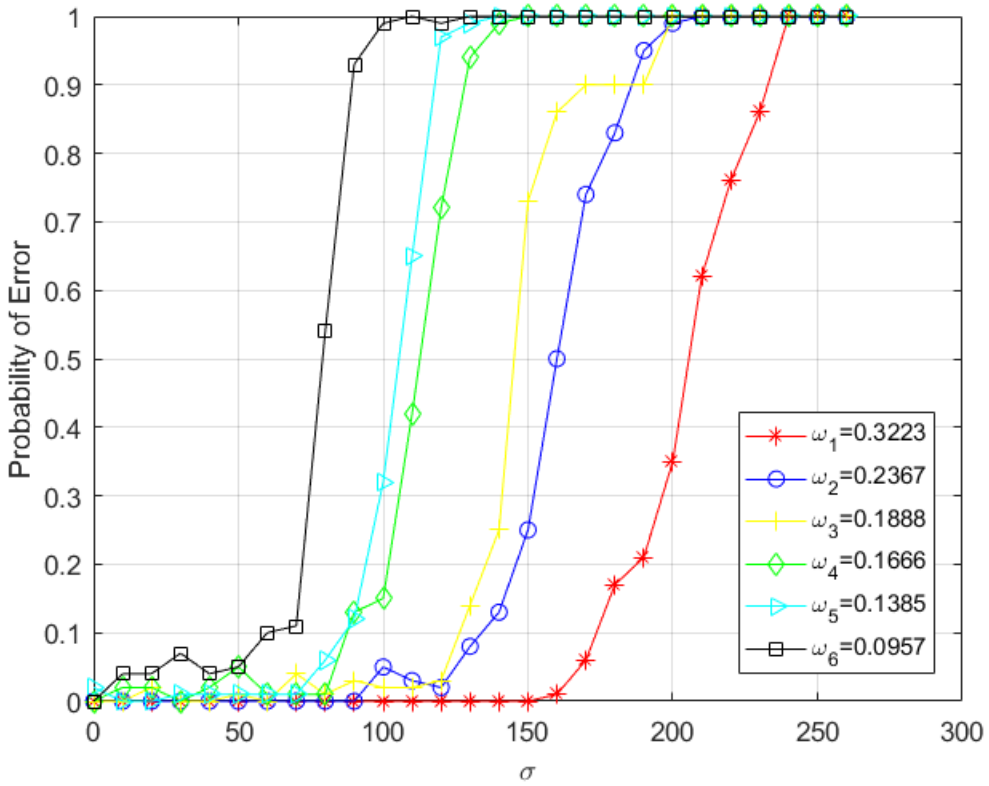

Figure 2.15: Performance when image is corrupted with Gaussian noise.

σ is the the standard deviation of the Gaussian noise. The different color represent the simulation carried out with different sample ratio. When the noise is getting large, some pixels may have a value out of the range of $[0, 255]$. We keep that value to implement this simulation. Thresholds can be given to make sure all the value is stay in the range of $[0, 255]$ so that it make sense for images.

A plot of the probability of error as a function of σ is shown in Fig. 2.15. Suppose the number of sample pixels used in the each implementation is S_a. Then the sample ratio is define as $\omega = \frac{S_a}{N^2}$. Different colors represent the simulation is done with the different sample ratio.

It can be seen that when the noise standard deviation is small, our algorithm is able to identify the patterns with high probability even when the sampling ratio is as low as 9.5%. This is the primary advantage of our algorithm. For a fixed sampling ratio, as the noise variance increases, the error probability increases and for a fixed noise variance, as the sample ratio increases, the probability of error decreases.

3. IMPROVED SPARSE WALSH-HADAMARD TRANSFORM FOR BINARY DATA

In this chapter, we present an adaptive algorithm for sparse signal recovery with the sparsity in the domain of the Walsh-Hadamard Transform (WHT) when the non-zero coefficients are ones.

3.1 Introduction

Walsh-Hadamard fransform (WHT) is another useful discrete transform similar to the famous discrete Fourier transform. It is widely used in the signal processing, video or image compression, and quantum computing. Here we give the definition of the Walsh-Hadamard transform and introduce some of its properties which play an essential role in our algorithm.

