

SPARSE IMAGE REPRESENTATION  
VIA COMBINED TRANSFORMS

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF STATISTICS  
AND THE COMMITTEE ON GRADUATE STUDIES  
OF STANFORD UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

Xiaoming Huo  
August 1999

© Copyright by Xiaoming Huo 1999  
All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

---

David L. Donoho  
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

---

Stephen Boyd

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

---

Michael A. Saunders

Approved for the University Committee on Graduate Studies:



*To my mother and father  
and my sister, Xiao-bin*



*To find a sparse image representation.*



# Abstract

We consider sparse image decomposition, in the hope that a sparser decomposition of an image may lead to a more efficient method of image coding or compression.

Recently, many transforms have been proposed. Typically, each of them is good at processing one class of features in an image but not other features. For example, the 2-D wavelet transform is good at processing point singularities and patches in an image but not linear singularities, while a recently proposed method—the edgelet-like transform—is good for linear singularities, but not for points. Intuitively, a combined scheme may lead to a sparser decomposition than a scheme using only a single transform.

Combining several transforms, we get an overcomplete system or dictionary. For a given image, there are infinitely many ways to decompose it. How to find the one with the sparsest coefficients? We follow the idea of Basis Pursuit—finding a minimum  $\ell^1$  norm solution. Some intuitive discussion and theoretical results show that this method is *optimal* in many cases. A big challenge in solving a minimum  $\ell^1$  norm problem is the computational complexity. In many cases, due to the intrinsic nature of the high-dimensionality of images, finding the minimum  $\ell^1$  norm solution is nearly impossible. We take advantage of the recent advances in convex optimization and iterative methods. Our approach is mainly based on two facts: first, we have fast algorithms for each transform; second, we have efficient iterative algorithms to solve for the Newton direction.

The numerical results (to some extent) verify our intuitions, in the sense that: [1] the combined scheme does give sparser representations than a scheme applying only a single transform; [2] each transform in the combined scheme captures the features that this transform is good at processing. (Actually, [2] is an extension of [1].)

With improved efficiency in numerical algorithms, this approach has the promise of producing more compact image coding and compression schemes than existing ones.



# Acknowledgments

I would like to thank my advisor, Professor David L. Donoho, for introducing me to the areas of statistical signal/image processing and optimization, and for providing me with a continuous stream of suggestions, feedbacks and encouragements during the period I worked on this thesis. He is the best scientist I have ever seen, and his penetrating views on many scientific fields have kept surprising me.

I am indebted to Professor Michael Saunders for letting me use and modify his software, and for helping me in programming and technical writing. His expertise in scientific computing is something I have always admired.

I would like to thank Professor Stephen Boyd for giving a tremendous amount of suggestions during the group meetings and for serving on my reading committee. His clarity and style in technical presentation and sharpness in abstract thinking are something that I have always tried to imitate.

I would like to thank Professor Brad Efron and Professor Iain Johnstone for writing recommendation letters on my behalf during my job search. I would also like to thank Professor Jim Dai for helping me create the interview opportunity in Georgia Institute of Technology, which will soon be the next step in my professional career.

I want to thank all the members of the Department of Statistics at Stanford for creating a stimulating and supporting environment. I would like to thank all the members in the Boyd-Donoho-Saunders group, whose presence sometimes is my major source of momentum. I want to thank all my friends at Stanford, who have made another part of my life full of joy and excitement. (I am afraid to list all their names here as we usually do, because I know that no matter how hard I try, the list will always be incomplete.)

Finally, I want to express my deepest gratitude to my parents and my sister for constant and unconditional love and support. Without them, none of my accomplishments would be possible. To them, I dedicate this thesis.



# Contents

<b>Abstract</b>	<b>ix</b>
<b>Acknowledgments</b>	<b>xi</b>
<b>Content</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Figures</b>	<b>xxiii</b>
<b>Nomenclature</b>	<b>xxv</b>
<b>List of Abbreviations</b>	<b>xxvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Outline . . . . .	2
<b>2 Sparsity in Image Coding</b>	<b>3</b>
2.1 Image Coding . . . . .	3
2.1.1 Shannon’s Mathematical Communication System . . . . .	3
2.1.2 Source and Channel Coding . . . . .	7
2.1.3 Transform Coding . . . . .	8
2.2 Sparsity and Compression . . . . .	13
2.2.1 Weak $l^p$ Norm . . . . .	15
2.2.2 Kolmogorov Entropy and Compression . . . . .	19
2.2.3 Summary . . . . .	21

2.3	Discussion . . . . .	21
2.4	Proof . . . . .	22
<b>3</b>	<b>Image Transforms and Image Features</b>	<b>25</b>
3.1	DCT and Homogeneous Components . . . . .	28
3.1.1	Discrete Fourier Transform . . . . .	28
3.1.2	Discrete Cosine Transform . . . . .	38
3.1.3	Discrete Sine Transform . . . . .	43
3.1.4	Homogeneous Components . . . . .	44
3.1.5	2-D DCT . . . . .	52
3.2	Wavelets and Point Singularities . . . . .	52
3.2.1	Multiresolution Analysis . . . . .	54
3.2.2	Filter Banks . . . . .	57
3.2.3	Discrete Algorithm . . . . .	58
3.2.4	Point Singularities . . . . .	60
3.3	Edgelets and Linear Singularities . . . . .	64
3.3.1	Edgelet System . . . . .	64
3.3.2	Edgelet Transform . . . . .	64
3.3.3	Edgelet Features . . . . .	65
3.3.4	Fast Approximate Edgelet Transform . . . . .	65
3.4	Other Transforms . . . . .	66
3.4.1	Transforms for 1-D Signals . . . . .	66
3.4.2	Transforms for 2-D Images . . . . .	73
3.5	Discussion . . . . .	74
3.6	Conclusion . . . . .	76
3.7	Proofs . . . . .	76
<b>4</b>	<b>Combined Image Representation</b>	<b>81</b>
4.1	Why Combined Image Representation? . . . . .	81
4.2	Sparse Decomposition . . . . .	83
4.3	Minimum $\ell^1$ Norm Solution . . . . .	84
4.4	Lagrange Multipliers . . . . .	86
4.5	How to Choose $\rho$ and $\lambda$ . . . . .	88
4.6	Homotopy . . . . .	91

4.7	Newton Direction . . . . .	93
4.8	Comparison with Existing Algorithms . . . . .	93
4.9	Iterative Methods . . . . .	95
4.10	Numerical Issues . . . . .	96
4.11	Discussion . . . . .	96
	4.11.1 Connection With Statistics . . . . .	96
	4.11.2 Non-convex Sparsity Measure . . . . .	97
	4.11.3 Iterative Algorithm for Non-convex Optimization Problems . . . . .	97
4.12	Proofs . . . . .	98
	4.12.1 Proof of Proposition 4.1 . . . . .	98
	4.12.2 Proof of Theorem 4.1 . . . . .	99
	4.12.3 Proof of Theorem 4.2 . . . . .	101
	4.12.4 Proof of Theorem 4.3 . . . . .	102
<b>5</b>	<b>Iterative Methods</b>	<b>103</b>
5.1	Overview . . . . .	103
	5.1.1 Our Minimization Problem . . . . .	103
	5.1.2 Iterative Methods . . . . .	104
	5.1.3 Convergence Rates . . . . .	105
	5.1.4 Preconditioner . . . . .	107
5.2	LSQR . . . . .	109
	5.2.1 What is LSQR? . . . . .	109
	5.2.2 Why LSQR? . . . . .	110
	5.2.3 Algorithm LSQR . . . . .	111
	5.2.4 Discussion . . . . .	111
5.3	MINRES . . . . .	112
	5.3.1 Algorithm MINRES . . . . .	112
	5.3.2 Algorithm PMINRES . . . . .	114
5.4	Discussion . . . . .	116
	5.4.1 Possibility of Complete Cholesky Factorization . . . . .	116
	5.4.2 Sparse Approximate Inverse . . . . .	117

<b>6</b>	<b>Simulations</b>	<b>119</b>
6.1	Dictionary . . . . .	119
6.2	Images . . . . .	120
6.3	Decomposition . . . . .	121
6.4	Decay of Coefficients . . . . .	122
6.5	Comparison with Matching Pursuit . . . . .	125
6.6	Summary of Computational Experiments . . . . .	127
6.7	Software . . . . .	127
6.8	Scale of Efforts . . . . .	128
<b>7</b>	<b>Future Work</b>	<b>129</b>
7.1	Experiments . . . . .	129
7.2	Modifying Edgelet Dictionary . . . . .	131
7.3	Accelerating the Iterative Algorithm . . . . .	131
<b>A</b>	<b>Direct Edgelet Transform</b>	<b>135</b>
A.1	Introduction . . . . .	135
A.2	Examples . . . . .	136
A.2.1	Edge Filter . . . . .	137
A.3	Details . . . . .	138
A.3.1	Definition . . . . .	139
A.3.2	Cardinality . . . . .	141
A.3.3	Ordering . . . . .	142
A.3.4	Computing a Single Edgelet Coefficient . . . . .	144
A.3.5	Adjoint of the Direct Edgelet Transform . . . . .	147
A.3.6	Discussion . . . . .	148
<b>B</b>	<b>Fast Edgelet-like Transform</b>	<b>149</b>
B.1	Transforms for 2-D Continuous Functions . . . . .	150
B.1.1	Fourier Slice Theorem . . . . .	150
B.1.2	Continuous Radon Transform . . . . .	151
B.2	Discrete Algorithm . . . . .	152
B.2.1	Synopsis . . . . .	153
B.2.2	Interpolation: from Discrete to Continuous Image . . . . .	153

B.2.3	X-interpolation: from Cartesian to Polar Coordinate . . . . .	154
B.2.4	Algorithm . . . . .	158
B.3	Adjoint of the Fast Transform . . . . .	159
B.4	Analysis . . . . .	161
B.4.1	Storage and Computational Complexity . . . . .	161
B.4.2	Effective Region . . . . .	162
B.4.3	Ill-conditioning of the Fast X-interpolation Transform . . . . .	162
B.5	Examples . . . . .	163
B.5.1	Basic Elements for the Fast Edgelet-like Transform . . . . .	163
B.5.2	Edgelet-like Transform for Some Artificial Images . . . . .	164
B.5.3	Edgelet-like Transform for Some Real Images . . . . .	165
B.6	Miscellaneous . . . . .	167
B.6.1	Examples of Interpolation Functions . . . . .	167

**Bibliography** **170**



# List of Tables

3.1	Comparison of transforms and the image features that they are good at processing. The third column lists the order of computational complexity for their discrete fast algorithms. . . . .	26
3.2	Definitions of four types of DCTs. . . . .	39
3.3	Number of multiplications and additions for various fast one-D DCT/IDCT algorithms. The number in $\{ \cdot \}$ is the value when $N$ is equal to 8. . . . .	42
3.4	Decomposition of covariance matrix that can be diagonalized by different types of DCT. . . . .	50
6.1	Table of Running Times . . . . .	127
A.1	Percentages of edgelet coefficients being used in the reconstructions. . . . .	137
A.2	Order of edgelets within a square. . . . .	145
B.1	Pairs of transforms and their adjoint . . . . .	160



# List of Figures

2.1	Shannon's schematic diagram of a general communication system. . . . .	5
2.2	Separation of encoding into source and channel coding. . . . .	7
2.3	Structure of a communication system with a transform coder. The signal is $x$ . Symbol $T$ denotes a transform operator. The output coefficients vector (sequence) is $y$ . Symbols $Q$ and $E$ denote a quantizer and an entropy coder, respectively. Symbols $Q^{-1}$ and $E^{-1}$ denote the inverse operators (if they exist) of operations $Q$ and $E$ . Symbol $U$ stands for the reconstruction transform of transform $T$ . Symbol $\hat{x}$ stands for the estimated (/received/recovered) signal. . . . .	9
2.4	Diagram of image transmission using transform coding. The sequence (or vector) $y$ is the coefficient sequence (or vector) and $\tilde{y}$ is the recovered coefficient sequence (or vector). . . . .	14
2.5	Intuition to compare the speeds of decay for two sequences. A sequence whose sorted-amplitudes-curve B is, as we consider, sparser than the sequence whose sorted-amplitudes-curve is A. . . . .	19
3.1	Sampling of four types of DCT at the index domain. For example, both the first index and the second index for the type-I DCT take integral values. . .	40
3.2	Two-dimensional discrete cosine transform basis functions on an $8 \times 8$ block. The upper-left corner corresponds to the low-low frequency, and the lower-right corner corresponds to the high-high frequency. . . . .	53
3.3	Illustration of a filter bank for forward orthonormal wavelet transform. . . .	58

3.4	Illustration of the discrete algorithm for forward orthonormal wavelet transform on a finite-length discrete signal. The upper graph is for cases having 3 layers. The width of each block is proportional to the length of the corresponding subsequence in the discrete signal. The bottom one is a symbolic version for general cases. . . . .	61
3.5	Multiresolution analysis of a point singularity with Haar wavelets. . . . .	62
3.6	Two-dimensional wavelet basis functions. These are $32 \times 32$ images. The upper left one is a tensor product of two scaling functions. The bottom right 2 by 2 images and the (2,2)th image are tensor products of two wavelets. The remaining images are tensor products of a scaling function with a wavelet. 63	63
3.7	Idealized tiling on the time-frequency plane for (a) sampling in time domain (Shannon), (b) Fourier transform, (c) Gabor analysis, (d) orthogonal wavelet transform, (e) chirplet, (f) orthonormal fan basis, (g) cosine packets, and (h) wavelet packets. . . . .	67
4.1	Quasi-sparsity and distortion curve. . . . .	87
4.2	Function $\bar{\rho}$ (as $\rho$ in figures) and its first and second derivatives. $\bar{\rho}$ , $\bar{\rho}'$ and $\bar{\rho}''$ are solid curves. The dashed curve in figure (a) is the absolute value. The dashed curve in figure (b) is the signum function. . . . .	89
4.3	Function $\bar{\rho}$ at a neighborhood of origin with different values of $\gamma$ , $\gamma = 10, 20, 40, 80, 160$ . The dashed line is the absolute value. . . . .	90
6.1	Four test images. . . . .	121
6.2	Decompositions. . . . .	123
6.3	Decaying amplitudes of coefficients. . . . .	124
6.4	A global algorithm vs. Matching Pursuit. . . . .	126
A.1	Edgelet transform of the Chinese character “Huo”: (a) is the original; (d) is the sorted coefficients; (b), (c), (e) and (f) are reconstructions based on the largest 200, 400, 800, 1600 coefficients, respectively. . . . .	138
A.2	Edgelet transform of the sticky image: (a) is the original; (b) is the filtered image; (c) is the sorted coefficients; (d), (e) and (f) are the reconstructions based on the largest 100, 300 and 500 coefficients, respectively. . . . .	139

A.3	Edgelet transform of the wood grain image: (a) is the original; (d) is the sorted coefficients; (b), (c), (e) and (f) are reconstructions based on the largest $1 \times 10^4$ , $2 \times 10^4$ , $4 \times 10^4$ , $8 \times 10^4$ coefficients, respectively. . . . .	140
A.4	Edgelet transform of Lenna image: (a) is the original Lenna image; (b) is the filtered version; (c) is the sorted largest 5,000 coefficients out of 428032. (d), (e) and (f) are the reconstructions based on the largest 1000, 2000 and 4000 coefficients, respectively. . . . .	141
A.5	Vertices at scale $j$ , for a $8 \times 8$ image with $l = 1$ . The arrows shows the trend of ordering. Integers outside are the labels of vertices. . . . .	144
A.6	Weights of pixels for one edgelet coefficient. . . . .	147
B.1	X-interpolation. . . . .	156
B.2	Effective region. . . . .	162
B.3	Basic elements of the fast edgelet-like transform. . . . .	163
B.4	Multiscale fast edgelet-like transform of artificial needle-like images. . . . .	164
B.5	Fast edgelet-like transform of wood grain image. . . . .	165
B.6	Fast edgelet-like transform of wood grain image. . . . .	166
B.7	Sinc function in (a) and its Fourier transform—blocky function in (b). . . . .	167
B.8	Raised cosine function in (b) and its counterpart in time domain in (a). . . . .	168
B.9	Fifty percent tapered window function in (b) and its counterpart in time domain in (a). . . . .	169



# Nomenclature

## Special sets

$\mathbb{N}$ .....	Set of natural numbers $\{1, 2, 3, \dots\}$
$\mathbb{R}$ .....	Set of real numbers
$\mathbb{Z}$ .....	Set of integer numbers $\{\dots, -2, -1, 0, 1, 2, 3, \dots\}$

## Operators

$\mathcal{I}$ .....	Insert operator
$\mathcal{S}$ .....	Downsample operator

## Matrices

$T$ .....	Toeplitz matrix
$H$ .....	Hankel matrix
$C$ .....	Circulant matrix

## Miscellaneous

$f(\cdot)$ .....	Function with domain $\mathbb{R}$
$f[\cdot]$ .....	Function with domain $\mathbb{Z}$
$\hat{X}$ .....	Fourier transform of $X$
$\{\cdot \cdot \cdot\}$ .....	Ordered sequence
$\mathbf{1}$ .....	All one column vector
Pr. ....	Probability



# List of Abbreviations

BCR .....	Block Coordinate Relaxation
BOB .....	Best Orthogonal Basis Method
BOBDN .....	Best Orthogonal Basis De-Noising
BP .....	Basis Pursuit
BPDN .....	Basis Pursuit De-Noising
CP .....	Cosine Packets
DCT .....	Discrete Cosine Transform
DFT .....	Discrete Fourier Transform
DSP .....	Digital Signal Processing
DST .....	Discrete Sine Transform
DWT .....	Discrete Wavelet Transform
EAD .....	Empirical Atomic Decomposition
FFT .....	Fast Fourier Transform
FOC .....	First Order Condition
FT .....	Fourier Transform
GMRF .....	Gaussian Markov Random Field
GRF .....	Gibbs Random Field
HPF .....	High-Pass Filter
HRP .....	High Resolution Pursuit
IDFT .....	Inverse Discrete Fourier Transform
IID .....	Independent and Identically Distributed
JPEG .....	Joint Photographic Experts Group
KLT .....	Karhunen-Loéve Transform

LPF .....	Low-Pass Filter
LS .....	Least Square
LTI .....	Linear and Time Invariant
MAD .....	Mean of the Absolute Deviation
MOF .....	Method of Frames
MOFDN .....	Method-of-Frames De-Noising
MP .....	Matching Pursuit
MPDN .....	Matching Pursuit De-Noising
MPEG .....	Moving Picture Experts Group
MRA .....	Multiresolution Analysis
MRF .....	Markov Random Field
MSE .....	Mean-Square Error
OMP .....	Orthogonal Matching Pursuit
PCA .....	Principal Component Analysis
RF .....	Random Field
RMS .....	Residual Mean Square
SAI .....	Sparse Approximate Inverse
SIRP .....	Sparse Image Representation Pursuit
TFP .....	Time Frequency Plane
TFR .....	Time Frequency Representation
TVDN .....	Total Variation De-Noising
p.d.f. ....	probability density function

# Chapter 1

## Introduction

### 1.1 Overview

Recently, many new methods of image representation have been proposed, including wavelets, cosine packets, brushlets, edgelets, ridgelets, and so on. Typically each of these is good for a specific class of features, but not good for others. We propose a method of combining image representations, more particularly, a method based on the 2-D wavelet transform and the edgelet-like transform. The 2-D wavelet transform is good at capturing point singularities, while the newly proposed edgelet-like transform is good at capturing linear singularities (edges). Both transforms have fast algorithms for digital images.

Wavelets and edgelet-like features (together) form an overcomplete dictionary. To find a sparse representation, we have had success by minimizing the objective function:  $\|y - \Phi x\|_2^2 + \lambda \rho(x)$ , where  $y$  is the image,  $\Phi$  is the matrix whose columns are all the basis vectors in all the image representations,  $x$  is the vector of coefficients and  $\rho(x)$  is a penalty function that is convex,  $l_1$ -like, and separable. Sparsity of representation is achieved by adding the penalty term:  $\lambda \rho(x)$  to balance the goodness of fit measure  $\|y - \Phi x\|$ .

To develop an efficient algorithm to solve this problem, we utilize insights from convex optimization and the LSQR solver from computational linear algebra. These methods combine to give a fast algorithm for our problem. Numerical experiments provide promising results.

## 1.2 Outline

This thesis is organized as follows:

- Chapter 2 explains why sparse decomposition may lead to a more efficient image coding and compression. It contains two parts: the first part gives a brief description of the mathematical framework of a modern communication system; the second part presents some quantitative results to explain (mainly in the asymptotic sense) why sparsity in coefficients may lead to efficiency in image coding.
- Chapter 3 is a survey of existing transforms for images. It serves as a background for this project.
- Chapter 4 explains our approach in finding a sparse decomposition in an overcomplete dictionary. We minimize the objective function that is a sum of the residual sum of squares and a penalty on coefficients. We apply Newton's method to solve this minimization problem. To find the Newton's direction, we use an iterative method—LSQR—to solve a system of linear equations, which happens to be equivalent to a least squares problem. (LSQR solves a least square problem.) With carefully chosen parameters, the solution here is close to the solution of an *exact* Basis Pursuit, which is the minimum  $\ell^1$  norm solution with exact equality constraints.
- Chapter 5 is a survey of iterative methods and explains why we choose LSQR. Some alternative approaches are discussed.
- Chapter 6 presents some numerical simulations. They show that a combined approach does provide a sparser decomposition than the existing approach that uses only one transform.
- Chapter 7 discusses some thoughts on future research.
- Appendix A documents the details about how to implement the exact edgelet transform in a direct way. This algorithm has high complexity. Some examples are given.
- Appendix B documents the details about how to implement an approximate edgelet transform in a fast way. Some examples are given. This transform is the one we used in simulations.

## Chapter 2

# Sparsity in Image Coding

This chapter explains the connection between sparse image decomposition and efficient image coding. The technique we have developed promises to improve the efficiency of transform coding, which is ubiquitous in the world of digital signal and image processing.

We start with an overview of image coding and emphasize the importance of transform coding. Then we review Donoho's work that answers in a deterministic way the question: *why does sparsity lead to good compression?* Finally we review some important principles in image coding.

### 2.1 Image Coding

This section consists of three parts. The first begins with an overview of Shannon's mathematical formulation of communication systems, and then describes the difference between source coding and channel coding. The second part describes a way of measuring the sparsity of a sequence and the connection between sparsity and coding, concluding with an asymptotic result. Finally, we discuss some of the requirements in designing a realistic transform coding scheme.

Since all the work described here can easily be found in the literature, we focus on developing a historic perspective and overview.

#### 2.1.1 Shannon's Mathematical Communication System

Nowadays, it is almost impossible to review the field of theoretical signal processing without mentioning the contributions of Shannon. In his 1948 paper [127], Shannon writes, "The

fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.” With these words, he described a mathematical framework for communication systems that has become the most successful model in communication and information theory.

Figure 2.1 gives Shannon’s illustration of a general communication system. (The figure is reproduced according to the same figure in [127].) There are five components in this system: information source, transmitter, channel, receiver, and destination.

1. An *information source* produces a message, or a sequence of messages, to be communicated to the destination. A message can be a function of one or more continuous or discrete variables and can itself be continuous- or discrete- valued. Some examples of messages are: (a) a sequence of letters in telegraphy; (b) a continuous function of time  $f(t)$  as in radio or telephony; (c) a set of functions  $r(x, y, t), g(x, y, t)$  and  $b(x, y, t)$  having three variables—in color television they are the intensity of red, green and blue as functions of spatial variables  $x$  and  $y$  and time variable  $t$ ; (d) various combinations—for example, television signals have both video and audio signals.
2. The *transmitter*, or *encoder*, transfers the message into a signal compatible with the channel. In old-fashioned telephony, this operation consists of converting sound pressure into a proportional electrical current. In telegraphy, we have an encoding system to transfer a sequence of words to a sequence of dots, dashes and spaces. More complex operations are applied to messages in modern communication systems.
3. The *channel* is the physical medium that conducts the transmitted signal. The mathematical abstraction of the transmission is a perturbation by noise. A typical assumption on the channel is that the noise is additive, but this assumption can be changed.
4. The *receiver*, or *decoder*, attempts to retrieve the message from the received signal. Naturally, the decoding scheme depends on the encoding scheme.
5. The *destination* is the intended recipient of the message.

Shannon’s model is both concrete and flexible. It has been proven efficient for modeling real world communication systems. From this model, some fundamental problems of communication theory are apparent.

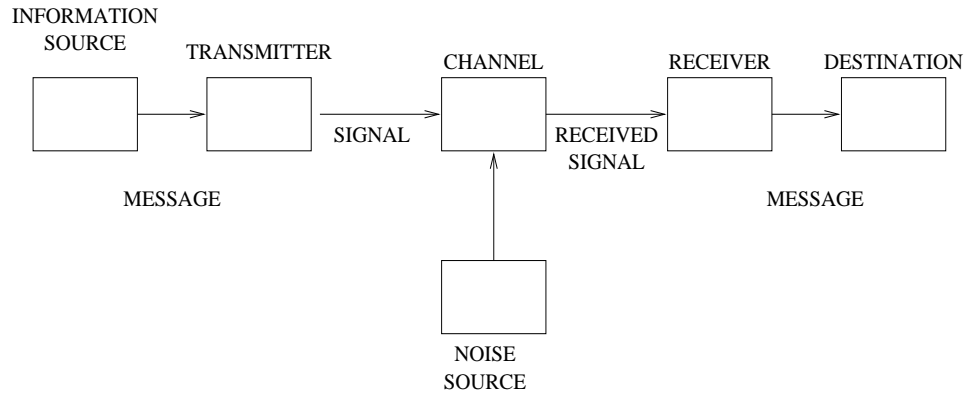


Figure 2.1: Shannon's schematic diagram of a general communication system.

1. Given an information source and the characterization of the channel, is it possible to determine if there exists a transmitter and receiver pair that will transmit the information?
2. Given that the answer to the previous question is *yes*, how efficient can the system (including both transmitter and receiver) be?
3. To achieve the maximum capacity, how should the transmitter and receiver be designed?

To answer these three questions thoroughly would require a review of the entire field of information theory. Here, we try to give a very brief, intuitive and non-mathematically-rigorous description. A good starting point in *information theory* is the textbook [33] by Cover and Thomas.

We need to introduce three concepts: *entropy rate*, *channel capacity* and *distortion*.

The *entropy rate* is a measure of complexity of the information source. Let us consider only a discrete information source. At times  $n \in \mathbb{N}$ , the information source emits a message  $X_n$  from a discrete set  $\mathbb{X}$ ; thus,  $X_n \in \mathbb{X}$ , for all  $n$ . So an infinite sequence of messages  $X = \{X_n, n \in \mathbb{N}\}$  is in  $\mathbb{X}^\infty$ . Suppose each  $X_n$  is drawn independently and identically distributed from  $\mathbb{X}$  with probability density function  $p(x) = \Pr.\{X_n = x\}$ . Then the *entropy rate* of this information source is defined as

$$H = \mathbf{H}(X) = \mathbf{H}(p(\cdot)) = - \sum_{x \in \mathbb{X}} p(x) \log_2 p(x). \quad (2.1)$$

Another intuitive way to view this is that if  $p(x)$  were a uniform distribution over  $\mathbb{X}$ , then the entropy rate  $\mathbf{H}(p(\cdot))$  would be  $\log_2 |\mathbb{X}|$ , where  $|\mathbb{X}|$  is the cardinality (size) of the set  $\mathbb{X}$ . In (2.1) we can view  $1/\log_2 p(x)$  as an analogue to the cardinality such that the entropy is the average logarithm of it.

The *channel capacity*, similarly, is the average of the logarithm of the number of bits (binary numbers from  $\{0, 1\}$ ) that can be reliably distinguished at receiver. Let  $Y$  denote the output of the channel. The channel capacity is defined as

$$C = \max_{p(X)} I(X, Y),$$

where  $I(X, Y)$  is the mutual information between channel input  $X$  and channel output  $Y$ :

$$I(X, Y) = \mathbf{H}(X) + \mathbf{H}(Y) - \mathbf{H}(X, Y).$$

[33] gives good examples in introducing the channel capacity concept.

The *distortion* is the statistical average of a (usually convex) function of the difference between the estimation  $\hat{X}$  at the receiver and the original message  $X$ . Let  $d(\hat{x} - x)$  be the measure of deviation. There are many options for function  $d$ : e.g., (1)  $d(\hat{x} - x) = (\hat{x} - x)^2$ , (2)  $d(\hat{x} - x) = |\hat{x} - x|$ . The distortion  $D$  is simply the statistical mean of  $d(\hat{x} - x)$ :

$$D = \int d(\hat{x} - x)p(x)dx.$$

For (1), the distortion  $D$  is the residual mean square (RMS); for (2), the distortion  $D$  is the mean of the absolute deviation (MAD).

The Fundamental Theorem for a discrete channel with noise [127] states that communication with an arbitrarily small probability of error is possible (from an asymptotic point of view, by coding many messages at once) if  $H < C$  and impossible if  $H > C$ .

A more practical approach is to code with a prescribed maximum distortion  $D$ . This is a topic that is covered by *rate-distortion theory*. The *rate-distortion function* gives the minimum rate needed to approximate a source for a given distortion. A well-known book on this topic is [10] by Berger.

### 2.1.2 Source and Channel Coding

After reviewing Shannon’s abstraction of communication systems, we focus our attention on the transmitter. Usually, the decoding scheme at the receiver is determined by the encoding scheme at the transmitter. There are two components in the transmitter (as shown in Figure 2.2): *source coder* and *channel coder*. For a discrete-time source, without loss of generality, the role of a source coder is to code a message or a sequence of messages into as small a number of bits as possible. The role of a channel coder is to transfer a bit stream into a signal that is compatible with the channel and, at the same time, perform error correcting to achieve an arbitrarily small probability of bit error.

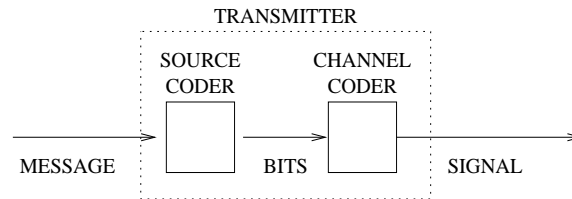


Figure 2.2: Separation of encoding into source and channel coding.

Shannon’s Fundamental Theorem of communication has two parts: the direct part and the converse part. The *direct* part states that if the minimum achievable source coding rate of a given source is strictly below the capacity of a channel, then the source can be reliably transmitted through the channel by appropriate encoding and decoding operations; the *converse* part states that if the source coding rate is strictly greater than the capacity of channel, then a reliable transmission is impossible. Shannon’s theorem implies that reliable transmission can be separated into two operations: source coding and channel coding. The design of source encoding and decoding can be independent of the characteristic of the channel; similarly, the design of channel encoding and decoding can be independent of the characteristics of the source. Owing to the converse theorem, the reliable transmission is doable either through a combination of separated source and channel coding or not possible at all—whether it is a joint source channel coding or not. Note that in Shannon’s proof, the Fundamental Theorem requires the condition that both source and channel are stationary and memoryless. A detailed discussion about whether the Fundamental Theorem holds in more general scenarios is given in [138].

One fact I must point out is that in statistical language, Shannon’s theorem is an

asymptotic result, because it based on an infinite amount of information. In practice, both the source coder that compresses the message to the entropy rate and the channel coder that transmits the bits at almost channel capacity may require intolerable amount of computation and memory. In general, the use of a joint source-channel (JSC) code can lead to the same performance but lower computational and storage requirements. Although separate source and channel coding could be less effective, its appealing property is that a separate source coder and channel coder are easy to design and implement.

From now on, we consider only source coding.

### 2.1.3 Transform Coding

The purpose of source encoding is to transfer a message into a bit stream; and source decoding is essentially an inverse operation. There is no need to overstate the ubiquity of transform coding in modern digital communication systems. For example, JPEG is an industry standard for still image compression and transmission. It implements a linear transform—a two-dimensional discrete cosine transform (2-D DCT). The MPEG standard is an international standard for video compression and transmission that incorporates the JPEG standards for still image compression. The MPEG standard could be the standard for future digital television.

Figure 2.3 gives a general depiction of a modern digital communication system with embedded transform coding. The input message is  $x$ . The operator  $T$  is a transform operator. Most of the time,  $T$  is a linear time invariant (LTI) transform. After the transform, we get coefficients  $y$ . Operator  $Q$  is a quantizer—it transfer continuous variables (in this case, they are coefficients after transform  $T$ ) into discrete values. Without loss of generality, we can assume each message is transferred into a bit stream of finite length. The operator  $E$  is an entropy coder. It further shortens the bit stream by using, for example, run-length coding for pictures [23], Huffman coding, arithmetic coding, or Lempel-Ziv coding, etc. Operators  $T$ ,  $Q$  and  $E$  together make the encoder. We assume that the bit stream is perfectly transmitted through an error-corrected channel. The operators  $E^{-1}$  and  $Q^{-1}$  are inverse operators of the entropy coder  $E$  and the quantizer  $Q$ , respectively. The operator  $U$  is the reconstruction transform of transform  $T$ . If transform  $T$  is invertible, then the transform  $U$  should be the inverse transform of  $T$ , or equivalently,  $U = T^{-1}$ . The final output at the receiver is the estimation, denoted by  $\hat{x}$ , of the original message. Operators  $E^{-1}$ ,  $Q^{-1}$  and  $U$  together form the decoder.

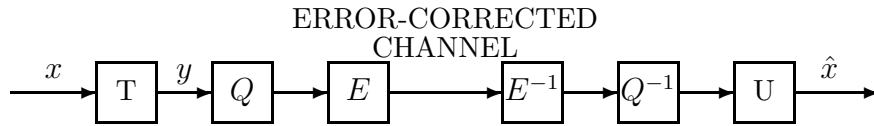


Figure 2.3: Structure of a communication system with a transform coder. The signal is  $x$ . Symbol  $T$  denotes a transform operator. The output coefficients vector (sequence) is  $y$ . Symbols  $Q$  and  $E$  denote a quantizer and an entropy coder, respectively. Symbols  $Q^{-1}$  and  $E^{-1}$  denote the inverse operators (if they exist) of operations  $Q$  and  $E$ . Symbol  $U$  stands for the reconstruction transform of transform  $T$ . Symbol  $\hat{x}$  stands for the estimated (/received/recovered) signal.

The idea of *transform coding* is simply to add a transform at the sender before a quantizer and to add the corresponding reconstruction (or sometimes inverse) transform at the receiver after an inverse of the quantizer. Transform coding improves the overall efficiency of a communication system if the coefficient vector in the transform domain (space made by coefficient vectors) is more sparse than the signal in the signal domain (space made by original signals). In general, the sparser a vector is, the easier it is to compress and quantize. So when the coefficient vector is sparser, it is relatively easy to quantize and compress in the transform domain than in the signal domain. The transform should have a corresponding reconstruction transform, or even be invertible, as we should be able to reconstruct the signal at the receiver. The efficiency of the communication system in Figure 2.3 is determined mostly by the effectiveness of the transform coder.

Now we talk about the history of transform coding in digital communication. The original idea was to use transforms to decorrelate dependent signals; later some researchers successfully explored the possibility of using transforms to improve the efficiency of communication systems. The next paragraph describes this history.

In 1933, in the *Journal of Educational Psychology*, Hotelling [82] first presented a decorrelation method for discrete data. This is the starting point of a popular methodology now called principal component analysis (PCA) in the statistics community. The analogous transform for continuous data was obtained by Karhunen in 1947 and Loève in 1948. The two papers by Karhunen and Loève are not in English, so they are rarely cited. Readers can find them in the reference in [126]. In 1956, Kramer and Mathews [115] introduced a method for transmitting correlated, continuous-time, continuous-amplitude signals. They showed that the total bandwidth (which in theory is equivalent to the capacity of the channel)

necessary to transmit a set of signals with prescribed fidelity can be reduced by transmitting a set of linear combinations of the signals, instead of the original signals. Assuming Gaussian signals and RMS or mean-squared error (MSE) distortion, the Karhunen-Loève transform (KLT) is optimal. Later, Huang and Schultheiss [84, 85] extended this idea to a system that includes a quantizer. Nowadays, PCA has become a ubiquitous technique in multivariate analysis. Its equivalent—KLT—has become a well-known method in the electrical engineering community.

To find a transform that will generate even sparser coefficients than most of the existing transform is the *ultimate* goal of this thesis. Our *key* idea is to combine several state-of-the-art transforms. The combination will introduce redundancy, but at the same time it will provide flexibility and potentially lead to a decomposition that is sparser than one with a single transform.

When describing image coding, we cannot avoid mentioning three topics: quantization, entropy coding and predictive coding. Each of the following subsections is dedicated to one of these topics.

## Quantization

All quantization schemes can be divided into two categories: *scalar quantization* and *vector quantization*. We can think of vector quantization as an extension of scalar quantization in high-dimensional space. Gray and Neuhoff 1998 [76] provides a good review paper about quantization. Two well-written textbooks are Gersho and Gray 1992 [70] and Sayood 1996 [126].

The history of quantization started from Pulse-code modulation (PCM), which was patented in 1938 by Reeves [122]. PCM was the first *digital* technique for conveying an analog signal over an analog channel. Twenty-five years later, a noteworthy article [37] was written by Reeves and Deloraine. It gave a historical review and an appraisal of the future of PCM. Some predictions in this article turned out to be prophetic, especially the one about the booming of digital technology.

Here we give a brief description of scalar quantization. A scalar quantization is a mapping from a real-valued source to a discrete subset; symbolically,

$$\mathbf{Q} : \mathbb{R} \rightarrow \mathbf{C},$$

where  $\mathbf{Q}$  stands for the quantization operator and set  $\mathbf{C}$  is a discrete subset of the real number set  $\mathbb{R}$ :

$$\mathbf{C} = \{y_1, y_2, \dots, y_K\} \subset \mathbb{R}, \quad y_1 < y_2 < \dots < y_K. \quad (2.2)$$

Set  $\mathbf{C}$  is also called a *codebook*.

The procedure of designing a quantizer can be divided into two steps: *partitioning* and *representers selection*. In the partitioning step, the real number set  $\mathbb{R}$  is partitioned into a group of subsets  $c_1, c_2, \dots, c_K$ —Note that this  $K$  is the same  $K$  as in (2.2)—so that these subsets are mutually exclusive ( $c_i \cap c_j = \emptyset, \forall i \neq j$ ) and also comprehensive ( $\cup_{i=1,2,\dots,K} c_i = \mathbb{R}$ ). Each of these  $c_i$ 's are called a *cell*. In the step of choosing representers, each cell  $c_i$  is associated with a real value  $y_i$  (the representer), such that when  $x \in c_i$ , the quantization of  $x$  (denoted by  $q(x)$ ) is equal to  $y_i$ . Equivalently,

$$x \in c_i \implies q(x) = y_i, \quad i = 1, 2, \dots, K.$$

In the measurement of quantization efficiency, two quantities, *distortion* and *rate*, are important. The difference between a quantizer input  $x$  and a quantizer output  $q(x)$ , namely,  $q(x) - x$ , is called the *quantization error*. The *distortion* of a quantizer is defined as a statistical average of a non-negative convex function, say  $d$ , of the quantization error:

$$D(q) = \mathbf{E}[d(q(x) - x)].$$

The *rate*, in the finite number of cells case, is the logarithm of the number of cells. For example, in (2.2) the rate is  $\log(K)$ . In a given system, the rate and distortion are two competing quantities. The larger the rate is, the smaller the minimum distortion could be. And vice versa.

Consider the MSE distortion measure,  $D(q) = \mathbf{E}[(q(x) - x)^2]$ . For each step in quantization, there is a necessary condition for the optimality of the design. So we have in total two conditions:

- *Nearest neighbor classification*: This is a rule for partitioning. Consider the codebook to be fixed. In encoding a source sample  $x$ , one should choose the codebook element

closest to  $x$ . This is written as

$$q(x) = \underset{y_i \in \mathbf{C}}{\operatorname{argmin}} |x - y_i|.$$

Each cell  $c_i$  ( $c_i = \{x : q(x) = y_i\}$ , where the quantization output is  $y_i$ ) is called a *Voronoi cell*.

- *Centroid condition:* When the partitioning  $\mathbb{R} = \cup_i c_i$  is fixed, the MSE distortion is minimized by choosing a representer of a cell as the statistical mean of a variable restricted within the cell:

$$y_i = \mathbf{E}[x|x \in c_i], \quad i = 1, 2, \dots, K.$$

Given a source with a known statistical probability density function (p.d.f.), designing a quantizer that satisfies both of the optimality conditions is not a trivial task. In general, it is done via an iterative scheme. A quantizer designed in this manner is called a *Lloyd-Max quantizer* [94, 104].

A uniform quantizer simply takes each cell  $c_i$  as an interval of equal length; for example, for fixed  $\Delta$ ,  $c_i$  is  $(i\Delta - \frac{\Delta}{2}, i\Delta + \frac{\Delta}{2}]$ . (Note that the number of cells here is not finite.) The uniform quantizer is easy to implement and design, but in general not optimal.

A detailed discussion about quantization theory is easy to find and beyond the scope of this thesis. We skip it.

## Entropy Coding

*Entropy coding* means compressing a sequence of discrete values from a finite set into a shorter sequence combined from elements of the same set. Entropy coding is a *lossless* coding, which means that from the compressed sequence one can perfectly reconstruct the original sequence.

There are two important approaches in entropy coding. One is a statistical approach. This category includes Huffman coding and arithmetic coding. The other is a dictionary based approach. The Lempel-Ziv coding belongs to this category. For detailed descriptions of these coding schemes we refer to some textbooks, notably Cover and Thomas 1991 [33], Gersho and Gray 1992 [70] and Sayood 1996 [126].

### Predictive Coding

*Predictive coding* is an alternative to transform coding. It is another popular technique being used in source coding and quantization. The idea is to use a feedback loop to reduce the redundancy (or correlation) in the signal. A review of the history and the recent developments of the predictive coding in quantization can be found in [77].

### Summary and Problem

For an image transmission system applying transform coding, the general scheme is depicted in Figure 2.4. First of all, a transform is applied to the image, and a coefficient vector  $y$  is obtained. Generally speaking, the sparser the vector  $y$  is, the easier it can be compressed. The coefficient vector  $y$  is quantized, compressed, and transmitted to the other end of the communication system. At the receiver, the observed vector  $\tilde{y}$  should be a very good approximation to the original coefficient vector  $y$ . An inverse transform is applied to recover the image.

We have not answered the question on how to quantify the sparsity of a vector and why the sparsity leads to a “good” compression. In the next section, to answer these questions, we introduce the work of Donoho that gives a quantification of sparsity and its implication in effective bit-level compression.

## 2.2 Sparsity and Compression

In the previous section, we imply a slogan: *sparsity leads to efficient coding*. In this section, we first quantify the measure of sparsity, then explain an implication of sparsity to efficiency of coding. Here *coding* means a cascade of a quantizer and a bit-level compressor. In the transform coding, the following question is answered to some extent:

*“What should the output of a transform be to simplify the subsequential quantization and bit-level compression?”*

There are two important approaches to quantify the difficulty (or the limitation) of coding. One is a statistical approach, which is sometimes called an entropy approach. The other is a deterministic approach introduced mainly by Donoho in [46] and [47]. We describe both of them.