3.1.1 Walsh-Hadamard Transform

The Walsh-Hadamard Transform X of a signal x of length $N = 2^n$ can be defined in the following way:

$$X[k] = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{\langle i,k \rangle} x[i], \tag{3.1}$$

where $\langle ., . \rangle$ is defined as the inner product of two binary vector over $\mathbb{F}_2^n$. Compared to the Fourier Transform, the most noticeable difference is that the base of the exponent is -1 instead of e while the idea is very much alike DFT. More importantly, the i, k in the equation above denote binary index vectors. Some basic properties of WHT is given as follow. These properties are of vital importance in our proposed algorithm.

1 Shift/Modulation

Suppose X is the WHT of x, then a shift of p in x will give a corresponding sign pattern to the coefficient of X.

$$x_{i+p} \overset{WHT}{\longleftrightarrow} X_k (-1)^{<p,k>} \tag{3.2}$$

This property will help to give the index information of the non-zero coefficient later.

2 Down-sampling/Aliasing

Given $B = 2^b$ and $N = 2^n$ where $b < n$, a downsample matrix can be formed in the following way:

$$\Psi_b = [\mathbf{0}_{n\times(n-b)} I_b]^T, \tag{3.3}$$

by multiplying an index $m \in \mathbb{F}_2^n$ with this matrix, the first $n - b$ bits are set as 0 and the last b bits remain unchanged. Then the down-sampling and aliasing property of WHT can be given as:

$$x_{\Psi_b i} \overset{WHT}{\longleftrightarrow} \sqrt{\frac{B}{N}} \sum_{j\in N(\Psi_b^T)} X_{\Psi_b k+j} \tag{3.4}$$

According to this transform relation, a down-sample in time domain will lead to an aliased spectrum in the frequency domain. Once the down-sample matrix ψ_b is chosen, the index of these aliased frequency coefficient can be known.

3 Permutaion

If $\Sigma \in GL(n, F_2)$, where $GL(n, F_2) = \{A \in F_2^{n\times n} | A^{-1} exists\}$,then

$$x_{\Sigma i} \overset{WHT}{\longleftrightarrow} X_{\Sigma^{-T} k} \tag{3.5}$$

$x_{\Sigma i}$ is the permutation of x_i by multiplying Σ with the original index. The equation means given X_k is the WHT of x_i, the WHT of the permutation signal $x_{\Sigma i}$ is a permutation signal of X_k.

3.1.2 Graph learning problems and cut function

In computer science, graph is a data structure which plays an important role both in academic research and industry applications. So how to learn a graph from its evaluation with a few samples is an interesting problem. Cut function is such an evaluation which maps the sub-graph of a ground graph to real numbers.

Consider a graph $G = (V, E, w)$, the cut function of it is defined as

$$x(A) = \sum_{s \in A, t \in V \setminus A} w(s, t), \tag{3.6}$$

for every $A \subseteq V$. The subset A can be represented by a indicator vector $t \in F_2^n$ where $n = |V|$. So the length of this cut function is $N = 2^n$. The WHT of x is:

$$X[f] = \begin{cases} \frac{1}{2} \sum_{s,t \in V} w(s, t), & iff = (0, 0, ..., 0) \\ -w(s, t)/2, & \text{if } f_s = f_t = 1 \text{ and } f_i = 0 \forall i \neq s, t \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

It's obvious that the WHT of the cut function is sparse in that domain with only $|E| + 1$ coefficients not zero (One edge in the graph corresponds to one non-zero coefficient as the second circumstance in 3.7). So learning a graph from the cut function can be solved as a sparse signal recovery problem.

3.2 Retated Work

We brief introduce the work of [5] in this section.

3.2.1 Fast Hadamard Transform with Sublinear Sparsity in the Transform Domain

Considering a time domain signal x of length n whose WHT X is sparse with k non-zero elements randomly chosen from a continuous distribution. A hash function is designed for the spectral domain of the signal. A $B = 2^b$ dimensional signal is defined by:

$$u_{\Sigma, p(i)} = \sqrt{\frac{N}{B}} x_{\Sigma \Psi_b i + p}. \tag{3.8}$$

where i denotes the index of time domain signal, k denotes the index of frequency domain signal, and $p \in \mathbb{F}_2^n$ is the shift.

Its WHT is:

$$U_{\Sigma, p(k)} = \sum_{j \in F_2^n | \Psi_b^T \Sigma^T j = k} X_j (-1)^{<p, j>}. \tag{3.9}$$

This gives B bins to hash the spectrum, then a ratio test is given to find the bin contains only one non-zero coefficient. Ratio is given by:

$$r_{\Sigma,p}(k) = \frac{U_{\Sigma,p}(k)}{U_{\Sigma,0}(k)}. \tag{3.10}$$

For $n-b$ well chosen shift p, if $|r_{\Sigma,p}(k)| = 1$, then there is only one non-zero coefficient in that bin. Further more, the index of that coefficient is given by:$j = \Sigma^T(\Psi_b k + \vec{v})$, where

$$\vec{v}_d = \begin{cases} 1_{\{r_{\Sigma,\sigma_d}<0\}}(k), & d \in [n-b] \\ 0, & otherwise \end{cases} \tag{3.11}$$

σ_i is the i^{th} columns of Σ.

An iteration algorithm of peeling decoder can be designed based on this ratio test.

3.2.2 Improved Sparse Walsh-Hadamard Transform

There is a work [9] showing that the algorithm introduced in the previous section can be speed up by recovering two non-zero coefficients in one iteration[9]. This can be done in the following way.

The downsampling matrix is formed in the same way as in 3.3. The shifts are chosen so that the sign of the aliased coefficients appearing like $\{+1, -1\}$ with different frequencies. If there are two non-zero coefficients chosen from a continuous distribution among these aliased coefficients, their sum is given by the observation from the non-shift branch. The flow chart of this algorithm is given as follow:

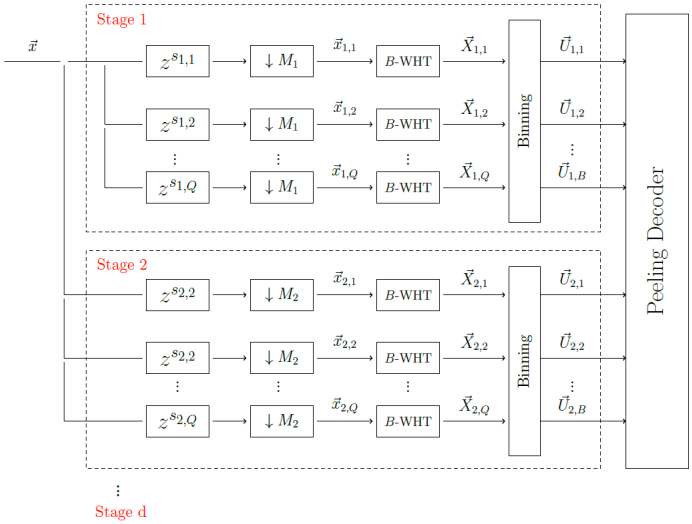

Figure 3.1: Schematic of improved sparse Walsh-Hadamard algorithm

U is the bin observation from each stage, and it is given as :

$$U = \Phi X \tag{3.12}$$

where U is the sensing matrix of size $\log_2 \frac{N}{B} + 1$ by $\frac{N}{B}$. A example is given for $\frac{N}{B} = 8$:

$$\begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{bmatrix} \tag{3.13}$$

For ith element in U, it has the value of $a_i X[j_1] + b_i X[j_2]$, a and b is chosen from$\{+1, -1\}$. A example of the decoder process is given to illustrate the algorithm.