First, the statistical approach. If we know the probability density function, we can calculate the entropy defined in (2.1). In fact, the entropy is a lower bound on the expected

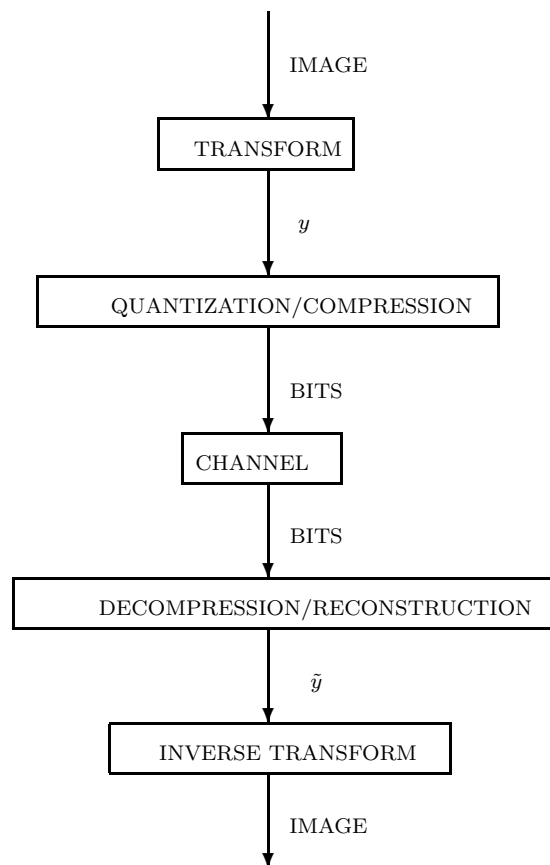


Figure 2.4: Diagram of image transmission using transform coding. The sequence (or vector)  $y$  is the coefficient sequence (or vector) and  $\tilde{y}$  is the recovered coefficient sequence (or vector).

average length of a bit string that is necessary to code a signal in a quantization-compression scheme. A variety of schemes have been developed in the literature. In different circumstances, some schemes approach this entropy lower bound. The field that includes this research is called *vector quantization*.

Furthermore, if the coefficients after a certain transform are independent and identically distributed (IID), then we can deploy the same scalar quantizer for every coordinate. The scalar quantizer can nearly achieve the informatic lower bound (the lower bound specified in information theory). For example, one can use the Lloyd-Max quantizer.

The statistical (or entropic) approach is elegant, but a practical disadvantage is that in image coding we generally do not know the statistical distribution of an image. Images are from an extremely high-dimensional space. Suppose a digital image has  $256 \times 256$  pixels and each pixel is assigned a real value. The dimensionality of the image is then 65,536. Viewing every image as a sample from a high-dimensional space, the number of samples is far less than the dimensionality of the space. Think about how many images we can find on the internet. The size of image databases is usually not beyond three digits; for example, the USC image database and the database at the Center for Imaging Science Sensor. It is hard to infer the probability distribution in such a high-dimensional space from such a small sample. This motivates us to find a non-stochastic approach, which is also an empirical approach. By “empirical” we mean that the method depends only on observations.

The key in the non-stochastic approach is to measure the empirical sparsity. We focus on the ideas that have been developed in [46, 47]. The measure is called the *weak  $l^p$  norm*. The following two subsections answer the following two questions:

1. What is the weak  $l^p$  norm and what are the scientific reasons to invent such a norm? (Answered in Section 2.2.1.)
2. How is the weak  $l^p$  norm related to asymptotic efficiency of coding? (Answered in Section 2.2.2.)

### 2.2.1 Weak $l^p$ Norm

We start with the definition of the weak  $l^p$  norm, for  $1 \leq p \leq 2$ , then describe its connection with *compression number*, measure of *numerosity* and *rate of recovery*, and finally we introduce the concept of *critical index*, which plays an important role in the next section (Section 2.2.2).

First, the formulation. Suppose  $\theta = \{\theta_i : i \in \mathbb{N}\}$  is an infinite-length real-valued sequence:  $\theta \in \mathbb{R}^\infty$ . Further assume that the sequence  $\theta$  can be sorted by absolute values. Suppose the sorted sequence is  $\{|\theta|_{(i)}, i \in \mathbb{N}\}$ , where  $|\theta|_{(i)}$  is the  $i$ -th largest absolute value. The weak  $l^p$  norm of the sequence  $\theta$  is defined as

$$|\theta|_{wl^p} = \sup_i i^{1/p} |\theta|_{(i)}. \quad (2.3)$$

Note that this is a *quasi-norm*. (A norm satisfies a triangular inequality,  $\|x+y\| \leq \|x\| + \|y\|$ ; a quasi-norm satisfies a *quasi-triangular inequality*,  $\|x+y\| \leq K(\|x\| + \|y\|)$  for some  $K > 1$ .) The weak  $l^p$  norm has a close connection with three other measures that are related to vector sparsity.

### Numerosity

A straightforward way to measure the sparsity of a vector is via its *numerosity*: the number of elements whose amplitudes are above a given threshold  $\delta$ . In a more mathematical language, for a fixed real value  $\delta$ , the numerosity is equal to  $\#\{i : |\theta_i| > \delta\}$ . The following lemma is cited from [47].

**Lemma 2.1** *For any sequence  $\theta$ , the following inequality is true:*

$$\#\{i : |\theta_i| > \delta\} \leq |\theta|_{wl^p}^p \delta^{-p}, \quad \delta > 0.$$

From the above lemma, a small weak  $l^p$  norm leads to a small number of elements that are significantly above zero. Since numerosity, which basically counts significantly large elements, is an obvious way to measure the sparsity of a vector, the weak  $l^p$  norm is a measure of sparsity.

### Compression Number

Another way to measure sparsity is to use the *compression number*, defined as

$$c(n) = \left( \sum_{i=n+1}^{\infty} |\theta|_{(i)}^2 \right)^{1/2}.$$

Again, the compression number is based on the sorted amplitudes. In an orthogonal basis, we have isometry. If we perform a thresholding scheme by keeping the coefficients associated

with the largest  $n$  amplitudes, then the compression number  $c(n)$  is the square root of the RSS distortion of the signal reconstructed by the coefficients with the  $n$  largest amplitudes. The following result can be found in [47].

**Lemma 2.2** *For any sequence  $\theta$ , if  $m = 1/p - 1/2$ , the following inequality is true:*

$$c(N) \leq \alpha_p N^{-m} |\theta|_{wlp}, \quad N \geq 1,$$

where  $\alpha_p$  is a constant determined only by the value of  $p$ .

From the above lemma, a small weak  $l^p$  norm implies a small compression number.

### Rate of Recovery

The *rate of recovery* comes from statistics, particularly in density estimation. For a sequence  $\theta$ , the rate of recovery is defined as

$$r(\epsilon) = \sum_{i=1}^{\infty} \min\{\theta_i^2, \epsilon^2\}.$$

**Lemma 2.3** *For any sequence  $\theta$ , if  $r = 1 - p/2$ , the following inequality is true:*

$$r(\epsilon) \leq \alpha'_p |\theta|_{wlp}^p (\epsilon^2)^r, \quad \epsilon > 0,$$

where  $\alpha'_p$  is a constant.

This implies that a small weak  $l^p$  norm leads to a small rate of recovery. In some cases (for example, in density estimation) we choose rate of recovery as a measure of sparsity. The weak  $l^p$  norm is therefore a good measure of sparsity too.

Lemma 1 in [46] shows that all these measures are equivalent in an asymptotic sense.

### Critical Index

In order to define the *critical index* of a functional space, we need to introduce some new notation. A detailed discussion of this can be found in [47]. Suppose  $\Theta$  is the functional space that we are considering. (In the transform coding scenario, the functional space  $\Theta$  includes all the coefficient vectors.) An infinite-length sequence  $\theta = \{\theta_i : i \in \mathbb{N}\}$  is in a weak

$l^p$  space if and only if  $\theta$  has a finite weak  $l^p$  norm. For fixed  $p$ , the statement “ $\Theta \subset wl^p$ ” simply means that every sequence in  $\Theta$  has a finite weak  $l^p$  norm. The critical index of a functional space  $\Theta$ , denoted by  $p^*(\Theta)$ , is defined as the infimum of  $p$  such that the weak  $l^p$  space includes  $\Theta$ :

$$p^*(\Theta) = \inf\{p : \Theta \subset wl^p\}.$$

In Section 2.2.2, we describe the linkage between critical index and *optimal exponent*, where the latter is a measure of efficiency of “optimal” coding in a certain functional space.

### A Result in the Finite Case

Before we move into the asymptotic discussion, the following result provides insight into coding a finite-length sequence. The key idea is that the faster the finite sequence decays, the fewer bits required to code the sequence. Here the sparsity is measured by the decay of the sorted amplitudes, which in the 1-D case is the decay of the sorted absolute values of a finite-length sequence.

In general, it is hard to define a “universal” measure of speed of decay for a finite-length sequence. We consider a special case. The idea is depicted in Figure 2.5. Suppose that two curves of sorted amplitudes of two sequences (indicated by A and B in the figure) have only one intersection. We consider the curve that is lower at the tail part (indicated by B in the figure) corresponds to a sparser sequence. (Obviously, this is a simplified version because there could be more than one intersection.) We prove that by using the most straightforward coding scheme, which is to take an identical uniform scalar quantizer at every coordinate, we need fewer bits to code B than to code A.

To be more specific, without loss of generality, suppose we have two normalized non-increasing sequences. Both of these sequences have  $L^2$  norm equal to 1. One decays “faster” than the other, in the sense that when after a certain index, the former sorted-amplitude sequence is always below the latter one, as in Figure 2.5. (Curve A is always above curve B after the intersection.) Suppose we deploy a uniform scalar quantizer with the same quantization parameter  $q$  on every coordinate. Then the number of bits required to code the  $i$ th element  $\theta_i$  in the sequence  $\theta$  is no more than  $\log_2[\theta_i/q] + 1$  (where  $[x]$  is the closest integral value to  $x$ ). Hence the total number of bits to code vector  $\theta = \{\theta_i : 1 \leq i \leq N\}$  is upper bounded by  $\sum_{i=1}^N \log_2[\theta_i/q] + N$ . Since  $q$  and  $N$  are constants, we only need to

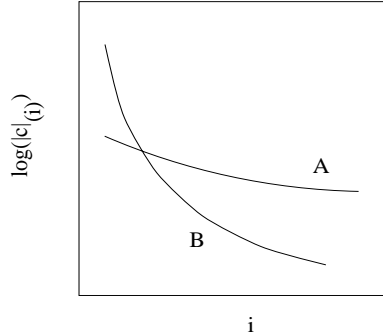


Figure 2.5: Intuition to compare the speeds of decay for two sequences. A sequence whose sorted-amplitudes-curve B is, as we consider, sparser than the sequence whose sorted-amplitudes-curve is A.

consider  $\sum_{i=1}^N \log |\theta_i|$ . The following theorem shows that a “sparser” sequence needs fewer bits to code.

First suppose  $\{x_i : 1 \leq i \leq N\}$  and  $\{y_i : 1 \leq i \leq N\}$  are two non-increasing positive real-valued sequences with a fixed  $l^2$  norm:

$$\begin{aligned} x_1 \geq x_2 \geq \dots \geq x_N > 0, \quad \sum_{i=1}^N x_i^2 = C; \\ y_1 \geq y_2 \geq \dots \geq y_N > 0, \quad \sum_{i=1}^N y_i^2 = C; \end{aligned}$$

where  $C$  is a constant. When  $\exists k, \forall j \geq k, x_j \geq y_j$ , and  $\forall j < k, x_j < y_j$ , we say that “sequence  $\{x_i : 1 \leq i \leq N\}$  majorizes sequence  $\{y_i : 1 \leq i \leq N\}$ ”, or simply say “ $x$  majorizes  $y$ ”.

**Theorem 2.1** *If sequence  $x$  majorizes sequence  $y$  and both of them have the same  $l^2$  norm, then  $\sum_{i=1}^N \log x_i \geq \sum_{i=1}^N \log y_i$ .*

A proof is contained in Section 2.4.

### 2.2.2 Kolmogorov Entropy and Compression

In the previous section, we described the weak  $l^p$  norm and critical index. It is interesting to note that the critical index is closely related to the asymptotics of compression in a functional space. To describe this relationship, we need to introduce a new concept: *optimal*

*exponent*. We first introduce the definition of optimal exponent, then cite the result that states the connection between critical index and optimal exponent.

### Optimal Exponent

We consider a functional space  $\Theta$  made by infinite-length real-valued sequences. An  $\epsilon$ -ball in  $\Theta$ , which is centered at  $\theta_0$ , is (by definition)  $B_\epsilon(\theta_0) = \{\theta : \|\theta - \theta_0\|_2 \leq \epsilon, \theta \in \Theta\}$ , where  $\|\cdot\|_2$  is the  $l^2$  norm in  $\Theta$  and  $\theta_0 \in \Theta$ . An  $\epsilon$ -net of  $\Theta$  is a subset of  $\Theta$   $\{f_i : i \in \mathbf{I}, f_i \in \Theta\}$  that satisfies  $\Theta \subset \bigcup_{i \in \mathbf{I}} B_\epsilon(f_i)$ , where  $\mathbf{I}$  is a given set of indices. Let  $N(\epsilon, \Theta)$  denote the minimum possible cardinality of the  $\epsilon$ -net  $\{f_i\}$ . The (Kolmogorov-Tikhomirov)  $\epsilon$ -entropy of  $\Theta$  is (by definition)

$$H_\epsilon(\Theta) = \log_2 N(\epsilon, \Theta).$$

The  $\epsilon$ -entropy of  $\Theta$ ,  $H_\epsilon(\Theta)$ , is to within one bit the minimum number of bits required to code in space  $\Theta$  with distortion less than  $\epsilon$ . If we apply the nearest-neighbor coding, the total number of possible cases needed to record is  $N(\epsilon, \Theta)$ , which should take no more than  $\lceil \log_2 N(\epsilon, \Theta) \rceil$  bits. (The value  $\lceil x \rceil$  is the smallest integer that is no smaller than  $x$ .) Obviously the  $\epsilon$ -entropy of  $\Theta$ , which is denoted by  $H_\epsilon(\Theta)$ , is within 1 bit of the total number of bits required to code in functional space  $\Theta$ .

It is not hard to observe that when  $\epsilon \rightarrow 0$ ,  $H_\epsilon(\Theta) \rightarrow +\infty$ . Now the question is how fast the  $\epsilon$ -entropy  $H_\epsilon(\Theta)$  increases when  $\epsilon$  goes to zero. The *optimal exponent* defined in [47] is a measure of the speed of increment:

$$\alpha^*(\Theta) = \sup\{\alpha : H_\epsilon(\Theta) = O(\epsilon^{-1/\alpha}), \epsilon \rightarrow 0\}.$$

We refer readers to the original paper [47] for a more detailed discussion.

### Asymptotic Result

The key idea is that there is a direct link between the critical index  $p^*$  and the optimal exponent  $\alpha^*$ . The following is cited from [47], Theorem 2.

**Theorem 2.2** *Let  $\Theta$  be a bounded subset of  $l^2$  that is solid, orthosymmetric, and minimally*

*tail compact. Then*

$$\alpha^* = 1/p^* - 1/2. \quad (2.4)$$

*Moreover, coder-decoder pairs achieving the optimal exponent of code length can be derived from simple uniform quantization of the coefficients  $(\theta_i)$ , followed by simple run-length coding.*

A proof is given in [47].

### 2.2.3 Summary

From above analysis, we can draw the following conclusions:

1. Critical index measures the sparsity of a sequence space, which usually is a subspace of  $l^2$ . In an asymptotic sense, the smaller the critical index is, the faster the sequence decays; and, hence, the sparser the sequence is.
2. Optimal exponent measures the efficiency of the best possible coder in a sequence space. The larger the optimal exponent is, the fewer the bits required for coding in this sequence space.
3. The optimal exponent and the critical index have an equality relationship that is described in (2.4). When the critical index is small, the optimal exponent is large. Together with the previous two results, we can draw the main conclusion: the sparser the sequences are in the sequence space, the fewer the bits required to code in the same space.

## 2.3 Discussion

Based on the previous discussion, in an asymptotic sense, instead of considering how many bits are required in coding, it is equivalent to study the sparsity of the coefficient sequences. From now on, for simplicity, we only consider the sparsity of coefficients. Note that the sparsity of coefficients can be empirically measured by the decay of sorted amplitudes. For infinite sequences, the sparsity can be measured by the weak  $l^p$  norm.

We required the decoding scheme to be computationally cheap, while the encoding scheme can be computationally expensive. The reason is that in practice (for example, in

the broadcasting business), the sender (TV station) can usually afford expensive equipment and a long processing time, while the receiver (single television set) must have cheap and fast (even real-time) algorithms.

Our objective is to find sparse decompositions. Following the above rule, in our algorithm we will tolerate high complexity in decomposing, but superposition must be fast and cheap. Ideally, the order of complexity of superpositioning must be no higher than the order of complexity of a Fast Fourier Transform (FFT). We choose FFT for comparison because it is a well-known technique and a milestone in the development of signal processing. Also by ignoring the logarithm factor, the order of complexity of FFT is almost equal to the order of complexity of copying a signal or image from one disk to another. In general, we can hardly imagine any processing scheme that can have a lower order of complexity than just copying. We will see that the order of complexity of our superposition algorithm is indeed no higher than the order of complexity of doing an FFT.

## 2.4 Proof

### Proof of Theorem 2.1

We only need to prove that

$$\sum_{i=1}^N \log x_i^2 \geq \sum_{i=1}^N \log y_i^2. \quad (2.5)$$

Let's define the following function for a sequence  $x$ :

$$f(x) = \sum_{i=1}^N \log x_i^2.$$

Consider the following procedure:

1. The sequence  $x^{(0)}$  is the same sequence as  $x$ :  $x^{(0)} = x$ .
2. For any integer  $n \geq 1$ , we do the following:
  - (a) Pick an index  $l$  such that

$$l = \min\{i : y_i > x_i^{(n-1)}\}.$$

The index  $l$  does not exist if and only if the two sequences  $\{x_i, 1 \leq i \leq N\}$  and  $\{y_i, 1 \leq i \leq N\}$  are the same.

(b) Pick an index  $u$  such that

$$u = \max\{i : y_i < x_i^{(n-1)}\}.$$

Again, the index  $u$  does not exist if and only if the two sequences  $\{x_i, 1 \leq i \leq N\}$  and  $\{y_i, 1 \leq i \leq N\}$  are the same.

(c) Take

$$\varepsilon = \min\{y_l^2 - (x_l^{(n-1)})^2, (x_u^{(n-1)})^2 - y_u^2\}.$$

(d) Make a new sequence according to the following rules:

$$\begin{aligned} x_l^{(n)} &\leftarrow \sqrt{(x_l^{(n-1)})^2 + \varepsilon}; \\ x_u^{(n)} &\leftarrow \sqrt{(x_u^{(n-1)})^2 - \varepsilon}; \\ x_k^{(n)} &\leftarrow x_k^{(n-1)}, \text{ where } k \neq l \text{ and } k \neq u. \end{aligned}$$

(e) Go back to step (a) unless there is no change in (d).

It should be easy to prove the following facts:

1. Each sequence  $x^{(n)}$ ,  $n \geq 0$ , will be a non-increasing positive real-valued sequence satisfying all the conditions imposed on sequence  $x$  in the Theorem 2.1.
2. The difference  $u - l$  has an integral value bounded between 1 and  $N - 1$ . At each step, the difference  $u - l$  should reduce by at least one. Based on this, the above procedure can repeat at most  $N - 1$  times.
3. It is clear that  $f(x^{(n-1)}) > f(x^{(n)})$ ,  $n \geq 1$ . And because  $f(x) = f(x^{(0)})$  and  $f(y)$  is equal to the last function value in  $f(x^{(n)})$ ,  $n \geq 1$ , we have  $f(x) \geq f(y)$ , which is the inequality (2.5).

The last item concludes our proof for Theorem 2.1.



## Chapter 3

# Image Transforms and Image Features

This chapter is a summary of early achievements and recent advances in the design of transforms. It also describes features that these transforms are good at processing.

Some mathematical slogans are given, together with their quantitative explanations. As many of these topics are too large to be covered in a single chapter, we give pointers in the literature. The purpose of this chapter is to build a concrete foundation for the remainder of this thesis.

One particular message that we want to deliver is: *“Each transform is good for one particular phenomenon, but not good for some others; at the same time, a typical image is made by a variety of phenomena. It is natural to think about how to combine different transforms, so that the combined method will have advantages from each of these transforms.”* This belief is the main motivation for this thesis.

We give emphasis to discrete algorithms and, moreover, fast linear algebra, because the possibility of efficient implementation is one of our most important objectives.

Note that there are two important ideas in signal transform methods: *decomposition* and *distribution*. The idea of decomposition, also called *atomic decomposition*, is to write the image/signal as a superposition (which is equivalently a linear combination) of pre-selected atoms. These atoms make a dictionary, and the dictionary has some special property; for example, orthonormality when a dictionary is made by a single orthonormal basis, or tightness when a dictionary is made by a single tight frame. The second idea is to map the signal into a distribution space. An example is the Wigner-Ville transform, which maps a

1-D signal into a distribution on the time-frequency plane (which is 2-D). Since this thesis mainly emphasizes decomposition, from now on “transform” means only the one in the sense of decomposition.

Some statements could be seemingly subjective. Given the same fact (in our case, for example, it could be a picture), different viewers may draw different conclusions. We manage to be consistent with the majority’s point of view. (Or at least we try to.)

### Key message

Before we move into the detailed discussion, we would like to summarize the key results. Readers will see that they play an important role in the following chapters. Table 3.1 summarizes the correspondence between three transforms, three image features, and orders of complexity of their fast algorithms. Note that the 2-D DCT is good for an image with homogeneous components. For an  $N$  by  $N$  image, its fast algorithm has  $O(N^2 \log N)$  order of complexity. The two-dimensional wavelet transform is good for images with point singularities. For an  $N$  by  $N$  image, its fast algorithm has  $O(N^2)$  order of complexity. The edgelet transform is good for an image with line singularities. For an  $N$  by  $N$  image, its order of complexity is  $O(N^2 \log N)$ , which is the same as the order of complexity for the 2-D DCT.

Transform	Image Feature	Complexity of Discrete Algorithms (for $N \times N$ image)
DCT	Homogeneous Components	$N^2 \log N$
Wavelet Transform	Point Singularities	$N^2$
Edgelet Transform	Line Singularities	$N^2 \log N$

Table 3.1: Comparison of transforms and the image features that they are good at processing. The third column lists the order of computational complexity for their discrete fast algorithms.

### Why so many figures?

In the main body of this chapter, we show many figures. The most important reason is that this project is an image processing project, and showing figures is the most intuitive way to illustrate points.