We consider 4 coefficients are aliased together and two of them are non-zero:$X[2] = 5, X[3] =$

10 with $j_1 = 2, j_2 = 3$. Bin observation U is given as:

$$U = \begin{bmatrix} U[1] \\ U[2] \\ U[3] \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \end{bmatrix} \times \begin{bmatrix} 0 \\ X[2] \\ X[3] \\ 0 \end{bmatrix} = \begin{bmatrix} 15 \\ -5 \\ 5 \end{bmatrix}. \tag{3.14}$$

$U[1]$ gives the sum of the two non-zero coefficients $X[j_1]$ and $X[j_2]$ equals 15. $U2$ is -5 whose absolute value is not the same as $U[1]$, so the sign of these two coefficients split. So $j_1 \in \{1, 2\}$ and $j_2 \in \{3, 4\}$. $U[3] = 5$ whose absolute value is the same as $U[2]$ with the different sign. So $\Phi[3, j_1] = -\Phi[2, j_1]$ and $\Phi[3, j_2] = -\Phi[2, j_2]$. Then it can be inferred that $j_1 = 2, j_2 = 3$. After the position of these two non-zero coefficients are known, the equation 3.14 can be rewritten as following:

$$\begin{cases} X[2] + X[3] = 15 \\ X[2] - X[3] = -5 \\ -X[2] + X[3] = 5 \end{cases} \tag{3.15}$$

There are two unknowns with more than two equations, so the value of $X[2], X[3]$ can be worked out by solving the system of linear equations.

However, the algorithm described above can not solve the problem if the value of the non-zero coefficients are draw from some other distribution rather than continuous or if the non-zero coefficients are more than two. So a novel algorithm is given in the following section which improve the performance of the algorithm in [5] when the data is binary and, for the first time, gives a method with the possibility to recovery more than two non-zero coefficients in one iteration.

3.3 Improved Sparse Walsh-Hadamard Transform for binary data

We consider the problem of computing the Walsh-Hadamard transform for a signal whose coefficients in the transform domain is sparse and binary of $\{0, +1\}$. We want to recover all the Walsh-Hadamard transform domain coefficients by evaluating a sub-sample version of the time

domain signal. The notation for describing the algorithm is listed in 3.1.

The time domain signal is of length $N = 2^n$ (Notation is given by 3.1). There will be d shift and down-sample stages. In each stage, a down-sample matrix is built as 3.3 with different value of b. The algorithm is designed in an adaptive way in contrast to the earlier work in [5]. To be clear, the algorithm is adaptive because of that the number of shift is not fixed. The WHT of the shifted and down-sampled signal is computed to get an observation vector. These shorter WHT signal can be viewed as bin observation, and each observation is a vector whose length is of the same value as the number of shifts in this stage. The kth vector with p shifts should look like this:

$$v_{\Psi_b k} = [\sum_{j \in N(\Psi_b^T)} \phi_{j,0} X_{\Psi_b k+j}, \sum_{j \in N(\Psi_b^T)} \phi_{j,1} X_{\Psi_b k+j}, ..., \sum_{j \in N(\Psi_b^T)} \phi_{j,p-1} X_{\Psi_b k+j}]^T \tag{3.16}$$

where $\phi_{j,p}$ is the sign pattern $\{-1, +1\}$ corresponding to the pth shift.

Symbol	Meaning
x	Time domain signal
X	Walsh-Hadamard domain signal
$N = 2^n$	length of the time domain signal
$B = 2^b$	length of the down-sampled signal
K	Number of non-zero coefficients in Walsh-Hadamard domain
d	Number of stages
α	$B = N^\alpha$
Ψ	Down-sample matrix

Table 3.1: Parameters involved in describing the algorithm.

One more shift in this stage, one more element will be added at the end of this vector. Clearly, if the first element is zero, there is no need to have another shift in this stage for having more

elements for this bin since all the coefficients fall into this bin are zeros. The value of the first element give the number of non-zero coefficients in this bin. We would like to find more than one non-zero coefficients at once if there is any by choosing appropriate shifts. The number of shifts should be at most 2^{n-b} otherwise the entire time domain signal is used for the recovery.

3.3.1 Choice of shift

When the number of shift is 2^{n-b}, the kth bin vector is nothing but a permutation of the WHT of $[X_{\psi_b k+j}]$ where $j \in N(\psi_b^T)$. We want to find a proper permutation so that as less as possible shifts are needed to distinguish different coefficients. Here we propose a way to choose the shift. Two phases are included in this method.