Some of the figures show the basis functions for various transforms. The reason for illustrating these basis functions is the following: the  $i$ th coefficient of a linear transform is the inner product of the  $i$ th basis function and the image:

$$\text{coefficient}_i = \langle \text{basis function}_i, \text{image} \rangle,$$

where  $\langle \cdot, \cdot \rangle$  is the inner product of two functions. Hence the pattern of basis functions determines the characteristics of the linear transform.

In functional analysis, a basis function is called a *Riesz representer* of the linear transform.

## Notations

We follow some conventions in signal processing and statistics. A function  $X(t)$  is a continuous function with a continuous time variable  $t$ . A function  $X[k]$  is a continuous function with variable  $k$  that only takes integral values. Function  $\hat{X}$  is the Fourier transform of  $X$ . We use  $\omega$  to denote a continuous frequency variable.

More specifics of notation can be found in the main body of this chapter.

## Organization

The rest of this chapter is organized as follows. The first three sections describe the three most dominant transforms that are used in this thesis: Section 3.1 is about the discrete cosine transform (DCT); Section 3.2 is about the wavelet transform; and Section 3.3 is about the edgelet transform. Section 3.4 is an attempt at a survey of other activities.<sup>1</sup> Section 3.5 contains some discussion. Section 3.6 gives conclusions. Section 3.7 contains some proofs.

---

<sup>1</sup>It is interesting to note that there are many other activities in this field. Some examples are the Gabor transform, wavelet packets, cosine packets, brushlets, ridgelets, wedgelets, chirplets, and so on. Unfortunately, owing to space constraints, we can hardly provide many details. We try to maintain most of the description at an introductory level, unless we believe our detailed description either gives a new and useful perspective or is essential for some later discussions.

### 3.1 DCT and Homogeneous Components

We start with the Fourier transform (FT). The reason is that the discrete cosine transform (DCT) is actually a real version of the discrete Fourier transform (DFT). We answer three key questions about the DFT:

1. (Definition) What is the discrete Fourier transform?
2. (Motivation) Why do we need the discrete Fourier transform?
3. (Fast algorithms) How can it be computed quickly?

These three items comprise the content of Section 3.1.1. Section 3.1.2 switches to the discrete cosine transform (DCT). It has two ingredients:

1. definition,
2. fast algorithms.

For fast algorithms, the theory follows two lines: (1) An easy way to implement fast DCT is to utilize *fast DFT*; the fast DFT is also called the fast Fourier transform (FFT). (2) We can usually do better by considering the sparse matrix factorization of the DCT matrix itself. (This gives a second method for finding fast DCT algorithms.) The algorithms from the second idea are computationally faster than those from the first idea. We summarize recent advances for fast DCT in Section 3.1.2. Section 3.1.3 gives the definitions of Discrete Sine Transform (DST). Section 3.1.4 provides a framework for homogeneous images, and explains when a transform is good at processing homogeneous images. The key idea is that the DCT almost diagonalizes the covariance matrix of certain Gaussian Markov random fields. Section 3.1.4 shows that the 2-D DCT is a good transform for images with homogeneous components.

#### 3.1.1 Discrete Fourier Transform

##### Definition

In order to introduce the DFT, we need to first introduce the continuous Fourier transform. From the book by Y. Meyer [107, Page 14], we learn that the idea of continuous Fourier transform dates to 1807, the year Joseph Fourier asserted that any  $2\pi$ -periodic function in  $L^2$ , which is a functional space made by functions whose square integral is finite, is

a superposition of sinusoid functions. Nowadays, Fourier analysis has become the most powerful tool in signal processing. Every researcher in science and engineering should know it.

The Fourier transform of a continuous function  $X(t)$ , by definition, is

$$\hat{X}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(t)e^{-i2\pi\omega t} dt,$$

where  $i^2 = -1$ . The following is the inversion formula:

$$X(t) = \int_{-\infty}^{+\infty} \hat{X}(\omega)e^{i2\pi\omega t} d\omega.$$

Since digital signal processing (DSP) has become the mainstream in signal processing, and only discrete transforms can be implemented in the digital domain, it is more interesting to consider the discrete version of Fourier transform.

For a finite sequence  $X[0], X[1], \dots, X[N-1]$ , where  $N$  is a given integer, the discrete Fourier transform is defined as

$$\hat{X}[l] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[k]e^{-i\frac{2\pi}{N}kl}, \quad 0 \leq l \leq N-1; \quad (3.1)$$

and consequently

$$X[k] = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \hat{X}[l]e^{i\frac{2\pi}{N}kl}, \quad 0 \leq k \leq N-1.$$

The discrete Fourier transform can be considered as a matrix-vector multiplication. Let  $\vec{X}$  and  $\vec{\hat{X}}$  denote the vectors made by the original signal and its Fourier transform respectively:

$$\vec{X} = \begin{pmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{pmatrix}_{N \times 1} \quad \text{and} \quad \vec{\hat{X}} = \begin{pmatrix} \hat{X}[0] \\ \hat{X}[1] \\ \vdots \\ \hat{X}[N-1] \end{pmatrix}_{N \times 1}.$$

Also let  $\mathbf{DFT}_N$  denote the  $N$ -point DFT matrix

$$\mathbf{DFT}_N = \left\{ e^{-i\frac{2\pi}{N}kl} \right\}_{\substack{k=0,1,2,\dots,N-1; \\ l=0,1,2,\dots,N-1}},$$

where  $k$  is the row index and  $l$  is the column index. We have

$$\vec{\hat{X}} = \mathbf{DFT}_N \cdot \vec{X}.$$

Apparently  $\mathbf{DFT}_N$  is an  $N$  by  $N$  square symmetric matrix. Both  $\vec{X}$  and  $\vec{\hat{X}}$  are  $N$ -dimensional vectors.

It is well known that the matrix  $\mathbf{DFT}_N$  is orthogonal, which is equivalent to saying that the inverse of  $\mathbf{DFT}_N$  is its complex conjugate transpose. Thus, DFT is an orthogonal transform and consequently it is an isometric transform. This result is generally attributed to Parseval.

Since the concise statements and proofs can be found in many textbooks, we skip the proof.

### Why Fourier Transform?

The preponderant reason that Fourier analysis is so powerful in analyzing linear time invariant system and cyclic-stationary time series is the fact that Fourier series are the eigenvectors of matrices associated with these transforms. These transforms are ubiquitous in DSP and time series analysis.

Before we state the key result, let us first introduce some mathematical notation: *Toeplitz matrix*, *Hankel matrix* and *circulant matrix*.

Suppose  $A_{N \times N}$  is an  $N$  by  $N$  real-valued matrix:  $A = \{a_{ij}\}_{1 \leq i \leq N, 1 \leq j \leq N}$  with all  $a_{ij} \in \mathbb{R}$ . The matrix  $A$  is a Toeplitz matrix when the elements on the diagonal, and the rows that are parallel to the diagonal, are the same, or mathematically  $a_{ij} = t_{i-j}$  [74, page 193]. The following is a Toeplitz matrix:

$$T = \begin{pmatrix} t_0 & t_{-1} & \dots & t_{-(n-2)} & t_{-(n-1)} \\ t_1 & t_0 & \dots & t_{-(n-3)} & t_{-(n-2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{n-2} & t_{n-3} & \dots & t_0 & t_{-1} \\ t_{n-1} & t_{n-2} & \dots & t_1 & t_0 \end{pmatrix}_{N \times N}.$$

A matrix  $A$  is a Hankel matrix when its elements satisfying  $a_{ij} = h_{i+j-1}$ . A Hankel matrix is symbolically a flipped (left-right) version of a Toeplitz matrix. The following is a Hankel matrix:

$$H = \begin{pmatrix} h_1 & h_2 & \dots & h_{n-1} & h_n \\ h_2 & h_3 & \dots & h_n & h_{n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{n-1} & h_n & \dots & h_{2n-3} & h_{2n-2} \\ h_n & h_{n+1} & \dots & h_{2n-2} & h_{2n-1} \end{pmatrix}_{N \times N} .$$

A matrix  $A$  is called a circulant matrix [74, page 201] when  $a_{ij} = c_{\{i-j \bmod N\}}$ , where  $x \bmod N$  is the non-negative remainder after a modular division. The following is a circulant matrix:

$$C = \begin{pmatrix} c_0 & c_{N-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & \dots & c_3 & c_2 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ c_{N-2} & c_{N-3} & \dots & c_0 & c_{N-1} \\ c_{N-1} & c_{N-2} & \dots & c_1 & c_0 \end{pmatrix}_{N \times N} . \quad (3.2)$$

Obviously, a circulant matrix is a special type of Toeplitz matrix.

The key mathematical result that makes Fourier analysis so powerful is the following theorem.

**Theorem 3.1** *For an  $N \times N$  circulant matrix  $C$  as defined in (3.2), the Fourier series  $\{e^{-j\frac{2\pi}{N}kl} : k = 0, 1, 2, \dots, N-1\}$ , for  $l = 0, 1, 2, \dots, N-1$ , are eigenvectors of the matrix  $C$ , and the Fourier transforms of the sequence  $\{\sqrt{N}c_0, \sqrt{N}c_1, \dots, \sqrt{N}c_{N-1}\}$  are its eigenvalues.*

A sketch of the proof is given in Section 3.7.

There are two important applications of this theorem. (Basically, they are the two problems that were mentioned at the beginning of this subsection.)

1. For a cyclic linear time-invariant (LTI) system, the transform matrix is a circulant matrix. Thus, Fourier series are the eigenvectors of a cyclic linear time invariant system, and the Fourier transform of the square root  $N$  amplified impulse response is

the set of eigenvalues of this system. Note: when the signal is infinitely long and the impulse response is relatively short, we can drop the cyclic constraint. Since an LTI system is a widely assumed model in signal processing, the Fourier transform plays an important role in analyzing this type of system.

2. A covariance matrix of a stationary Gaussian (i.e., Normal) time series is Toeplitz and symmetric. If we can assume that the time series does not have long-range dependency<sup>2</sup>, which implies that off-diagonal elements diminish when they are far from the diagonal, then the covariance matrix is almost circulant. Hence, the Fourier series again become the eigenvectors of this matrix. The DFT matrix  $\mathbf{DFT}_N$  nearly diagonalizes the covariance matrix, which implies that after the discrete Fourier transform of the time series, the coefficients are almost independently distributed. This idea can be depicted as

$$\mathbf{DFT}_N \cdot \{\text{the covariance matrix}\} \cdot \mathbf{DFT}_N^T \approx \{\text{a diagonal matrix}\}.$$

Generally speaking, it is easier to process independent coefficients than a correlated time series. The eigenvalue amplitudes determine the variance of the associated coefficients. Usually, the eigenvalue amplitudes decay fast: say, exponentially. The coefficients corresponding to the eigenvalues having large amplitudes are important coefficients in analysis. The remaining coefficients could almost be considered constant. Some research in this direction can be found in [58], [59] and [101].

From the above two examples, we see that FT plays an important role in both DSP and time series analysis.

### Fast Fourier Transform

To illustrate the importance of the fast Fourier transform in contemporary computational science, we find that the following sentences (the first paragraph of the Preface in [136]) say exactly what we would like to say.

The fast Fourier transform (FFT) is one of the truly great computational developments of this century. It has changed the face of science and engineering so

---

<sup>2</sup>No long-range dependency means that if two locations are far away in the series, then the corresponding two random variables are nearly independent.

much so that it is not an exaggeration to say that *life as we know it would be very different without the FFT*. — Charles Van Loan [136].

There are many ways to derive the FFT. The following theorem, which is sometimes called Butterfly Theorem, seems the most understandable way to describe why FFT works. The author would like to thank Charles Chui [30] for being the first fellow to introduce this theorem to me. The original idea started from Cooley and Tukey [32].

To state the theorem, let us first give some notation for matrices. Let  $P_1$  and  $P_2$  denote two permutation matrices satisfying for any  $mn$ -dimensional vector  $x = (x_0, x_1, \dots, x_{mn-1}) \in \mathbb{R}^{mn}$ , we have,

$$P_1 \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{mn-1} \end{pmatrix}_{mn \times 1} = \begin{pmatrix} \begin{pmatrix} x_0 \\ x_n \\ \vdots \\ x_{(m-1)n} \end{pmatrix}_{m \times 1} \\ \begin{pmatrix} x_1 \\ x_{1+n} \\ \vdots \\ x_{1+(m-1)n} \end{pmatrix}_{m \times 1} \\ \bullet \\ \bullet \\ \bullet \\ \begin{pmatrix} x_{(n-1)} \\ x_{(n-1)+n} \\ \vdots \\ x_{(n-1)+(m-1)n} \end{pmatrix}_{m \times 1} \end{pmatrix}; \quad (3.3)$$



Armed with these definitions, we are now able to state the Butterfly Theorem. The following is a formal statement.

**Theorem 3.2 (Butterfly Theorem)** *Consider any two positive integers  $m$  and  $n$ . Let  $\mathbf{DFT}_{mn}$ ,  $\mathbf{DFT}_m$  and  $\mathbf{DFT}_n$  be the  $mn$ -,  $m$ - and  $n$ - point discrete Fourier transform matrices respectively. The matrices  $P_1$ ,  $P_2$  and  $\Omega_{mn}$  are defined in (3.3), (3.4) and (3.5), respectively. The following equality is true:*

$$\mathbf{DFT}_{mn} = P_1^T \left[ \begin{array}{ccc} \mathbf{DFT}_m & & \\ & \ddots & \\ & & \mathbf{DFT}_m \\ \hline & & \underbrace{\hspace{10em}}_n \end{array} \right] P_2 \Omega_{mn} \left[ \begin{array}{ccc} \mathbf{DFT}_n & & \\ & \ddots & \\ & & \mathbf{DFT}_n \\ \hline & & \underbrace{\hspace{10em}}_m \end{array} \right] P_2^T. \quad (3.6)$$

The proof would be lengthy and we omit it. At the same time, it is not difficult to find a standard proof in the literature; see, for example, [136] and [134].

Why does Theorem 3.2 lead to a *fast* DFT? To explain this, we first use (3.6) in the following way. Suppose  $N$  is even. Letting  $m = N/2$  and  $n = 2$ , we have

$$\mathbf{DFT}_N = P_1^T \left[ \begin{array}{ccc} \mathbf{DFT}_{N/2} & & \\ & \ddots & \\ & & \mathbf{DFT}_{N/2} \\ \hline & & \underbrace{\hspace{10em}}_2 \end{array} \right] P_2 \Omega_N \left[ \begin{array}{ccc} \mathbf{DFT}_2 & & \\ & \ddots & \\ & & \mathbf{DFT}_2 \\ \hline & & \underbrace{\hspace{10em}}_{N/2} \end{array} \right] P_2^T. \quad (3.7)$$

Let  $C(N)$  denote the number of operations needed to multiply a vector with matrix  $\mathbf{DFT}_N$ . Operations include shifting, scalar multiplication and scalar addition. Since  $P_1$  and  $P_2$  are permutation matrices, it takes  $N$  shifting operations to do vector-matrix-multiplication with  $P_1^T$ ,  $P_2$  and  $P_2^T$ . Note that  $\mathbf{DFT}_2$  is a  $2 \times 2$  matrix. Obviously, it takes  $3N$  operations ( $2N$

multiplications and  $N$  additions) to multiply with the following matrix:

$$\left[ \begin{array}{ccc} \mathbf{DFT}_2 & & \\ & \ddots & \\ & & \mathbf{DFT}_2 \end{array} \right].$$

$\underbrace{\hspace{10em}}_{N/2}$

Finally, since  $\Omega_N$  is a diagonal matrix, it takes  $N$  multiplications to multiply with it. From all the above, together with (3.7), we can derive the following recursive relationship:

$$\begin{aligned} C(N) &= N + 2C(N/2) + N + N + 3N + N \\ &= 2C(N/2) + 7N. \end{aligned} \tag{3.8}$$

If  $N = 2^m$ , we have

$$\begin{aligned} C(N) &\stackrel{\text{from (3.8)}}{=} 2C(N/2) + 7N \\ &= \dots \\ &= 2^{m-1}C(2) + 7(m-1)N \\ &= 7N \log_2 N - 6N \\ &\asymp N \log_2 N, \end{aligned} \tag{3.9}$$

where symbol “ $\asymp$ ” means that asymptotically both sides have the same order. More specifically, we have

$$\lim_{N \rightarrow +\infty} \frac{C(N)}{N \log_2 N} = \text{a constant.}$$

We see that the complexity of the DFT can be as low as  $O(N \log_2 N)$ . Since the matrix corresponding to the inverse DFT (IDFT) is the element-wise conjugate of the matrix corresponding to DFT, the same argument also applies to the IDFT. Hence there is an  $O(N \log_2 N)$  algorithm for the inverse discrete Fourier transform as well.

We can utilize FFT to do fast convolution, According to the following theorem.

**Theorem 3.3 (Convolution Theorem)** *Consider two finite-length sequences  $X$  and  $Y$*

of length  $N$ ,  $N \in \mathbb{N}$ :

$$\begin{aligned} X &= \{X_0, X_1, \dots, X_{N-1}\}, \\ Y &= \{Y_0, Y_1, \dots, Y_{N-1}\}. \end{aligned}$$

Let  $\widehat{X}$  and  $\widehat{Y}$  denote the discrete Fourier transform (as defined in (3.1)) of  $X$  and  $Y$  respectively. Let  $X * Y$  denote the convolution of these two sequences:

$$X * Y[k] = \sum_{i=0}^k X[i]Y[k-i] + \sum_{i=k+1}^{N-1} X[i]Y[N+k-i], \quad k = 0, 1, 2, \dots, N-1.$$

Let  $\widehat{X * Y}$  denote the discrete Fourier transform of the sequence  $X * Y$ . We have

$$\widehat{X * Y}[k] = \widehat{X}[k]\widehat{Y}[k], \quad k = 0, 1, 2, \dots, N-1. \quad (3.10)$$

In other words, the discrete Fourier transform of a convolution of any two sequences, is equal to the elementwise multiplication of the discrete Fourier transforms of these two sequences.

We skip the proof.

A crude implementation of the convolution would take  $N$  multiplications and  $N-1$  additions to calculate each element in  $\widehat{X * Y}$ , and hence  $N(2N-1)$  operations to get the sequence  $\widehat{X * Y}$ . This is an order  $O(N^2)$  algorithm.

From (3.10), we can first calculate the DFT of the sequences  $X$  and  $Y$ . From the result in (3.9), this is an  $O(N \log_2 N)$  algorithm. Then we multiply the two DFT sequences, using  $N$  operations. Then we can calculate the inverse DFT of the sequence  $\{\widehat{X}[k]\widehat{Y}[k] : k = 0, 1, \dots, N-1\}$ : another  $O(N \log_2 N)$  algorithm. Letting  $C'(N)$  denote the overall complexity of the method, we have

$$\begin{aligned} C'(N) &= 2C(N) + N + C(N) \\ &= 3C(N) + N \\ &\stackrel{\text{by (3.9)}}{\asymp} N \log_2 N. \end{aligned}$$

Hence we can do fast convolution with complexity  $O(N \log_2 N)$ , which is lower than order  $O(N^2)$ .

### 3.1.2 Discrete Cosine Transform

#### Definition

Coefficients of the discrete cosine transform (DCT) are equal to the inner product of a discrete signal and a discrete cosine function. The discrete cosine function is an equally spaced sampling of a cosine function. Note that we stay in a discrete setting. Let the sequence

$$X = \{X[l] : l = 0, 1, 2, \dots, N - 1\}$$

be the signal sequence, and denote the DCT of  $X$  by

$$Y = \{Y[k] : k = 0, 1, 2, \dots, N - 1\}.$$

Then by definition, each  $Y[k]$  is given by

$$Y[k] = \sum_{l=0}^{N-1} C_{kl} X[l], \quad 0 \leq k \leq N - 1,$$

and the element  $C_{kl}$  is determined by

$$C_{kl} = \alpha_1(k) \alpha_2(l) \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(k + \delta_1)(l + \delta_2)}{N}\right), \quad 0 \leq k, l \leq N - 1. \quad (3.11)$$

For  $i = 1, 2$ , the choice of function  $\alpha_i(k)$ ,  $k \in \mathbb{N}$  is determined by the parameter  $\delta_i$ :

$$\left\{ \begin{array}{l} \text{if } \delta_i = 0, \text{ then } \alpha_i(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } k = 0, \\ 1, & \text{if } k = 1, 2, \dots, N - 1; \end{cases} \\ \text{if } \delta_i = 1/2, \text{ then } \alpha_i(k) \equiv 1. \end{array} \right.$$

The purpose of choosing different values for the function  $\alpha_i(k)$  is to make the DCT an orthogonal transform, or equivalently to make the transform  $C_N = \{C_{kl}\}_{\substack{k=0,1,\dots,N-1 \\ l=0,1,\dots,N-1}}$  an orthonormal matrix.

A good reference book about DCT is Rao and Yip [121].

By choosing different values for  $\delta_1$  and  $\delta_2$  in (3.11), we can define four types of DCT, as summarized in Table 3.2.

Type of DCT	$\delta_1$	$\delta_2$	transform matrix
type-I DCT	0	0	$C_N^I$
type-II DCT	0	1/2	$C_N^{II}$
type-III DCT	1/2	0	$C_N^{III}$
type-IV DCT	1/2	1/2	$C_N^{IV}$

Table 3.2: Definitions of four types of DCTs.

For all four types, the DCT matrices are orthogonal ( $C_N^{-1} = C_N^T$ ). Since they are real, of course they are unitary. Another noteworthy fact is that  $C_N^{II} = (C_N^{III})^{-1}$ .

We can view the four types of DCT as choosing the first index (which usually corresponds to the frequency variable) and the second index (which usually corresponds to the time variable) at different places (which can be either integer points or half integer points). Figure 3.1 depicts this idea.

### DCT via DFT

Particularly for the type-I DCT, the transform matrix  $C_N^I$  is connected to the transform matrix of a  $2N$ -point DFT ( $\mathbf{DFT}_{2N}$ ) in the following way:

$$\begin{bmatrix} C_N^I & \bullet \\ \bullet & \bullet \end{bmatrix} = \mathcal{R} \left[ \begin{pmatrix} \frac{1}{\sqrt{2}} & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \mathbf{DFT}_{2N} \begin{pmatrix} \frac{1}{\sqrt{2}} & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \right],$$

where  $\mathcal{R}$  is a real operator:  $\mathcal{R}(A)$  is the real part of matrix  $A$ . Symbol “ $\bullet$ ” represents an arbitrary  $N \times N$  matrix.