An example is given to demonstrate the choice of shift. A Hadamard matrix of order 4 is given as follow:

$$\begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix} \tag{3.17}$$

Phase 1 In this phase, different shifts are chosen so that the sign of the aliased coefficients is of different frequencies of $\{+1, -1\}$. In general, if there are $\frac{N}{B}$ coefficients are aliased together, $\log \frac{N}{B} + 1$ shifts are chosen in this phase. The index of these shifts is given by $2^i + 1$ for any possible i (indices start from 1).

If $\frac{N}{B} = 4$, the sign pattern is given by the $1^{st}, 3^{rd}, 2^{nd}$ row of the Hadamard matrix of order 4.

Phase 2 In this phase, shifts are chosen so that the sign pattern of aliased coefficients is given by the unchosen row of the Hadamard matrix. The number of the unchosen row is given by $\frac{N}{B} - (\log \frac{N}{B} + 1)$.

In the example of $\frac{N}{B} = 4$, the 4^{th} row belongs to the second phase.

3.3.2 Bin Classification

The bin observation can be classified into several classes based on the number of non-zero coefficients aliased in that bin. Not like the earlier work, $Singleton$ is more attractive than the other kind of bin observations. $Order$ of a bin observation is defined as the number of non-zero coefficients connected to it. The $Order$ of a bin can be simply tell by evaluating its first element since it is the summation of all the coefficients connected to that bin and the value of each non-zero coefficient is one. A method to determine the distribution of the non-zero coefficients is given when the order of a bin is found. The algorithm will be described in the following section.

3.3.3 Position Identification by Look up table

Since the value of the unknown coefficients are all ones, what has to be find out is the position of them. If there are k of $m = 2^{n-b}$ are non-zero coefficients, the total number of possible distributions is $\binom{m}{k}$. An example of $m = 4, k = 2$ is given as follow:

$$C = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{3.18}$$

where each column gives a possible distribution.

We use the permutation of the Hadamard matrix H_p to generate a look up table. The permutation is given following the order of the choice of the shift:

$$H_p = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix} \tag{3.19}$$

The look up table can be computed as:

$$L = H_p C = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 0 & -2 \\ 0 & 2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 \end{bmatrix} \tag{3.20}$$

Each column in L represent a possible bin observation vector. It should be read from the top to the bottom. With one more shift in each this stage, one more entry of the vector can be obtained. After a new entry is given, look up table is checked to see whether it is enough to distinguish the case of distribution of these non-zero coefficients. In this case of $m = 4, n = 2$, if the first two shifts give the observation vector as $(2, 2)$, the distribution can be identified that there are 2 non-zero coefficients and they are the first two of them. However, we do not want to find every possible cases by using this strategy and, indeed, it is impossible to do so if only a sub-sampled version of time domain signal can be used. With a sub-sampled signal, only a subset of the cases can be identified and the proportion of the subset is depending on how much the sample can be used.

It should be noticed that this method works not limited to the $Order - 2$ cases but any order of bin. However, under the same down-sample rate, the benefit from a higher order recovery can be negligible because of the following reason. Under the same down-sample rate, with the uniformly distributed coefficients in the frequency domain, the possibility is much lower for more non-zero coefficients to fall into the same bin observation.

3.3.4 Peeling Decoder

A peeling decoder algorithm is given after the position identification process. In each iteration, if the position of non-zero coefficients are identified, the contribution of these coefficients are subtracted from the signal. The idea behind the peeling process is that by subtract the decoded coefficients, at least one more bin observation became decodable. So the iteration can keep on working until the entire signal is recovered.

3.3.5 Sample and Computation Complexity

3.3.5.1 Sample Complexity

The down-sample matrix build for each stage looks like this:

$$\Psi_b = [\mathbf{0}_{n\times(n-b)} I_b]^T. \tag{3.21}$$

The length of each shifted down-sampled signal is $2^b = B = N^\alpha$. In the simulation shown in the following section, the number of shifts in each down-sample stage is fixed to be

$O(\log \frac{N}{B}) = O(\log N)$. There will be d such down-sample stages, so the total sample is given as $O(N^\alpha \log N)$.