The other types of DCT, generally speaking, can be implemented via the  $8N$ -point DFT. We explain the idea for the case of type-IV DCT. For type-II and type-III DCT, a similar idea should work. Note that in practice, there are better computational schemes.

Let us first give some mathematical notation. Let symbol  $\mathcal{I}_{N \rightarrow 8N}$  denote an insert operator: it takes an  $N$ -D vector and expand it into an  $8N$ -D vector, so that the subvector at locations  $\{2, 4, 6, \dots, 2N\}$  of the  $8N$ -D vector is exactly the  $N$ -D vector, and all other

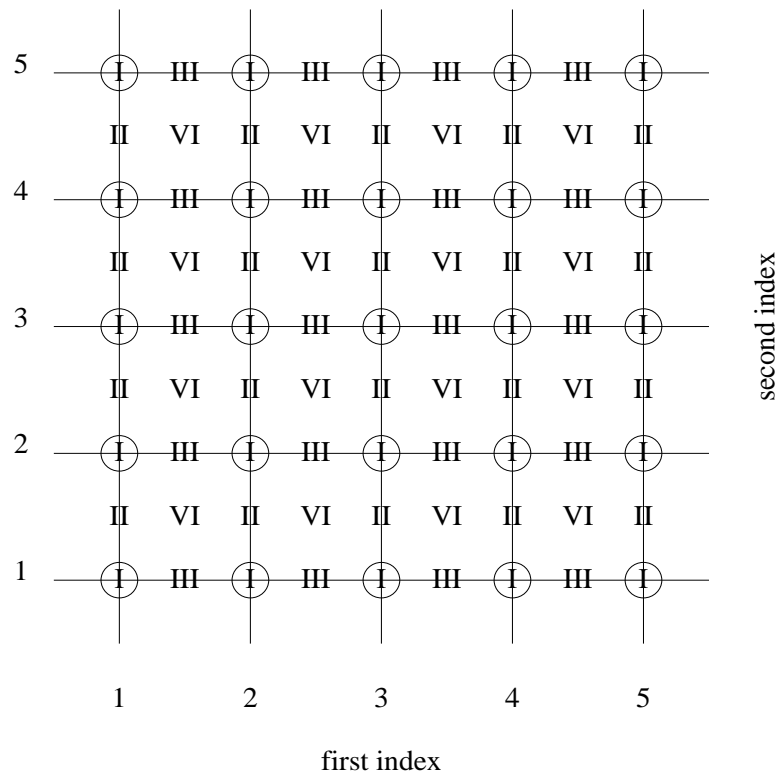


Figure 3.1: Sampling of four types of DCT at the index domain. For example, both the first index and the second index for the type-I DCT take integral values.

elements in the  $8N$ -D vector are zero. Equivalently, for  $\{X[0], X[1], \dots, X[N-1]\} \in \mathbb{R}^N$ ,

$$\begin{aligned} & \mathcal{I}_{N \rightarrow 8N}(\{X[0], X[1], \dots, X[N-1]\}) \\ &= \{0, X[0], 0, X[1], 0, \dots, 0, X[N-1], \underbrace{0, \dots, 0}_{6N}\} \in \mathbb{R}^{8N}. \end{aligned}$$

Let symbol  $\mathcal{S}_{8N \rightarrow N}$  denote the inverse operator of  $\mathcal{I}_{N \rightarrow 8N}$ . The operator  $\mathcal{S}_{8N \rightarrow N}$  is actually a downsampler:

$$\mathcal{S}_{8N \rightarrow N}(\{\bullet, X[0], \bullet, X[1], \bullet, \dots, \bullet, X[N-1], \underbrace{\bullet, \dots, \bullet}_{6N}\}) = \{X[0], X[1], \dots, X[N-1]\},$$

where “ $\bullet$ ” represents any complex number. If  $\mathbf{DFT}_{8N}$  denotes the  $8N$ -point DFT operator and  $\mathbf{DCT}_N^{IV}$  denotes the  $N$ -point type-IV DCT operator, then the following is true (note the index of DFT starts at zero):

$$\mathbf{DCT}_N^{IV} = \mathcal{R} \circ \mathcal{S}_{8N \rightarrow N} \circ \mathbf{DFT}_{8N} \circ \mathcal{I}_{N \rightarrow 8N}.$$

In order to do an  $N$ -point type-IV DCT, we can first insert the sequence into an eight-times expanded sequence, and then apply the  $8N$ -point DFT to the expanded vector, and then downsample the output of DFT, and then take the real part of the downsampled result. Note in this section, we always assume that the input is a real sequence or a real vector.

The reason that we need a much higher dimension (8 times) in DFT to implement DCT is because DCT could be sampled at half integer points (multipliers of  $1/2$ ).

### Summary of Development in Fast Algorithms

We summarize the development of fast algorithms for DCTs. Since the type-II DCT has been used in JPEG—an international standard for image coding, compression and transmission—research in this area has been very active. The 2-D DCT on small blocks—for example, an  $8 \times 8$  block or an  $16 \times 16$  block—is of particular importance. JPEG uses 2-D DCT on an  $8 \times 8$  block.

Let’s start with the 1-D DCT. Table 3.3 summarizes some key results. (The table is originally from Antonio Ortega for a talk given at ISL, Stanford University.) By definition, the 1-D  $N$ -point DCT takes  $N^2$  multiplications and  $N(N-1)$  additions; when  $N = 8$ , it is equivalent to 64 multiplications and 56 additions. In 1977 [28], Chen, Smith and

Fralick developed a fast algorithm that would take  $N \log_2 N - 3N/2 + 4$  multiplications and  $3N/2(\log_2 N - 1) + 2$  additions; when  $N = 8$ , their algorithm would take 16 multiplications and 26 additions. This is a significant reduction in complexity. In 1984, Wang [139] gave an algorithm that would take  $N(3/4 \log_2 N - 1) + 3$  multiplications and  $N(7/4 \log_2 N - 2) + 3$  additions; when  $N = 8$ , this is 13 multiplications and 29 additions. Also in 1984, Lee [92] introduced an algorithm that would take  $N/2 \log_2 N$  multiplications and  $3N/2 \log_2 N - N + 1$  additions; when  $N = 8$ , this is 12 multiplications (one less than 13) and 29 additions.

Algorithm	# of Multiplications	# of Additions
Definition	$N^2$ {64}	$N(N - 1)$ {56}
Chen, Smith and Fralick, 1977 [28]	$N \log_2 N - 3N/2 + 4$ {16}	$3N/2(\log_2 N - 1) + 2$ {26}
Wang, 1984 [139]	$N(3/4 \log_2 N - 1) + 3$ {13}	$N(7/4 \log_2 N - 2) + 3$ {29}
Lee, 1984, [92]	$N/2 \log_2 N$ {12}	$3N/2 \log_2 N - N + 1$ {29}
Duhamel, 1987 [62] (theoretical bound)	$2N - \log_2 N - 2$ {11}	Not Applicable

Table 3.3: Number of multiplications and additions for various fast one-D DCT/IDCT algorithms. The number in  $\{ \cdot \}$  is the value when  $N$  is equal to 8.

The overall complexity of the DCT arises from two parts: *multiplicative complexity* and *additive complexity*. The multiplicative complexity is the minimum number of non-rational multiplications necessary to perform DCTs. The additive complexity, correspondingly, is the minimum number of additions necessary to perform DCTs. Since it is more complex to implement multiplications than additions, it is more important to consider multiplicative complexity. In 1987, Duhamel [62] gave a theoretical bound on the 1-D  $N$ -point DCT: it takes at least  $2N - \log_2 N - 2$  multiplications. In 1992, Feig and Winograd [67] extended this result to an arbitrary dimensional DCT with input sizes that are powers of two. Their conclusion is that for  $L$ -dimensional DCTs whose sizes at coordinates are  $2^{m_1}, 2^{m_2}, \dots, 2^{m_L}$ , with  $m_1 \leq m_2 \leq \dots \leq m_L$ , the multiplicative complexity is lower bounded by  $2^{m_1+m_2+\dots+m_{L-1}}(2^{m_L+1} - m_L - 2)$ . When  $L = 1$ , this result is the same as Duhamel's result [62].

In image coding, particularly in JPEG, 2-D DCTs are being used. In 1990, Duhamel

and Guillemot [61] derived a fast algorithm for the 2-D DCT that would take 96 multiplications and more than 454 additions for an  $8 \times 8$  block. In 1992, Feig and Winograd [66] proposed another algorithm that would take  $94 (< 96)$  multiplications and 454 additions for an  $8 \times 8$  block. They also mentioned that there is an algorithm that would take 86 ( $< 94$ ) multiplications, but it would take too many additions to be practical.

In practical image coding/compression, a DCT operator is usually followed by a quantizer. Sometimes it is not necessary to get the exact values of the coefficients; we only need to get the scaled coefficients. This leads to a further saving in the multiplicative complexity. In [66], Feig and Winograd documented a scaled version of DCT that would take only 54 multiplications and 462 additions. Moreover, in their fast scaled version of the 2-D DCT, there is no computation path that uses more than one multiplication. This makes parallel computing feasible.

### 3.1.3 Discrete Sine Transform

As for the DCT, the output of the 1-D discrete sine transform (DST) is the inner product of the 1-D signal and the equally sampled sine function. The sampling frequencies are also equally spaced in the frequency domain. Again, there are four types of DST. More specifically, if we let

$$X = \{X[l] : l = 0, 1, 2, \dots, N - 1\}$$

be the signal sequence, and denote the DST of  $X$  by

$$Y = \{Y[k] : k = 0, 1, 2, \dots, N - 1\},$$

we have

$$Y[k] = \sum_{l=0}^{N-1} S_N(k, l)X[l], \quad k = 0, 1, 2, \dots, N - 1,$$

where  $S_N(k, l)$ ,  $k, l = 0, 1, 2, \dots, N - 1$ , are values of sine functions. For the four types of DST, let

$$b_k := \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \text{ or } N, \\ 1 & \text{if } k = 1, \dots, N - 1; \end{cases} \quad (3.12)$$

then we have

- DST-I:  $S_N(k, l) = \sqrt{\frac{2}{N}} \sin \frac{\pi kl}{N}$ ;
- DST-II:  $S_N(k, l) = b_{k+1} \sqrt{\frac{2}{N}} \sin \frac{\pi(k+1)(l+\frac{1}{2})}{N}$ ;
- DST-III:  $S_N(k, l) = b_{l+1} \sqrt{\frac{2}{N}} \sin \frac{\pi(k+\frac{1}{2})(l+1)}{N}$ ;
- DST-IV:  $S_N(k, l) = \sqrt{\frac{2}{N}} \sin \frac{\pi(k+\frac{1}{2})(l+\frac{1}{2})}{N}$ .

For all the four types of DST, the transform matrices

$$\{S_N(k, l)\}_{\substack{k=0,1,2,\dots,N-1 \\ l=0,1,2,\dots,N-1}}$$

are orthogonal (and also unitary).

If  $\mathbf{DST}_N^{II}$  and  $\mathbf{DST}_N^{III}$  denote the DST-II and DST-III operators, similar to DCT, we have

$$\mathbf{DST}_N^{II} = (\mathbf{DST}_N^{III})^{-1}.$$

### 3.1.4 Homogeneous Components

The reason that the DCT is so powerful in analyzing homogeneous signals is that it is nearly (in some asymptotic sense) the Karhunen-Loève transform (KLT) of some Gaussian Markov Random Fields (GMRFs). In this section, we first describe the definition of Gaussian Markov Random Fields, then argue that the covariance matrix is the key statistic for GMRFs; for a covariance matrix, we give the necessary and sufficient conditions of diagonalizability of different types of DCTs; finally we conclude that under some appropriate boundary conditions, the DCT is the KLT of GMRFs.

As we stated earlier, in this thesis, not much attention is given to mathematical rigor.

#### Gaussian Markov Random Field

This subsection is organized in the following way: we start with the definition of a *random field*, then introduce the definition of a *Markov random field* and a *Gibbs random field*; the Hammersley-Clifford theorem creates an equivalence between a Markov random field and a Gibbs random field; then we describe the definition of a *Gaussian Markov random field*; eventually we argue that DCT is a KLT of a GMRF.

*Definition of a random field.* We define a random field on a lattice. Let  $\mathbb{Z}^d$  denote the  $d$ -dimensional integers, or the lattice points, in  $d$ -dimensional space, which is denoted by  $\mathbb{R}^d$ . The finite set  $D$  is a subset of  $\mathbb{Z}^d$ :  $D \subset \mathbb{Z}^d$ . For two lattice points  $x, y \in \mathbb{Z}^d$ , let  $|x - y|$  denote the Euclidean distance between  $x$  and  $y$ . The set  $D$  is *connected* if and only if for any  $x, y \in D$ , there exists a finite subset  $\{x_1, x_2, \dots, x_n\}$  of  $D$ ,  $n \in \mathbb{N}$ , such that (1)  $|x - x_1| \leq 1$ , (2)  $|x_i - x_{i+1}| \leq 1, i = 1, 2, \dots, n - 1$ , and (3)  $|x_n - y| \leq 1$ . We call a connected set  $D$  a *domain*. The dimension of the set  $D$  is, by definition, the number of integer points in the set  $D$ . We denote the dimension of  $D$  by  $\dim(D)$ . On each lattice point in set  $D$ , a real value is assigned. The set  $\mathbb{R}^D$ , which is equivalent to  $\mathbb{R}^{\dim(D)}$ , is called a *state space*. Follow some conventions, we denote the state space by  $\Omega$ , so we have  $\Omega = \mathbb{R}^{\dim(D)}$ . Let  $\mathcal{F}$  be the  $\sigma$ -algebra that is generated from the Borel sets in  $\Omega$ . Let  $\mathcal{P}$  be the Lebesgue measure. The triple  $(\Omega, \mathcal{F}, \mathcal{P})$  is called a random field (RF) on the domain  $D$ .

Note that we define a random field on a subset of all the lattice points.

Now we give the definition of a *neighbor*. Intuitively, under Euclidean distance, two integer (lattice) points  $x$  and  $y$  are neighbors when  $|x - y| \leq 1$ . This definition can be extended. We define a non-negative, symmetric and translation-invariant bivariate function  $N(x, y)$  on domain  $D$ , such that for  $x, y \in D$ , the function  $N$  satisfies

1.  $N(x, x) = 0$ ,
2.  $N(x, y) \geq 0$  (non-negativity),
3.  $N(x, y) = N(y, x)$  (symmetry),
4.  $N(x, y) = N(0, y - x)$  (homogeneity, or translation invariance).

Any two points are called *neighbors* if and only if  $N(x, y) > 0$ . For example, in Euclidean space, if we let  $N(x, y) = 1$  when  $|x - y| = 1$ , and  $N(x, y) = 0$  elsewhere, then we have the ordinary definition of neighbor that is mentioned at the beginning of this paragraph.

*Definition of a Markov random field.* The definition of a Markov random field is based upon conditional probability. The *key* idea of Markovity is that conditional probability should depend only on neighbors. To be more precise, we need some terminology. Let  $\omega$  denote an element of  $\Omega$ . We call  $\omega$  a *realization*. Let  $p(\omega)$  denote the probability density function of  $\omega$ . The p.d.f.  $p$  is associated with the Lebesgue measure  $\mathcal{P}$ . Let  $\omega(x)$  be the value of the realization  $\omega$  at the point  $x$ . For a subset  $A \subset D$ , suppose the values at points

in  $A$  are given by a deterministic function  $f(\cdot)$ , the conditional p.d.f. at point  $x \in D$  given values in  $A$  is denoted by  $p\{\omega(x)|\omega(y) = f(y), y \in A \subset D\}$ . To simplify, we let

$$p\{\omega(x)|\omega(\cdot) = f(\cdot) \text{ on } A\} = p\{\omega(x)|\omega(y) = f(y), y \in A \subset D\}.$$

If the p.d.f. satisfies the following conditions:

1. positivity:  $p(\omega) > 0, \forall \omega \in D$ ,
2. Markovity:  $p\{\omega(x)|\omega(z) = f(z), z \in D, z \neq x\} = p\{\omega(x)|\omega(z) = f(z), z \in D, x \text{ and } z \text{ are neighbor}\}$ ,
3. translation invariance: if two functions  $f, f'$  satisfy  $f(x+z) = f'(y+z)$  for all  $z \in D$ , then  $p\{\omega(x)|\omega(\cdot) = f(\cdot), \text{ on } D - \{x\}\} = p\{\omega(y)|\omega(\cdot) = f'(\cdot), \text{ on } D - \{y\}\}$ ,

then the random field  $(\Omega, \mathcal{F}, \mathcal{P})$  is a *Markov random field* (MRF).

*Definition of a Gibbs random field.* The key idea of defining a Gibbs random field is writing the p.d.f. as an exponential function. Let  $p(\omega)$  represent the same p.d.f. as in the previous paragraph. A random field is a *Gibbs random field* (GRF) if function  $p(\omega)$  has the form

$$p(\omega) = Z^{-1} \exp \left( -\frac{1}{2} \sum_{x \in D} \sum_{y \in D} \omega(x)\omega(y)U(x, y) \right), \quad (3.13)$$

where the constant  $Z$  is a normalizing constant, and the function  $U(x, y)$  has the following properties: for  $x, y \in D$ , we have

1.  $U(x, y) = U(y, x)$  (symmetry),
2.  $U(x, y) = U(0, y - x)$  (homogeneity),
3.  $U(x, y) = 0$  when points  $x$  and  $y$  are not neighbors (nearest neighbor property).

The function  $U(x, y)$  is sometimes called *pair potential* [129].

Albeit the two types of random field are ostensibly different, they are actually equivalent. The following theorem, which is attributed to Hammersley and Clifford, creates the equivalence. Note that the theorem is not expressed in a rigorous form. We are just trying to demonstrate the idea.

**Theorem 3.4 (Hammersley-Clifford Theorem)** *Every Markov random field on a domain  $D$  is a Gibbs random field on  $D$  and vice versa.*

The rigorous statement and the actual proof is too long to be presented here, readers are referred to [12, 129] for technical details.

*Definition of a Gaussian Markov random field.* In (3.13), if the p.d.f.  $p(\omega)$  is also the p.d.f. of a multivariate Normal distribution, then there are two consequences: first, from the Hammersley-Clifford Theorem, it is a Markov random field; second, the random field is also Gaussian. We call such a random field a *Gaussian Markov random field* (GMRF). Let  $\vec{\omega}$  denote a vector corresponding to a realization  $\omega$ . (The vector  $\vec{\omega}$  is simply a list of all the values taken by  $\omega$ .) We further suppose the vector  $\vec{\omega}$  is a column vector. Let  $\Sigma$  denote the covariance matrix of the corresponding multivariate Normal distribution. We have

$$p(\omega) = (2\pi)^{-\frac{1}{2}\dim(D)} \frac{1}{\det^{1/2}(\Sigma)} \exp\left(-\frac{1}{2}\vec{\omega}^T \Sigma^{-1} \vec{\omega}\right).$$

This is a p.d.f. of a GMRF.

A Gaussian Markov random field is a way to model homogeneous images. In the image case, the dimensionality of the domain is 2 ( $d = 2$ ). The gray scale at each integer point in  $\mathbb{Z}^2$  is a continuous value. The fact that an image is homogeneous and only has the second order correlation, is equivalent to the fact that the image is a GMRF.

For a GMRF, if there is a transform that diagonalizes the covariance matrix  $\Sigma$ , then this transform is the KLT of the GMRF. Since we are discussing the connection between DCT and the homogeneous signal, the interesting question to ask is “*Which kind of covariance matrices does the DCT diagonalize?*”. We answer in the next subsection.

There have been some efforts to extend the GMRF model; for example, the work by Zhu, Wu and Mumford [145].

### Covariance Matrix Diagonalization

In this subsection, we answer the question raised in the previous subsection. We answer it in a converse manner. Instead of giving the conditions of the covariance matrices and then discussing the diagonalizability of matrices associated with this type of DCT, we try to find out which group of matrices can be diagonalized by which type of DCT.

The following Theorem is a general result. A reason we list it here is that we have not seen it documented explicitly in any other references.

The basic idea is that if the covariance matrix can be diagonalized by a type of DCT, then the covariance matrix must be written as a summation of two matrices. Moreover, the two matrices must have some particular properties. One matrix must be Toeplitz and symmetric, or such a matrix multiplied by some diagonal matrices. The other matrix must be Hankel and counter symmetric, or counter antisymmetric, or a matrix obtained by shifting such a matrix and then multiplying it with a diagonal matrix. Furthermore, for an  $N \times N$  covariance matrix, the dimensionality (or the degrees of freedom) of the matrix is no higher than  $N$ .