3.3.5.2 Computation Complexity

The computation complexity is given by adding all the shorter WHT in each branch. The complexity of fast Walsh-Hadamard transform for a signal of length B is $O(B \log B)$. Then number of shorter WHT in each stage is $\log \frac{N}{B}$. There are d stages. So the computation complexity is given as $O(N^\alpha \log^2 N)$.

3.3.6 Simulation

In this section, simulation results are given to show the performance of our algorithm when compared with the earlier work.

3.3.6.1 Random generated binary data

In this section, the test data is generated by generating the Walsh-Hadamard Transform domain coefficients randomly from a Bernoulli distribution. The time domain signal can be obtained by computing its inverse Walsh-Hadamard Transform. It should be noted that this does not reflect the WHT of the graph cut function.

There are $d = 2$ down sample stages and $\log \frac{N}{B} + 1$ shifts in each stages. The number of shifts is fixed for comparison with the work in [5]. The performance is shown in Fig. 3.2. The figure shows the probability of errors versus the sparsity for a fixed number of samples. The length of the

test data is $N = 2^8$ with $b = 4$.

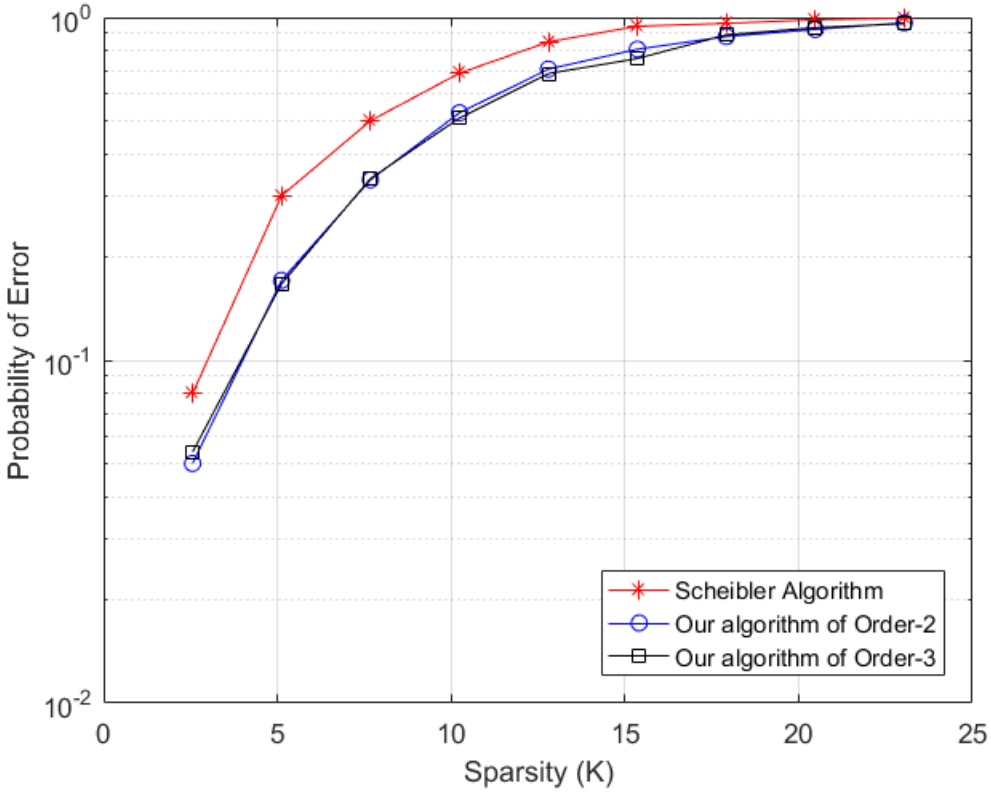

Figure 3.2: Performance of the proposed algorithms on random generated binary data.

With the same sparsity of the test data, the probability of error for the algorithm in[5] is higher for the algorithm we proposed. However, the performance improvement in recovering more than 2 coefficients in each bin is not significant.