To be more clear, we have to use some notation. Let  $N$  denote the length of the signal, so the size of the covariance matrix is  $N \times N$ . Let  $b_k, k = 0, 1, \dots, N - 1$  be the sequence defined in (3.12). Let  $B$  be the diagonal matrix

$$B = \begin{pmatrix} b_0 & & & \\ & b_1 & & \\ & & \ddots & \\ & & & b_{N-1} \end{pmatrix}. \quad (3.14)$$

Let  $\{c_0, c_1, c_2, \dots, c_{2N-1}\}$  and  $\{c'_0, c'_1, c'_2, \dots, c'_{2N-1}\}$  be two finite real-valued sequences whose elements  $c_i$  and  $c'_i$  satisfy a symmetric or an antisymmetric condition:

$$\begin{cases} c_i = c_{2N-i} & i = 0, 1, 2, \dots, 2N - 1, \\ c'_i = -c'_{2N-i} & i = 0, 1, 2, \dots, 2N - 1. \end{cases}$$

Note that when  $i = N$ , the second condition implies  $c'_N = 0$ . Suppose  $C_1$  and  $C'_1$  are the Toeplitz and symmetric matrices

$$C_1 = \begin{pmatrix} c_0 & c_1 & \dots & c_{N-1} \\ c_1 & c_0 & & c_{N-2} \\ \vdots & & \ddots & \vdots \\ c_{N-1} & c_{N-2} & \dots & c_0 \end{pmatrix}, \quad \text{and} \quad C_2 = \begin{pmatrix} c'_0 & c'_1 & \dots & c'_{N-1} \\ c'_1 & c'_0 & & c'_{N-2} \\ \vdots & & \ddots & \vdots \\ c'_{N-1} & c'_{N-2} & \dots & c'_0 \end{pmatrix}. \quad (3.15)$$

Suppose  $C_2$  and  $C'_2$  are the two Hankel matrices

$$C_2 = \begin{pmatrix} c_1 & c_2 & \dots & c_{N-1} & c_N \\ c_2 & c_3 & \dots & c_N & c_{N-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-1} & c_N & \dots & c_3 & c_2 \\ c_N & c_{N-1} & \dots & c_2 & c_1 \end{pmatrix}, \quad (3.16)$$

$$C'_2 = \begin{pmatrix} c'_1 & c'_2 & \dots & c'_{N-1} & 0 \\ c'_2 & c'_3 & \dots & 0 & -c'_{N-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c'_{N-1} & 0 & \dots & -c'_3 & -c'_2 \\ 0 & -c'_{N-1} & \dots & -c'_2 & -c'_1 \end{pmatrix}. \quad (3.17)$$

Note that  $C_2$  is counter symmetric (if  $a_{\{i,j\}}$  is the  $\{i,j\}$ th element of the matrix, then  $a_{\{i,j\}} = a_{\{N+1-i,N+1-j\}}$ , for  $i, j = 1, 2, \dots, N$ ) and  $C'_2$  is counter antisymmetric (if  $a_{\{i,j\}}$  is the  $\{i,j\}$ th element of the matrix, then  $a_{\{i,j\}} = -a_{\{N+1-i,N+1-j\}}$ , for  $i, j = 1, 2, \dots, N$ ).

Let  $\mathcal{D}$  denote a down-right-shift operator on matrix, such that

$$\mathcal{D}(C_2) = \begin{pmatrix} c_0 & c_1 & \dots & c_{N-2} & c_{N-1} \\ c_1 & c_2 & \dots & c_{N-1} & c_N \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-2} & c_{N-1} & \dots & c_4 & c_3 \\ c_{N-1} & c_N & \dots & c_3 & c_2 \end{pmatrix}, \quad (3.18)$$

and

$$\mathcal{D}(C'_2) = \begin{pmatrix} c'_0 & c'_1 & \dots & c'_{N-2} & c'_{N-1} \\ c'_1 & c'_2 & \dots & c'_{N-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c'_{N-2} & c'_{N-1} & \dots & -c'_4 & -c'_3 \\ c'_{N-1} & 0 & \dots & -c'_3 & -c'_2 \end{pmatrix}. \quad (3.19)$$

Note that  $\mathcal{D}(C_2)$  and  $\mathcal{D}(C'_2)$  are still Hankel matrices.

Let  $\Sigma$  denote the covariance matrix, and let  $\Sigma_1$  and  $\Sigma_2$  denote two matrices that have special structures that we will specify later.

We will refer to the following table (Table 3.4) in the next Theorem.

	$\Sigma_1$ (Toeplitz part)	$\Sigma_2$ (Hankel part)	$\Sigma_1 + \Sigma_2$ (covariance matrix)
type-I	$B \cdot C_1 \cdot B$	$B \cdot \mathcal{D}(C_2) \cdot B$	$B \cdot C_1 \cdot B + B \cdot \mathcal{D}(C_2) \cdot B$
type-II	$C_1$	$C_2$	$C_1 + C_2$
type-III	$B \cdot C'_1 \cdot B$	$B \cdot \mathcal{D}(C'_2) \cdot B$	$B \cdot C'_1 \cdot B + B \cdot \mathcal{D}(C'_2) \cdot B$
type-IV	$C'_1$	$C'_2$	$C'_1 + C'_2$

Table 3.4: Decomposition of covariance matrix that can be diagonalized by different types of DCT.

**Theorem 3.5 (DCT Diagonalization)** *If a covariance matrix  $\Sigma$  can be diagonalized by one type of DCT, then it can be decomposed into two matrices,  $\Sigma = \Sigma_1 + \Sigma_2$ , and the matrices  $\Sigma_1$  and  $\Sigma_2$  have special structures as listed in Table 3.4. For example, if a covariance matrix  $\Sigma$  can be diagonalized by DCT-I, then*

$$\Sigma = \Sigma_1 + \Sigma_2 = B \cdot C_1 \cdot B + B \cdot \mathcal{D}(C_2) \cdot B.$$

*The matrices  $B, C_1, C'_1, C_2, C'_2, \mathcal{D}(C_2)$  and  $\mathcal{D}(C'_2)$  are defined in equations (3.14), (3.15), (3.16), (3.17), (3.18) and (3.19), respectively.*

*Moreover, the elements  $c_i$  and  $c'_i$  are determined by the following expressions:*

$$c_i = \frac{1}{N} \sum_{k=0}^{N-1} \lambda_k b_k^2 \cos \frac{\pi k i}{N}, i = 0, 1, \dots, 2N - 1,$$

$$c'_i = \frac{1}{N} \sum_{k=0}^{N-1} \lambda_k \cos \frac{\pi(k + \frac{1}{2})i}{N}, i = 0, 1, \dots, 2N - 1,$$

*where the sequence  $\{\lambda_k : k = 0, 1, \dots, N - 1\}$  is a real-valued sequence.*

The diagonal matrix

$$\text{diag}\{\lambda_0, \lambda_1, \dots, \lambda_{N-1}\} = \begin{pmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{pmatrix}$$

is the result of DCT diagonalization.

Since  $\Sigma$  is a covariance matrix, the value of  $\lambda_k$  is the variance of the  $k$ th coefficient of the corresponding DCT.

Why is this result meaningful? Because it specifies the particular structure of matrices that can be diagonalized by DCTs. Some previous research has sought the specifications (for example, the boundary conditions) of a GMRF so that the covariance matrix has one of the structures that are described in Table 3.4, or has a structure close to one of them. The research reported in [110, 111] is an example.

#### Conclusion of Section 3.1.4

When an image, or equivalently a GMRF, is homogeneous, the covariance matrix should be Toeplitz. Generally we also assume locality, or Markovity; hence the covariance matrix is nearly diagonal.

In order to show that DCT almost diagonalizes the covariance matrix of a class of homogeneous signals, we take type-II DCT as an example. We explain in a reverse order. If a covariance matrix can be diagonalized by type-II DCT, based on Theorem 3.5 and Table 3.4, the covariance matrix should look like  $C_1 + C_2$ . If we take the sequence  $\{c_0, c_1, \dots, c_N\}$ , such that the sequence has large amplitude at  $c_0$  and vanishing elements at the other end (which is equivalent to saying  $|c_i| \rightarrow 0$  as  $i \rightarrow 0$ ), then the Toeplitz part  $C_1$  becomes significant and the Hankel part  $C_2$  vanishes—only the upper-left corner and the bottom-right parts of the matrix  $C_2$  are significantly nonzero. It would be possible for us to choose a sequence  $\{c_0, c_1, \dots, c_N\}$  such that the matrix  $C_1 + C_2$  is a close approximation to the covariance matrix. Hence the type-II DCT can approximately diagonalize the covariance matrix, and is nearly the KLT of this class of images or GMRFs.

### 3.1.5 2-D DCT

The 2-D DCT is an extension of the 1-D DCT. On a matrix, we simply apply a 1-D DCT to each row, and then apply a 1-D DCT to each column. The basis function of the 2-D DCT is the tensor product of (two) basis functions for the 1-D DCT.

The optimality of the 2-D DCT is nearly a direct extension of the optimality of the 1-D DCT. As long as the homogeneous components of images are defined consistently with the definition corresponding to 1-D homogeneous components, we can show that the 2-D DCT will diagonalize the covariance matrices of the 2-D signal (images) too. Based on this, we can generate a slogan: “The 2-D DCT is good for images with homogeneous components.”

Another way to show the optimality of the DCT is to look at its basis functions. The properties of the basis functions represent the properties of a transform. A coefficient after a transform is equal to the inner product of an input and a basis function. Intuitively, if basis functions are similar to the input, then we only need a small number of basis functions to represent the original input. Note here that the input is a signal. When the basis functions are close to the features we want to capture in the images, we say that the corresponding transform is *good* for this type of images.

To show that the 2-D DCT is good for homogeneous images, let’s look at its basis functions. Figure 3.2 shows all 2-D DCT basis functions on an  $8 \times 8$  block. From these figures, we should say that the basis functions are pretty homogeneous, because it is impossible to say that the statistical properties in one area are different from the statistical properties in another.

## 3.2 Wavelets and Point Singularities

It is interesting to review the history of the development of wavelets. The following description is basically from Deslauriers and Dubuc [60, Preface]. In 1985 in France, the first orthogonal wavelet basis was discovered by Meyer. Shortly thereafter Mallat introduced multiresolution analysis, which explained some of the mysteries of Meyer’s wavelet basis. In February, 1987, in Montreal, Daubechies found an orthonormal wavelet basis that has compact support. Next came the biorthogonal wavelets related to splines. In Paris, in May 1985, Deslauriers and Dubuc used dyadic interpolation to explain multiresolution, a term that by then was familiar to many. Donoho and Johnstone developed the wavelet shrinkage

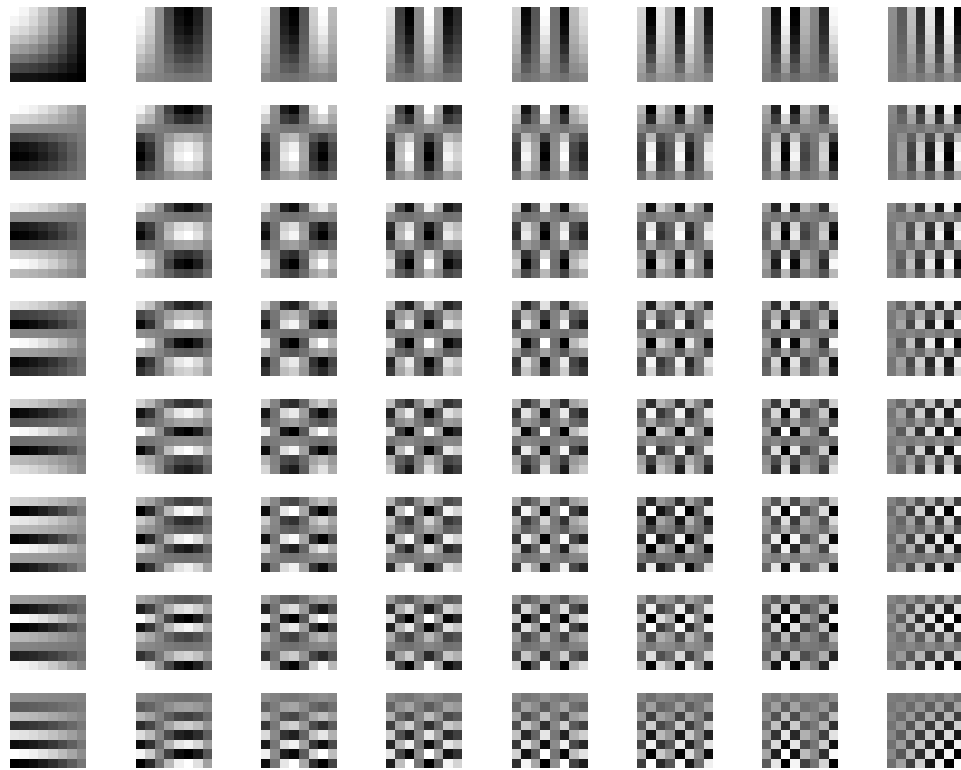


Figure 3.2: Two-dimensional discrete cosine transform basis functions on an  $8 \times 8$  block. The upper-left corner corresponds to the low-low frequency, and the lower-right corner corresponds to the high-high frequency.

method in denoising, density estimation, and signal recovery. Wavelets and related technologies have a lot of influence in contemporary fields like signal processing, image processing, statistic analysis, etc. Some good books that review this literature are [20, 34, 100, 107].

It is almost impossible to cover this abundant area in a short section. In the rest of the section, we try to summarize some key points. The ones we selected are (1) multiresolution analysis, (2) filter bank, (3) fast discrete algorithm, and (4) optimality in processing signals that have point singularities. Each of the following subsections is devoted to one of these subjects.

### 3.2.1 Multiresolution Analysis

*Multiresolution analysis* (MRA) is a powerful tool, from which some orthogonal wavelets can be derived.

It starts with a special multi-layer structure of square integrable functional space  $L^2$ . Let  $\mathcal{V}_j$ ,  $j \in \mathbb{Z}$ , denote some subspaces of  $L^2$ . Suppose the  $\mathcal{V}_j$ 's satisfy a special *nesting* structure, which is:

$$\dots \subset \mathcal{V}_{-3} \subset \mathcal{V}_{-2} \subset \mathcal{V}_{-1} \subset \mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \subset \dots \subset L^2.$$

At the same time, the following two conditions are satisfied: (1) the intersection of all the  $\mathcal{V}_j$ 's is a null set:  $\bigcap_{j \in \mathbb{Z}} \mathcal{V}_j = \emptyset$ ; (2) the union of all the  $\mathcal{V}_j$ 's is the entire space  $L^2$ :  $\bigcup_{j \in \mathbb{Z}} \mathcal{V}_j = L^2$ .

We can further assume that the subspace  $\mathcal{V}_0$  is spanned by functions  $\phi(x - k)$ ,  $k \in \mathbb{Z}$ , where  $\phi(x - k) \in L^2$ . By definition, any function in subspace  $\mathcal{V}_0$  is a linear combination of functions  $\phi(x - k)$ ,  $k \in \mathbb{Z}$ . The function  $\phi$  is called a *scaling function*. The set of functions  $\{\phi(x - k) : k \in \mathbb{Z}\}$  is called a *basis* of space  $\mathcal{V}_0$ . To simplify, we only consider the *orthonormal* basis, which, by definition, gives

$$\int \phi(x)\phi(x - k)dx = \delta(k) = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0, \end{cases} \quad k \in \mathbb{Z}. \quad (3.20)$$

A very essential assumption in MRA is the 2-scale relationship, which is: for  $\forall j \in \mathbb{Z}$ , if function  $f(x) \in \mathcal{V}_j$ , then  $f(2x) \in \mathcal{V}_{j+1}$ . Consequently, we can show that  $\{2^{j/2}\phi(2^j x - k) : k \in \mathbb{Z}\}$  is an orthonormal basis of the functional space  $\mathcal{V}_j$ . Since  $\phi(x) \in \mathcal{V}_0 \subset \mathcal{V}_1$ , there

must exist a real sequence  $\{h(n) : n \in \mathbb{Z}\}$ , such that

$$\phi(x) = \sum_{n \in \mathbb{Z}} h(n) \phi(2x - n). \quad (3.21)$$

Equation (3.21) is called a *two-scale relationship*. Based on different interpretations, or different points of view, it is also called *dilation equation*, *multiresolution analysis equation* or *refinement equation*. Let function  $\Phi(\omega)$  and function  $H(\omega)$  be the Fourier transforms of function  $\phi(x)$  and sequence  $\{h(n) : n \in \mathbb{Z}\}$ , respectively, so that

$$\begin{aligned} \Phi(\omega) &= \int_{-\infty}^{\infty} \phi(x) e^{-ix\omega} dx; \\ H(\omega) &= \sum_{n \in \mathbb{Z}} h(n) e^{-in\omega}. \end{aligned}$$

The two-scale relationship (3.21) can also be expressed in the Fourier domain:

$$\Phi(\omega) = H\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right).$$

Consequently, one can derive the result

$$\Phi(\omega) = \Phi(0) \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right). \quad (3.22)$$

The last equation is important for deriving compact support wavelets.

Since the functional subspace  $\mathcal{V}_j$  is a subset of the functional subspace  $\mathcal{V}_{j+1}$ , or equivalently  $\mathcal{V}_j \subset \mathcal{V}_{j+1}$ , we can find the orthogonal complement of the subspace  $\mathcal{V}_j$  in space  $\mathcal{V}_{j+1}$ . We denote the orthogonal complement by  $\mathcal{W}_j$ , so that

$$\mathcal{V}_j \oplus \mathcal{W}_j = \mathcal{V}_{j+1}. \quad (3.23)$$

Suppose  $\{\psi(x - k) : k \in \mathbb{Z}\}$  is a basis for the subspace  $\mathcal{W}_0$ . The function  $\psi$  is called a *wavelet*. Note that the function  $\phi$  is in the functional space  $\mathcal{V}_0$  and the function  $\psi$  is in its orthogonal complement. From (3.23), we say that a wavelet is designed to capture the fluctuations; and, correspondingly, a scaling function captures the trend. We can further

assume that the basis  $\{\psi(x - k) : k \in \mathbb{Z}\}$  is orthonormal:

$$\int \psi(x)\psi(x - k)dx = \delta(k), \quad k \in \mathbb{Z}. \quad (3.24)$$

Since  $\psi(x)$  is in  $\mathcal{V}_1$ , there must exist a real-valued sequence  $\{g(n) : n \in \mathbb{Z}\}$  such that

$$\psi(x) = \sum_{n \in \mathbb{Z}} g(n)\phi(2x - n). \quad (3.25)$$

Since functional subspace  $\mathcal{V}_0$  and functional subspace  $\mathcal{W}_0$  are orthogonal, we have

$$\int \psi(x)\phi(x - k) = 0, \quad k \in \mathbb{Z}. \quad (3.26)$$

From (3.20) and (3.21), we have

$$\sum_{n \in \mathbb{Z}} h(n)h(n - 2k) = 2\delta(k), \quad k \in \mathbb{Z}. \quad (3.27)$$

From (3.24) and (3.25), we have

$$\sum_{n \in \mathbb{Z}} g(n)g(n - 2k) = 2\delta(k), \quad k \in \mathbb{Z}. \quad (3.28)$$

From (3.26), (3.21) and (3.25), we have

$$\sum_{n \in \mathbb{Z}} g(n)h(n - 2k) = 0, \quad k \in \mathbb{Z}. \quad (3.29)$$

Actually, if relations (3.27), (3.28) and (3.29) hold, then the sequence  $g$  can only be the reverse of the sequence  $h$  modulated by the sequence  $\{(-1)^n : n \in \mathbb{Z}\}$ . The following theorem provides a formal description.

**Theorem 3.6 (Quadratic Mirror Filter)** *If two sequences  $\{h(n) : n \in \mathbb{Z}\}$  and  $\{g(n) : n \in \mathbb{Z}\}$  satisfy the relations listed in (3.27), (3.28) and (3.29), then the sequence  $\{g(n) : n \in \mathbb{Z}\}$  is uniquely determined by the sequence  $\{h(n) : n \in \mathbb{Z}\}$  via*

$$\forall n \in \mathbb{Z}, \quad g(n) = (-1)^n h(1 + 2k - n), \quad (3.30)$$

where  $k$  is a pre-determined integer.

After we specify the sequence  $\{h(n) : n \in \mathbb{Z}\}$ , the sequence  $\{g(n) : n \in \mathbb{Z}\}$  is determined. The Fourier transform of  $\{h(n) : n \in \mathbb{Z}\}$ , which is denoted by  $H(\omega)$ , is determined too. From relation (3.22), the function  $\phi(x)$  is determined, so the sequence  $\{h(n) : n \in \mathbb{Z}\}$  plays a deterministic role in the whole design scheme. The choice of  $\{h(n) : n \in \mathbb{Z}\}$  is determined by the conditions that are imposed on the wavelet functions. Some such conditions are symmetry, vanishing moments, compact support, etc. Details on how to design wavelet functions (or, equivalently, how to choose the  $h$  sequence) are not directly relevant to this thesis. Interested readers are referred to [34] and other books.

### 3.2.2 Filter Banks

The previous subsection is based on a functional analysis point of view—in particular, the multi-layer structure of the functional space. The *filter bank* takes a signal processing point of view. The basic idea is to partition a signal dyadically at the frequency domain. More specifically, we assume a signal is made by some basic components, with different components residing at different frequencies. Suppose we have a pair of filters: a “perfect” *low-pass filter* (LPF) and a “perfect” *high-pass filter* (HPF). Simultaneously applying this pair of filters divides the signal into two parts: one part contains only low-frequency components; the other part, high-frequency components. Typically, the low-frequency part is more important, and we want to know more about its structure. We deploy a pair of filters on the low-frequency part. Note that the low-frequency part is the same as the output of the previous low-pass filter. Repeating this process, we get a detailed structure in the low frequency part; at the same time, we keep the information in the high frequency part. In a discrete case, we apply downsamplers to make the previously described scheme consistent.

Figure 3.3 illustrates the scheme described in the previous paragraph. We call this scheme a *filter bank* scheme, ostensibly from the figure (Figure 3.3). The advantages of applying filter banks are: (1) usually, the implementation of an LPF and an HPF can be very efficient; (2) if a signal is indeed made by components at a few frequencies, the filter bank scheme tends to provide an abbreviated description of the input. (The output is sparse.)

In the latter part of this section, we see that it is not necessary to restrict ourselves to the filters arrangement depicted in Figure 3.3. We can deploy a pair of filters after an HPF. This idea leads to a theory of wavelet packets.

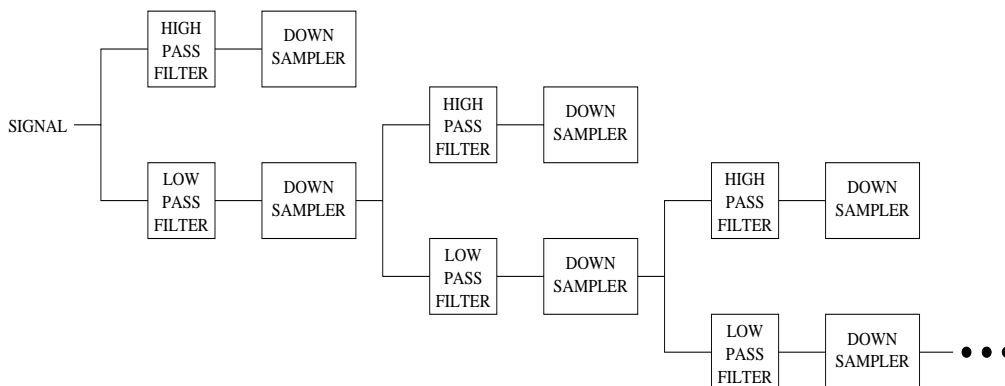


Figure 3.3: Illustration of a filter bank for forward orthonormal wavelet transform.

### 3.2.3 Discrete Algorithm

Nowadays, computers are widely used in scientific computing, and most of a signal processing job is done by a variety of chips. All chips use digital signal processing (DSP) technology. It is not an exaggeration to say that a technique is crippled if it does not have a corresponding discrete algorithm. A nice feature of wavelets is that it has a fast discrete algorithm. For length  $N$  signal, the order of computational complexity is  $O(N)$ . It can be formulated as an orthogonal transform. Each of the wavelets can have finite support.

To explain how the discrete wavelet transform (DWT) works, we need to introduce some notation. Consider a function  $f \in L^2$ . Let  $\alpha_k^j$  denote a coefficient of a transform of  $f$ ;  $\alpha_k^j$  corresponds to the  $k$ th *scaling* function at the  $j$ th scale:

$$\alpha_k^j = \int f(x)\phi(2^j x - k)dx.$$

Let  $\beta_{k'}^j$  also denote a coefficient of a transform of  $f$  corresponding to the  $k'$ th *wavelet* function at the  $j$ th scale:

$$\beta_{k'}^j = \int f(x)\psi(2^j x - k')dx.$$

Note that coefficients  $\alpha_k^j$  and  $\beta_{k'}^j$  are at the same scale. The following two relations can be

derived from the two-scale relationships in (3.21) and (3.25):

$$\begin{aligned}
\alpha_k^j &= \int f(x)\phi(2^j x - k)dx \\
&\stackrel{(3.21)}{=} \sum_{n \in \mathbb{Z}} h(n) \int f(x)\phi(2^{j+1}x - 2k - n)dx \\
&= \sum_{n \in \mathbb{Z}} h(n)\alpha_{n+2k}^{j+1};
\end{aligned} \tag{3.31}$$

$$\begin{aligned}
\beta_k^j &= \int f(x)\psi(2^j x - k)dx \\
&\stackrel{(3.25)}{=} \sum_{n \in \mathbb{Z}} g(n) \int f(x)\phi(2^{j+1}x - 2k - n)dx \\
&= \sum_{n \in \mathbb{Z}} g(n)\alpha_{n+2k}^{j+1}.
\end{aligned} \tag{3.32}$$

These two relations, (3.31) and (3.32), determine the two-scale relationship in the discrete algorithm.

Suppose that function  $f$  resides in a subspace  $\mathcal{V}_j$ :  $f \in \mathcal{V}_j$ ,  $j \in \mathbb{N}$ . By definition, there must exist a sequence of coefficients  $\alpha^j = \{\alpha_k^j, k \in \mathbb{Z}\}$ , such that

$$f = \sum_{n \in \mathbb{Z}} \alpha_k^j \phi(2^j x - k).$$

The subspace  $\mathcal{V}_j$  can be divided as

$$\mathcal{V}_j = \mathcal{V}_0 \oplus \mathcal{W}_0 \oplus \dots \oplus \mathcal{W}_{j-1}.$$

We want to know the decompositions of function  $f$  in subspaces  $\mathcal{V}_0$ ,  $\mathcal{W}_0$ ,  $\mathcal{W}_1$ ,  $\dots$ , and  $\mathcal{W}_{j-1}$ . Let  $\beta^l = \{\beta_k^l, k \in \mathbb{Z}\}$  denote the set of coefficients that correspond to the *wavelets* in the subspace  $\mathcal{W}_l$ ,  $1 \leq l < j$ ,  $l \in \mathbb{N}$ . Let  $\alpha^l = \{\alpha_k^l, k \in \mathbb{Z}\}$  denote the set of coefficients that correspond to the *scaling functions* in the subspace  $\mathcal{V}_l$ , for  $1 \leq l < j$ ,  $l \in \mathbb{N}$ . From (3.31), for any  $j \in \mathbb{N}$ , the sequence  $\alpha^{j-1}$  is a downsampled version of a convolution of two sequences  $\alpha^j$  and  $h$ :

$$\alpha^{j-1} = (h * \alpha^j) \downarrow 2. \tag{3.33}$$

Similarly, from (3.32), for any  $j \in \mathbb{N}$ , the sequence  $\beta^{j-1}$  is a downsampled version of a convolution of two sequences  $\alpha^j$  and  $g$ :

$$\alpha^{j-1} = (h * \alpha^j) \downarrow 2. \quad (3.34)$$

The algorithm for the DWT arises from a combination of equations (3.33) and (3.34). We start with  $\alpha^j$  in subspace  $\mathcal{V}_j$ ,  $j \in \mathbb{N}$ . From (3.33) and (3.34) we get sequences  $\alpha^{j-1}$  and  $\beta^{j-1}$  corresponding to decompositions in subspaces  $\mathcal{V}_{j-1}$  and  $\mathcal{W}_{j-1}$ , respectively. Then we further map the sequence  $\alpha^{j-1}$  into sequences  $\alpha^{j-2}$  and  $\beta^{j-2}$  corresponding to decompositions in subspaces  $\mathcal{V}_{j-2}$  and  $\mathcal{W}_{j-2}$ . We continue this process until it stops at scale 0. The concatenation of sets  $\alpha^0, \beta^0, \beta^1, \dots, \beta^{j-1}$  gives the decomposition into a union of all the subspaces  $\mathcal{V}_0, \mathcal{W}_0, \mathcal{W}_1, \dots, \mathcal{W}_{j-1}$ . The sequence  $\alpha^0$  corresponds to a projection of function  $f$  into subspace  $\mathcal{V}_0$ . For  $l = 0, 1, \dots, j-1$ , the sequence  $\beta^l$  corresponds to a projection of function  $f$  into subspace  $\mathcal{W}_l$ . The set  $\alpha^0 \cup \beta^0 \cup \beta^1 \cup \dots \cup \beta^{j-1}$  is called the DWT of the sequence  $\alpha^j$ .

The first graph in Figure 3.4 gives a scaled illustration of the DWT algorithm for sequence  $\alpha^3$ . Note that we assume a sequence has finite length. The length of each block is proportional to the length of its corresponding coefficient sequence, which is a sequence of  $\alpha$ 's and  $\beta$ 's. We can see that for a finite sequence, the length of the DWT is equal to the length of the original sequence. The second graph in Figure 3.4 is a non-scaled depiction for a more generic scenario.

### 3.2.4 Point Singularities

We will explain why the wavelet transform is good at processing point singularities. The key reason is that a single wavelet basis function is a time-localized function. In other words, the support of the basis function is finite. Sometimes we say that wavelet basis function has compact support. The wavelet basis is a system of functions made by dilations and translations of a single wavelet function, together with some scaling functions at the coarsest scale. In the discrete wavelet transform, for a signal concentrated at one point (or, equivalently, for one point singularity) at every scale because of the time localization property, there are only a few significant wavelet coefficients. For a length- $N$  signal, the number of scales is  $O(\log N)$ , so for a time singularity, a DWT should give no more than  $O(\log N)$  wavelet coefficients that have significant amplitudes. Compared with the length

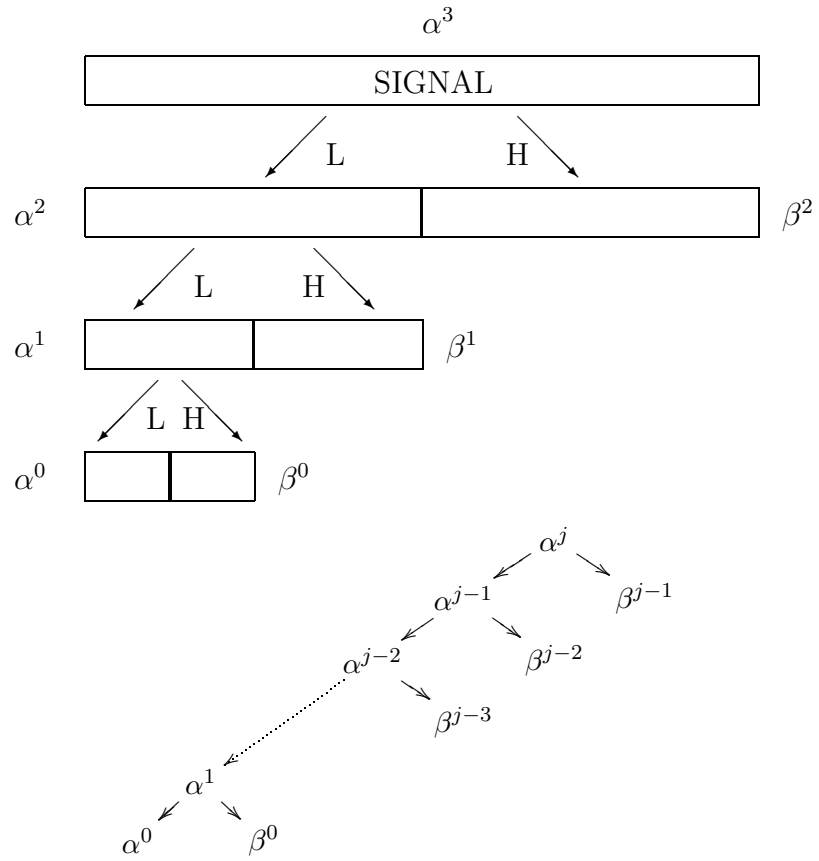


Figure 3.4: Illustration of the discrete algorithm for forward orthonormal wavelet transform on a finite-length discrete signal. The upper graph is for cases having 3 layers. The width of each block is proportional to the length of the corresponding subsequence in the discrete signal. The bottom one is a symbolic version for general cases.

of the signal (namely  $N$ ),  $O(\log N)$  can be treated as a constant. Hence, we say that the wavelet transform is good at processing point singularities.

Figure 3.5 illustrates a multiresolution analysis of a time singularity. The top left curve is a function that contains a time singularity—it is generated by a Laplacian function. The other plots in the left column show its decompositions into the coarsest scale subspace  $\mathcal{V}_0$  (made by dilation and translation of a scaling function) and the wavelet subspace  $\mathcal{W}_0$  through  $\mathcal{W}_5$ . The right column contains illustrations of the DWT coefficients corresponding to different functional subspaces at different scales. Note that at every scale, there are only a few significant coefficients. Compared with the length of the signal (which is  $N$ ), the total number of the significant DWT coefficients ( $O(\log N)$ ) is relatively small.

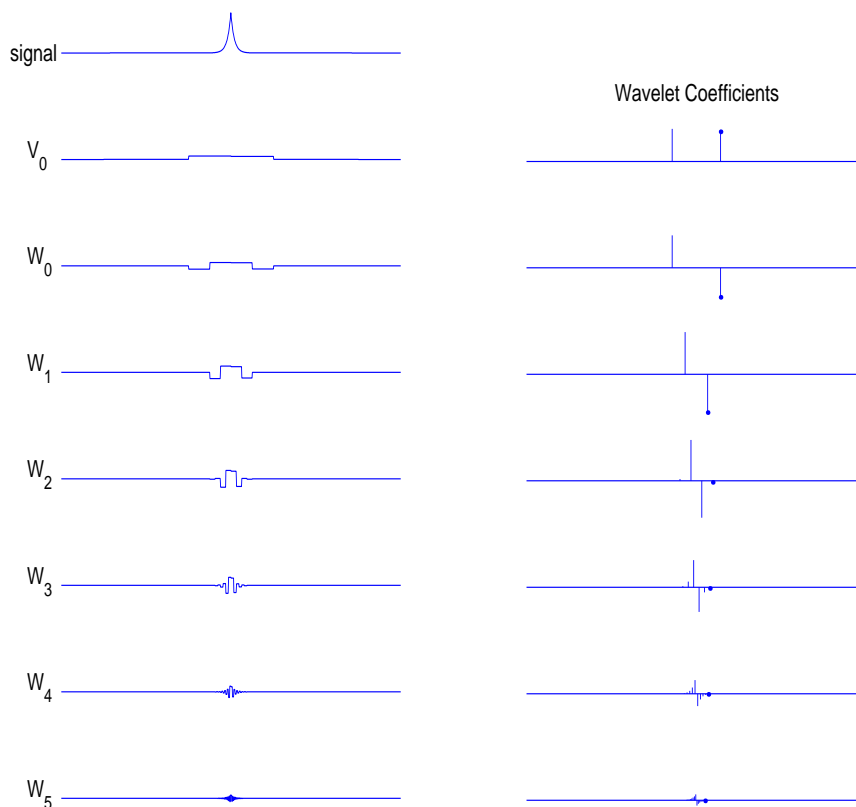


Figure 3.5: Multiresolution analysis of a point singularity with Haar wavelets.

## 2-D Wavelets

The conclusion that the 2-D wavelet is good at processing point singularities in an image is a corollary of the 1-D result.

We consider a 2-D wavelet as a tensor product of two 1-D wavelets. It is easy to observe that the 2-D wavelet is a spatially localized function. Utilizing the idea from the filter bank and multi-resolution analysis, we can derive a fast algorithm for 2-D wavelet transform. The fast algorithm is also based on a two-scale relationship.

Using a similar argument as in the 1-D case, we can claim that for an  $N \times N$  image, no more than  $O(\log N)$  significant 2-D wavelet coefficients are needed to represent a point singularity in an image. Hence *the wavelet transform is good at processing point singularities in images.*

Figure 3.6 shows some 2-D wavelet basis functions. We see that wavelet functions look like points in the image domain.

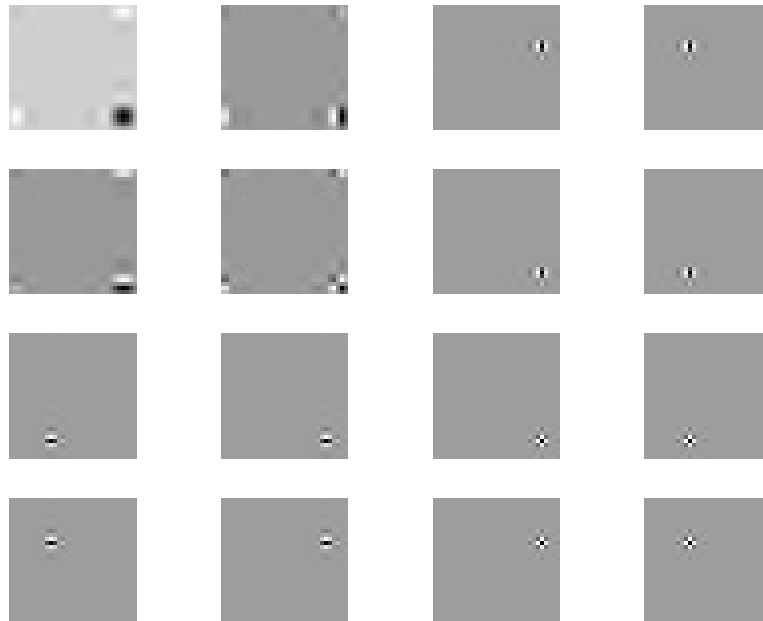


Figure 3.6: Two-dimensional wavelet basis functions. These are  $32 \times 32$  images. The upper left one is a tensor product of two scaling functions. The bottom right 2 by 2 images and the (2,2)th image are tensor products of two wavelets. The remaining images are tensor products of a scaling function with a wavelet.

### 3.3 Edgelets and Linear Singularities

We describe the edgelet system and its associated transforms. They are designed for linear features in images.

Detailed description of edgelet system, edgelet transform and fast approximate edgelet transform are topics of Appendix A and Appendix B.

#### 3.3.1 Edgelet System

The edgelet system, defined in [50], is a finite dyadically-organized collection of line segments in the unit square, occupying a range of dyadic locations and scales, and occurring at a range of orientations. It is a low cardinality system. This system has a nice property: any line segment in an image (in the system or not) can be approximated *well* by *a few* elements from this system. More precisely, it takes at most  $8 \log_2(N)$  edgelets to approximate any line segment within distance  $1/N + \delta$ , where  $N$  is the size of the image and  $\delta$  is a constant. The so-called edgelet transform takes integrals along these line segments.

The edgelet system is constructed as follows:

- [E1] Partition the unit square into dyadic sub-squares. The sides of subsquares are  $1/2, 1/4, 1/8, \dots$
- [E2] On each subsquare, put equally-spaced vertices on the boundary. The inter-distance is prefixed and dyadic.
- [E3] If a line segment connecting any two vertices is not on a boundary, then it is an *edgelet*.
- [E4] The edgelet system is a collection of all the edgelets as defined in [E3].

The size of the edgelet system is  $O(N^2 \log_2 N)$ .

#### 3.3.2 Edgelet Transform

Edgelets are not functions and do not make a basis; instead, they can be viewed as geometric objects—line segments in the square. We can associate these line segments with linear functionals: for a line segment  $e$  and a smooth function  $f(x_1, x_2)$ , let  $e[f] = \int_e f$ . Then the edgelet transform can be defined as the mapping:  $f \rightarrow \{e[f] : e \in \mathcal{E}_n\}$ , where  $\mathcal{E}_n$  is the collection of edgelets, and  $e[f]$  is, as above, the linear functional  $\int_e f$ .

Some examples of the edgelet transform on images are given in Appendix A.

### 3.3.3 Edgelet Features

Suppose we have a digital image  $I$  and we calculate  $f = \varphi * I$ , the filtering of image  $I$  by  $\varphi$  according to semi-discrete convolution:  $f(x_1, x_2) = \sum_{k_1, k_2} \varphi(x_1 - k_1, x_2 - k_2) I_{k_1, k_2}$ . Assuming  $\varphi$  is smooth, so is  $f$ , and we may define the collection of *edgelet features* of  $I$  via  $T[I] = \{e[f] : e \in \mathcal{E}_n\}$ . That is, the edgelet transform of the smoothed image  $f = \varphi * I$ . In terms of the unsmoothed image  $\tilde{I} = \sum_{k_1, k_2} I_{k_1, k_2} \delta_{k_1, k_2}$  with  $\delta_{k_1, k_2}$  the Dirac mass, we may write  $(T[I])_e = \int \psi_e(x_1, x_2) \tilde{I}(x_1, x_2) dx_1 dx_2$ , where  $\psi_e(x_1, x_2) = \int_0^1 \varphi(x_1 - e_1(t), x_2 - e_2(t)) dt$ , where  $t \rightarrow (e_1(t), e_2(t))$  is a unit-speed parameterized path along edgelet  $e$ . In short,  $\psi_e$  is a kind of “smoothed-out edgelet” with smoothing  $\varphi$ .

Since edgelets are features with various locations, scales and orientations, the associated transform—the edgelet transform—is well-suited to processing linear features.

### 3.3.4 Fast Approximate Edgelet Transform

The edgelet transform (defined in previous sections) is an  $O(N^3 \log_2 N)$  algorithm. The order of complexity can be reduced to  $O(N^2 \log_2 N)$ , if we allow a slight modification of the original edgelet system.

Recently, a fast algorithm for calculating a discrete version of the Radon transform has been developed. The Radon transform of a 2-D continuous function is simply a projection of the 2-D function onto lines passing through the origin. In discrete case, people can utilize the idea from the Fourier Slice Theorem (see Appendix B) to design a fast algorithm. It takes the following three steps:

$$\text{image} \rightarrow \text{2-D FFT} \rightarrow \text{X-interpolation} \rightarrow \text{1-D inverse FFT} \rightarrow \text{output}$$

(using fractional FT)

The so-called *fast approximate edgelet transform* is essentially the discrete Radon transform of an image at various scales: we partition a unit square into subsquares having various widths, and then apply the discrete version of the Radon transform on images limited within each of these subsquares. This method was described in [48] and will be described in detail in Appendix B. As we can see, the existence of this fast algorithm relies on the existence of FFT and fractional FT.

The Riesz representers of the fast approximate edgelet transform are close to edgelets but not exactly the same. Illustrations of some of these representers and some examples

of this transform on images are given in Appendix B, together with detailed discussion on algorithm design issues and analysis.

## 3.4 Other Transforms

In this section, we review some other widely known transforms. The first subsection is for one-dimensional (1-D) transforms. The second subsection is for 2-D transforms. We intend to be comprehensive, but sacrifice depth and mathematical rigor. The motivation for writing this section is to provide us with some starting points in this field.

### 3.4.1 Transforms for 1-D Signals

We briefly describe some 1-D transforms. Note that all the transforms we mention are decomposition schemes.

As mentioned at the beginning of this chapter, there is another popular idea in designing the 1-D transform. This idea is to map the 1-D signal into a 2-D function, which is sometimes called a *distribution*. This idea is called the idea of *distribution*. Typically the mapping is a bilinear transform; for example, the Wigner-Ville distribution. In this thesis, we are *not* going to talk about the distribution idea. Readers can learn more about it in Flandrin's book [68].

We focus on the idea of *time-frequency decomposition*. Suppose every signal is a superposition of some elementary atoms. Each of these elementary atoms is associated with a 2-D function on a plane, in which the horizontal axis indicates the time and the vertical axis indicates the frequency. This two-dimensional plane is called a *time frequency plane* (TFP). The 2-D function is called a *time frequency representation* (TFR) of the atom. Usually the TFR of an atom is a bilinear transform of the atom. Accordingly, the same transform of a signal is the TFR of the signal. We pick atoms such that the TFR of every atom resides in a small cell on the TFP. We further assume that these previously mentioned cells cover the entire TFP. The collection of these cells is called a *time frequency tiling*. When a signal is decomposed as a linear combination of these atoms, the decomposition can also be viewed as a decomposition of a function on the TFP, so we can call the decomposition a time-frequency decomposition.

Based on Heisenberg's uncertainty principle, these cells, which correspond to the elementary atoms, cannot be too small. There are many ways to present the Heisenberg's

uncertainty principle. One way is to say that for any signal (here a signal corresponds to a function in  $L^2$ ), the multiplication of the variances in both the time domain and the frequency domain are lower bounded by a constant. From a time-frequency decomposition point of view, the previous statement means that the area of a cell for each atom can *not* be smaller than a certain constant. A further result from Slepian, Pollak and Landau claims that a signal can never have both finite time duration and finite bandwidth simultaneously. For a careful description about them, readers are referred to [68].