3.3.6.2 Random generated graph data

In this section, we consider a randomly generated graph and the associated WHT of the cut function. For a cut function associated with a graph, The non zero coefficients appears in the frequency signal only at the position whose binary representation contains two ones. The zero frequency coefficient is the negative sum of all the other coefficients. When the length of signal is

$N = 2^n$, there are at most $\binom{n}{2}$ non-zero coefficients in the frequency domain except the position of zero. To generate a signal of sparsity K, K positions are randomly chosen among all $\binom{n}{2}$ possible positions.

To ran the algorithm successfully on the graph data. The last $B - 1$ bin observations are used to identify all the non-zero coefficients except the zero-frequency value. The first bin observation vector is evaluated to infer the zero-frequency value when all the other bin observation vectors turn into zero vectors along the peeling process. The simulation result is shown in 3.3.

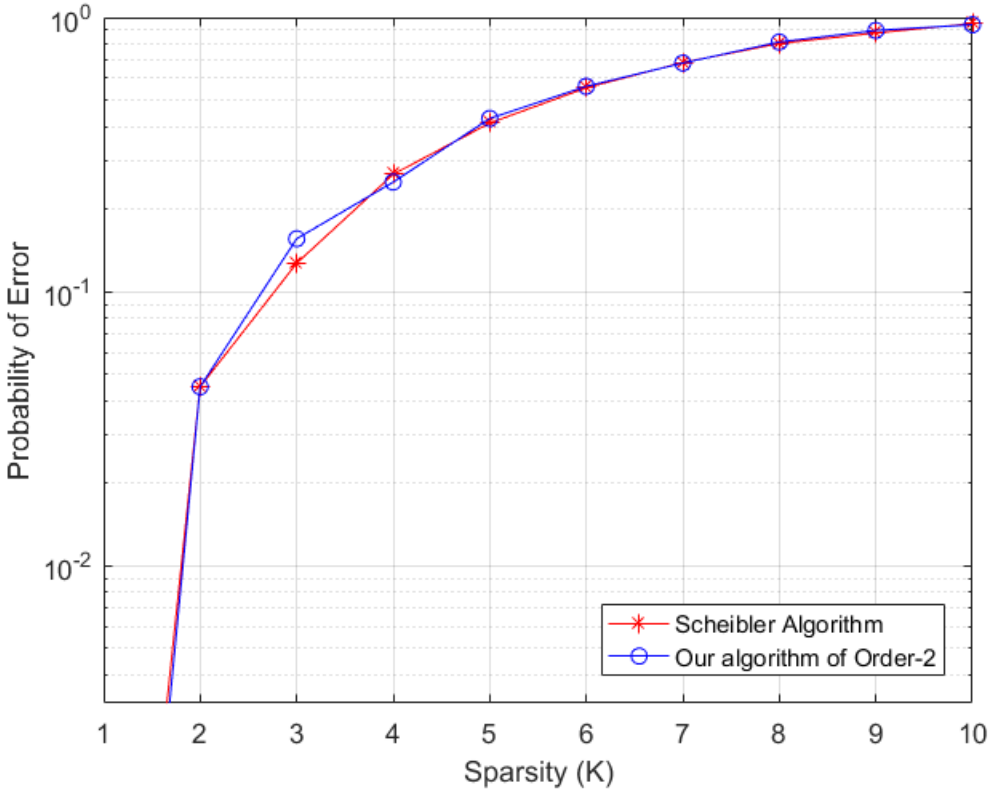

Figure 3.3: Performance of the proposed algorithms on graph data.

The performance of our algorithm and Scheibler's algorithm are tested with all the parameters same as the tests in previous section. However there is no noticeable improvement can be found in this result. More analyses are needed to explain the result.

4. CONCLUSIONS

We have studied two problems related to sparse signal computation: one is 2-D pattern matching and the other is sparse Walsh-Hadamard transform computation for binary frequency data.

For the 2-D pattern matching, we developed a graph code based 2-D sparse recovery algorithm for computing the peaks in the cross-correlation between the pattern and the database. Also, we proposed a possible application of this 2-D pattern matching algorithm for vision-based positioning system. Special patterns can be designed to deliver a desired sparse cross-correlation signal. Simulation results were shown for randomly generated binary images and when the designed patterns are embedded in the real image. For binary data, our algorithm was able to determine around 5000 matching positions in a database of size 1287×1287 with a probability of error that is less than 1%. For the real image, our algorithm can find the position of a special designed pattern in a real image corrupted with Gaussian noise. By using less than 10% of samples, the algorithm can find the position of the pattern with a probability of error less than 5% when the standard deviation of the noise is 50.

For the second problem, we designed an adaptive algorithm to recover a signal with a sparse binary Walsh-Hadamard transform. The signal is shifted by p and down-sampled by a matirx Ψ_b. For binary coefficients, our algorithm gives the possibility to recover more than one non-zero coefficients in one iteration of the peeling decoder algorithm. We tested the performance of the algorithm on randomly generated signals and compared the performance of our algorithm with that of earlier work in [5]. Our algorithm has a lower probability of error given the same amount of samples.

4.1 Further work

4.1.1 2-D Pattern Matching

The pattern designed in 2.8 is very simple and contains no information more than the position. We believe that some part of the pattern can be used to convey information in the way like bar code

or QR code.

4.1.2 Improved sparse Walsh-Hadamard transform computation for binary data

Our algorithm showed better performance than earlier work only in the case of randomly generated signals. Further work is required to design better algorithms for learning graphs. We present a few possible future works.

4.1.2.1 Graph Learning

We consider a graph learning problem as we introduced at the beginning of this chapter. The structure of the graph is to be learned. The structure information can be give by convert a weighted graph to an unweighted one. When the graph is an undirected unweighted graph, the Fourier coefficient of the cut function is given by considering the weight for all edges are -2:

$$X[f] = \begin{cases} -\frac{1}{2}|E|, & if f = (0,0,...,0) \\ 1, & \text{if } f_s = f_t = 1 \text{ and } f_i = 0 \forall i \neq s,t \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

This is a sparse signal indeed and all but one of the Fourier coefficients are of the same value. This data is used for the testing of the algorithm proposed in [8]. We notice that the property of the coefficient value has not been used in their earlier work, and we hope our proposed algorithm can improve the speed of the learning process for undirected unweighted graph by taking the advantage of this property. If the number of the edge can be known at first, then the problem will reduce to learning a function whose non-zero WHT coefficients are all the same. If not, we can still develop a method to learn the zero-frequency coefficient by isolating it.

4.1.2.2 Adaptivity of the proposed algorithm

In our algorithm, the number of shifts used in each bin is fixed to be $\log_2 \frac{N}{B} + 1$. It is possible to obtain improved performance by using more shifts per bin in the algorithm. Studying the performance of our algorithm as a function of the number of shifts remains an open problem.

REFERENCES

[1] F. Ong, S. Pawar, and K. Ramchandran, "Fast sparse 2-d dft computation using sparse-graph alias codes," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4059–4063, 2016.

[2] N. T. Janakiraman, A. Vem, K. R. Narayanan, and J.-F. Chamberland, "Substring/pattern matching in sub-linear time using a sparse fourier transform approach," *ArXiv*, vol. abs/1704.07852, 2017.

[3] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o(k log k) complexity," in *2013 IEEE International Symposium on Information Theory*, pp. 464–468, 2013.

[4] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-optimal average-case sparse fourier transform in two dimensions," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1258–1265, 2013.

[5] R. Scheibler, S. Haghighatshoar, and M. Vetterli, "A fast hadamard transform for signals with sublinear sparsity in the transform domain," *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 2115–2132, 2015.

[6] Z. Qin, J. Fan, Y. Liu, Y. Gao, and G. Y. Li, "Sparse representation for wireless communications: A compressive sensing approach," *IEEE Signal Processing Magazine*, vol. 35, no. 3, pp. 40–58, 2018.

[7] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk, "Faster GPS via the sparse Fourier transform," in *mobicom*, pp. 353–364, ACM, 2012.

[8] A. Amrollahi, A. Zandieh, M. Kapralov, and A. Krause, "Efficiently learning fourier sparse set functions," in *NeurIPS*, 2019.

[9] N. T. Janakiraman, *Applications of Coding Theory to Sub-linear Time Sparse Recovery Problems*. PhD thesis, Texas A&M University, 2019.