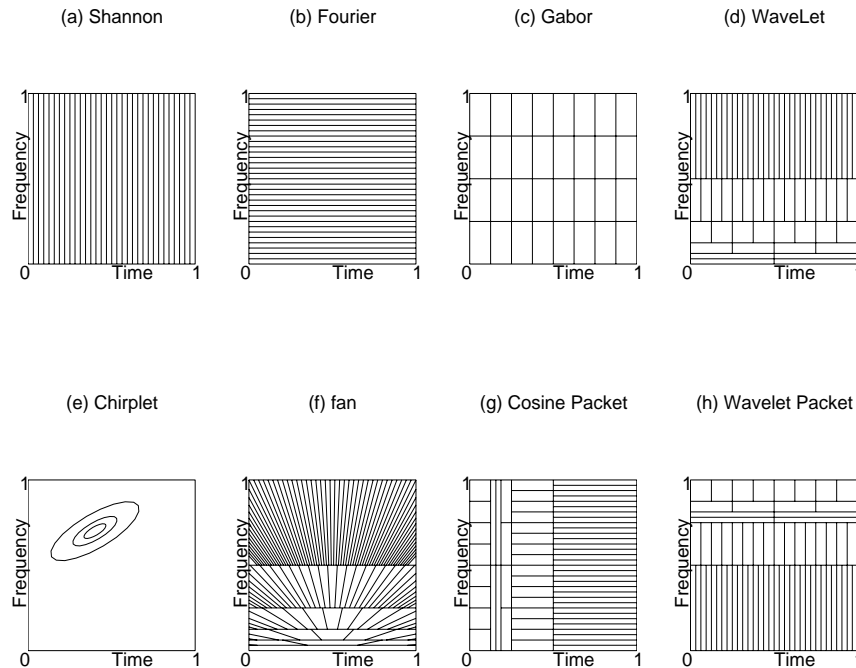


Figure 3.7: Idealized tiling on the time-frequency plane for (a) sampling in time domain (Shannon), (b) Fourier transform, (c) Gabor analysis, (d) orthogonal wavelet transform, (e) chirplet, (f) orthonormal fan basis, (g) cosine packets, and (h) wavelet packets.

The way to choose these cells determines the time-frequency decomposition scheme. Two basic principles are generally followed: (1) the tiling on TFP does not overlap, and (2) the union of these cells cover the entire TFP.

Time-frequency decomposition is a powerful idea, because we can idealize different transforms as different methods of tiling on the TFP. Figure 3.7 gives a pictorial tour of idealized tiling for some transforms.

In the remainder of this subsection, we introduce four transforms: Gabor, chirplet, cosine packets and wavelet packets. They are selected because of their importance in the literature. (Of course the selection might be biased by personal preference.)

### Gabor Analysis

The Gabor transform was first introduced in 1946 [69]. A contemporary description of the Gabor transform can be found in many books, for example, [68] and [100]. For a 1-D continuous function  $f(t)$  and a continuous time variable  $t$ , the Gabor transform is

$$G(m, n) = \int_{-\infty}^{+\infty} f(t) \cdot h(t - mT) e^{-i\omega_0 n t} dt, \quad \text{for integer } m, n \in \mathbb{Z}, \quad (3.35)$$

where  $h$  is a function with finite support, also called a *window function*. The constants  $T$  and  $\omega_0$  are sampling rates for time and frequency. A function

$$b_{m,n}(t) = h(t - mT) e^{-i\omega_0 n t} \quad (3.36)$$

is called a basis function of Gabor analysis. (Of course we assume that the set  $\{b_{m,n} : m, n \in \mathbb{Z}\}$  does form a basis.) Function  $b_{m,n}(t)$  should be both time and frequency localized.

A basis function of the Gabor transform is a shifted version of one initializing function. The shifting is operated in both time (by  $mT$ ) and frequency (by  $n\omega_0$ ) domains, so the Gabor transform can be compared to partitioning the TFP into rectangles that have the same width and height. Figure 3.7 (c) depicts an idealized tiling corresponding to Gabor analysis.

Choices of  $T$  and  $\omega_0$  determine the resolution of a partitioning. The choice of the window function  $h$ , which appears in both (3.35) and (3.36), determines how concentrated (in both time and frequency) a basis function  $b_{m,n}$  is. Two theoretical results are noteworthy:

- If a product  $T \cdot \omega_0$  is greater than  $2\pi$ , then a set  $\{h(t - mT) e^{-i\omega_0 n t} : m, n \in \mathbb{Z}\}$  can never be a frame for  $L^2(\mathbb{R})$  (Daubechies [34], Section 4.1).
- When  $T \cdot \omega_0 = 2\pi$ , if a set  $\{h(t - mT) e^{-i\omega_0 n t} : m, n \in \mathbb{Z}\}$  is a frame for  $L^2(\mathbb{R})$ , then either  $\int x^2 |g(x)|^2 dx = \infty$  or  $\int \xi^2 |\hat{g}(\xi)|^2 d\xi = \infty$  (Balian-Low [34]).

Gabor analysis is a powerful tool. It has become a foundation for the joint time frequency analysis.

### Chirplets

A chirp is a signal whose instantaneous frequency is a linear function of the time. A detailed and concise definition of instantaneous frequency is in Flandrin [68]. A chirplet can be viewed as a piece of a chirp, or equivalently a windowized version of a chirp. The following is an example of a chirplet function:

$$h(t - nT)e^{iq(t)}, \quad (3.37)$$

where  $h$  is still a window function,  $T$  is the time sampling rate, and  $q(t)$  is a quadratic polynomial of  $t$ . An example of  $h$  is the Gaussian function  $h(t) = e^{-t^2}$ . Note  $h$  is a real function. Suppose the general form for the quadratic polynomial is  $q(t) = a_2t^2 + a_1t + a_0$ . The instantaneous frequency is (in this case)  $q'(t) = 2a_2t + a_1$ , which is linear in  $t$ .

On the TFP, a support of a chirplet is a needle-like atom. Figure 3.7(e) shows a chirplet. An early reference on this subject is [109]. A more general but complicated way to formulate chirplets is described in [103]. An idealistic description of the chirplet formulation in [103] is that we consider the chirplet as a quadrilateral on the time frequency plane. Since each corner of the quadrilateral has two degrees of freedom, the degrees of freedom under this formulation can go up to 8. A signal associated with this quadrilateral is actually a transform of a signal associated with a rectangle in TFP; for example, the original signal could be a Gabor basis function. There are eight allowed transforms: (1) translation in time, (2) dilation in time, (3) translation in frequency, also called modulation, (4) dilation in frequency, (5) shear in time (Fourier transformation, followed by a multiplication by a chirp, then an inverse Fourier transformation), (6) shear in frequency (multiplication by a chirp), (7) perspective projection in the time domain, and finally (8) perspective projection in the frequency domain.

Generally, a chirplet system is overcomplete. The following is about how to build a chirplet-like orthonormal basis. It is instructive to see that a new orthonormal basis can be built out of an existing orthonormal basis. The following idea is described in [7]. They introduced an operator called *axis warping*: if  $f(x)$  is a function in  $L^2(\mathbb{R})$ , the axis warping of function  $f(x)$  is

$$(\mathcal{W}_c f)(x) = |c|^{1/2} |x|^{(c-1)/2} f(|x|^c \text{sign}(x)),$$

where the constant  $c$  is a positive real number. It can be proven that a transform  $\mathcal{W}_c$ , for  $c > 0$ , is isometric in  $L^2(\mathbb{R})$ . We also know that the Fourier transform is isometric in  $L^2(\mathbb{R})$  (Parseval). Suppose we have an orthonormal wavelet basis  $W = \{w_{j,k} : j, k \in \mathbb{Z}\}$ . We first apply a Fourier transform on the basis  $W$ , then apply an axis warping transform, then apply the inverse Fourier transform. Because each step is an isometric transform, the result must be another orthonormal basis in  $L^2(\mathbb{R})$ . Thus we get a new orthonormal basis, called a *fan* basis in [7]. Figure 3.7(d) gives an idealized tiling of wavelets on the time frequency plane. Figure 3.7(f) gives an idealized tiling of the fan basis.

### Cosine Packets

Here we describe cosine packets. We review some landmarks in the historic development of time frequency localized orthonormal basis. Real-valued time frequency localized atoms not only have some intuitive optimality, but also have been the tool to circumvent the barrier brought by traditional Gabor analysis. The content of this subsection is more abundant than its title suggests.

*Collection of localized cosine and sine functions.* Cosine packets are a collection of localized cosine and sine functions. A readily observable optimality of cosine and sine functions is that they are real-valued. Typically, an analyzed signal is a real-valued signal, so localized cosine and sine functions are closer to a signal than these complex-valued functions do. We also want a basis function to be localized. The localization should be in both time and frequency domains: in both time and frequency (Fourier) domain, a function must either have a finite support, or decays faster than any inverse of polynomials.

*Orthonormal basis for  $L^2(\mathbb{R})$ .* In Gabor analysis, finding an orthonormal basis for  $L^2(\mathbb{R})$  was a topic that has been intensively studied. The Balian-Low theorem says that if we choose a time-frequency atom following (3.36), then the atom cannot be simultaneously localized in both time domain and frequency domain—the atom must have infinite variance either in time or in frequency. To circumvent this barrier, Wilson in 1987 [34, page 120] proposed a basis function that has two peaks in frequency. In [34], Daubechies gives a construction of an orthonormal basis for  $L^2(\mathbb{R})$ . In her construction, basis functions have exponential decay in both time and frequency. From [34], *the key idea to ideal time-frequency localization and orthonormality in the windowed Fourier framework is to use sine and cosine rather than complex exponentials.* This is a not-so-obvious optimality of using sine and cosine functions in basis.

*Folding.* Another way to obtain localized time-frequency basis, instead of using the constructive method in Daubechies [34], is to apply folding to an existing orthonormal basis for periodic functions on a fixed interval. The idea of folding is described in [141] and [142]. The folding operation is particularly important in discrete algorithms because if the basis function is from the folding of a known orthonormal basis function (e.g., Fourier basis function), since the coefficient of the transform associated with the basis is simply the inner product of the signal with the basis function, we can apply the adjoint of the folding operator to the signal, then calculate its inner product with the original basis function. If there is a fast algorithm to do the original transform (for example, for DFT, there is a FFT), then there is a fast algorithm to do the transform associated with the new time-frequency atoms.

*Overcomplete dictionary.* For a fixed real constant, we can construct an orthonormal basis whose support of basis functions has length exactly equal to this constant. But we can choose different lengths for support. In the discrete case, for the simplicity of implementation, we usually choose the length of support equal to a power of 2. The collection of all the basis functions make an overcomplete dictionary. A consequential question is how to find a subset of this dictionary. The subset should construct an orthonormal basis, and, at the same time, be the best representation for a particular class of signal.

*Best orthogonal basis.* The best orthogonal basis (BOB) algorithm is proposed to achieve the objective just mentioned. The key idea is to assign a probability distribution on a class of signals. Usually, the class of signals is a subspace of  $L^2(\mathbb{R})$ . Without loss of generality, we consider signals that are restricted on the interval  $[0, 1)$ . Suppose we consider only functions whose support are dyadic intervals that have forms  $[2^{-l}(k-1), 2^{-l}k)$ , for  $l \in \mathbb{N}, k = 1, 2, \dots, 2^l$ . At each level  $l$ , for the basis functions whose support are intervals like  $[2^{-l}(k-1), 2^{-l}k)$ , the associated coefficients can be calculated. We can compute the entropy of these coefficients. For an interval  $[2^{-l}(k-1), 2^{-l}k)$ , we can consider its immediate two subintervals:  $[2^{-l-1}(2k-2), 2^{-l-1}(2k-1))$  and  $[2^{-l-1}(2k-1), 2^{-l-1}2k)$ . (Note  $[2^{-l}(k-1), 2^{-l}k) = [2^{-l-1}(2k-2), 2^{-l-1}(2k-1)) \cup [2^{-l-1}(2k-1), 2^{-l-1}2k)$ .) Note in the discrete case, a set of the coefficients associated with an interval  $[2^{-l}(k-1), 2^{-l}k)$  and a set of coefficients associated with intervals  $[2^{-l-1}(2k-2), 2^{-l-1}(2k-1))$  and  $[2^{-l-1}(2k-1), 2^{-l-1}2k)$  have the same cardinality. In fact, there is a binary tree structure. Each dyadic interval corresponds to a node in the tree. Suppose a node in the tree corresponds to an interval  $I$ . Two subsequent nodes in the tree should correspond to the two subintervals of the interval  $I$ .

The idea of the BOB is the following: if the entropy associated with the two subintervals is lower than the entropy associated with the larger dyadic interval, then we choose to divide the larger dyadic interval. This process is repeated until there is no more partitioning to do. The final partition of the unit interval  $[0, 1)$  determines a subset of localized cosine functions. It's not hard to observe that this subset of local cosine functions makes an orthonormal basis. For a given signal, instead of computing the entropy of a statistical distribution, we can compute the *empirical entropy*, which is just the entropy of a set of coefficients. Note that the coefficients are from a transform of the signal. Based on this empirical entropy, we can use the BOB algorithm to choose a subset of coefficients, and correspondingly a subset of basis functions that form a basis.

*Complexity.* Suppose the length of the signal is  $N$ , for  $N \in \mathbb{N}$ . Note that when we take cosine functions as an orthonormal basis over an interval, the “folding” transform gives another orthonormal system whose elements are functions that are defined on the entire real axis, but are localized (have finite support). For every dyadic interval, we have such an orthonormal system. Combining all these systems, we actually get the cosine packets (CP). Suppose dyadic intervals having the same length is a cover of the entire real axis. We say that CP elements associated with these intervals are at the same *scale*. For length- $N$  signals, we have  $\log_2 N$  scales. To calculate the CP coefficients at one scale, we can first apply “folding” to the data, then apply a DCT. The folding is an  $O(N)$  operation. The discrete cosine transform has  $O(N \log N)$  complexity. Since we have  $\log_2 N$  scales, the total complexity of computing CP coefficients is  $O(N \log^2 N)$ .

### Wavelet Packets

In MRA, instead of dividing the low-frequency part, or a scaling space (a space spanned by scaling functions), we can apply a quadratic mirror filter to divide the high-frequency part, or a wavelet space (which is spanned by the wavelet functions). This idea unveils developments of wavelet packets. Equivalently, in the filter banks algorithm, at each step, instead of deploying a pair of filters after an LPF, we can deploy a pair of filters after an HPF. By doing this, we partition the high-frequency part. Since a wavelet space contains high-frequency components, partition in a high-frequency part is equivalent to partition in a wavelet space.

In every step, we can adaptively choose to deploy a pair of filters either after an LPF or an HPF. In this framework, all possible sets of coefficients have a binary tree structure.

We can apply the idea of BOB, which is described in the previous subsection, to select an orthonormal basis.

*Computational complexity.* Computation of the coefficients at every step involves merely filtering. The complexity of filtering a length- $N$  signal with a finite-length filter is no more than  $O(N)$ , so in each step, the complexity is  $O(N)$ . Since the wavelet packets can go from scale 1 down to scale  $\log_2 N$ . The total complexity of calculating all possible wavelet packets coefficients is  $O(N \log_2 N)$ .

### 3.4.2 Transforms for 2-D Images

To capture some *key* features in an image, different transforms have been developed. The main idea of these efforts is to construct a basis, or frame, such that the basis functions, or the frame elements, have the interesting features. Some examples of the interesting features are anisotropy, directional sensitivity, etc.

In the remainder of this subsection, we introduce brushlets and ridgelets.

#### Brushlets

The brushlet is described in [105]. A key idea is to construct a windowized smooth orthonormal basis in the frequency domain; its correspondent in the time domain is called *brushlets*. By constructing a “perfectly” localized basis in the frequency domain, if an original signal has a peak in the frequency domain, then the constructed basis, brushlets, tends to capture this feature. A 2-D brushlet is a tensor product of two 1-D brushlets. A nice property of a 2-D brushlet is that it is an anisotropic, directionally sensitive, and spatially localized basis. More details are in Meyers’ original paper [105].

#### Ridgelets

A ridgelet system is a system designed to process a high-dimensional signal that is a superposition of some linear singularities. In image analysis, linear singularities can be considered as edges. It is known that edge features are important features of an image.

The term *ridgelet* was first coined by Candès [21] [22]. A ridge function is a function that is constant over a hyper-plane. For example, function

$$f(u \cdot x - b)$$

is a ridge function, where  $u$  denotes a constant vector,  $b$  denotes a scalar constant, and  $u \cdot x$  is the inner product of vector  $u$  and vector  $x$ . This idea is originally from *neural networks*. When a function  $f$  is a *sigmoid* function, the function  $f(u \cdot x - b)$  is a single *neuron* in neural networks. For Candès' ridgelets, a key idea is to cleverly choose some conditions for  $f$ ; for example, an admissibility condition, so that when a function  $f$  satisfies these conditions, we **not only** are able to have a continuous representation based on ridge functions that have the form  $f(\text{a linear term})$ , **but also** can construct a frame for a compact-supported square-integrable functional space. The latter is based on carefully choosing a spatial discretization on the  $(u, b)$  plane. Note that a compact-supported square-integrable functional space should be a subspace of  $L^2(\mathbb{R})$ . Candès [21] proves that his ridgelet system is optimal in approximating a class of functions that are *merely* superpositions of linear singularities. A function that is a superposition of linear singularities is a function that is smooth everywhere except on a few lines. A 2-D example of this kind of function is a half dome: for  $x = (x_1, x_2) \in \mathbb{R}^2$ ,  $f(x) = 1_{\{x_1 > 0\}} e^{-x_1^2 - x_2^2}$ . More detailed discussion is in [21].

A ridge function is generally not in  $L^2(\mathbb{R}^2)$ , so in  $L^2(\mathbb{R}^2)$ , Candès's system cannot be a frame, or a basis. Donoho [51] constructs another system, called orthonormal ridgelets. In Donoho's ridgelet system, a basic element is an angularly-integrated ridge function. He proves that his ridgelet system is an orthonormal basis in  $L^2(\mathbb{R}^2)$ . The efficiency of using Donoho's ridgelet basis to approximate a ridge function is explored in [52]. In discrete cases (this is for digital signal processing), a fast algorithm to implement a *quasi*-ridgelet transform for digital images is proposed in [49]. This algorithm is based on a fast Cartesian to polar coordinate transform. The computational complexity (for an  $N \times N$  image) is  $O(N^2 \log N)$ .

### 3.5 Discussion

In the design of different transforms, based on what we have seen, the following three principles are usually followed:

- *Feature capture.* Suppose  $\mathbf{T}$  is a transform operator. We can think of  $\mathbf{T}$  as a linear operator in a function space  $\Omega$ , such that for a function  $f \in \Omega$ ,  $\mathbf{T}(f)$  is a set of coefficients,  $\mathbf{T}(f) = \{c_\alpha, \alpha \in \mathcal{I}\}$ . Furthermore, we impose that  $f = \sum_{\alpha \in \mathcal{I}} c_\alpha b_\alpha$ , where  $b_\alpha$ 's are the basis functions. If basis functions of  $\mathbf{T}$  reflect features that we try to catch, and if a signal  $f$  is just a superposition of a few features, then the transform  $\mathbf{T}$  should

give only a few coefficients, so the result should be sparse. When this happens, we say that the transform  $\mathbf{T}$  has captured features in the signal  $f$ . Hence the transform  $\mathbf{T}$  is ideal at processing the signal  $f$ .

In this chapter, we have presented graphs of different basis functions for different transforms. We intend to show what kind of features these transforms may capture.

- *Fast algorithm.* Since most of contemporary signal processing is done in digital format, a continuous transform is unlikely to make a strong impact unless it has a fast discrete algorithm. Some examples are the continuous Fourier transform and the continuous wavelet transform. Both of them have fast discrete correspondents. A fast algorithm usually means that the complexity should not be higher than a product of the original size (which typically is  $N$  for a 1-D signal and  $N^2$  for a 2-D  $N$  by  $N$  image) and  $\log N$  (or a polynomial of  $\log N$ ).
- *Asymptotic optimality.* One way to quantify the optimality of a basis in a certain space is to consider its asymptotics. This type of research has been reported in various papers, for example [46]. The idea is to find the asymptotic exponent of a minimax risk in an  $l^2$  sense. Here is the idea. Suppose a functional space that we are interested in is  $\Omega$ . A function  $d \in \Omega$  is what we want to approximate, or estimate. For a fixed basis (or a dictionary, or a frame) that is denoted by  $B$ , let  $d_N$  be a superposition of no more than  $N$  elements from  $B$ , and at the same time,  $d_N$  minimizes the mean square error risk  $\|d - d_N\|_2^2$ . Let's consider the minimax risk, which is

$$\tau_N = \sup_{d \in \Omega} \inf_{d_N} \|d - d_N\|_2^2.$$

If  $\tau_N$  has an asymptotic exponent  $p$ ,  $\tau_N \asymp N^p$ , then we call  $p$  the asymptotic exponent of  $B$  in  $\Omega$ . Note that the exponent  $p$  is always negative. The smaller (more negative) the asymptotic exponent  $p$  is, the more efficient the basis  $B$  is in approximating functions in the space  $\Omega$ . In this sense, we say the asymptotic exponent measures the optimality of basis  $B$ .

In designing a transform, which is equivalent to specifying the set  $B$ , we always want to achieve a small  $p$ .

### 3.6 Conclusion

We have reviewed some transforms. A key point is that none of these transform is ideal for processing images with multiple features. In reality, an image tends to have multiple features.

Two consequential questions are whether and how we can combine multiple transforms, so that we can simultaneously take advantage of all of them. The remainder of this thesis will try to answer these questions.

Note that since images are typically very large, efficient numerical algorithms are crucial for determining if a scheme is successful or not. Readers may notice that the remainder of this thesis is very computationally oriented.

### 3.7 Proofs

#### Proof of Theorem 3.1

Suppose  $a_{mn}$  is the  $mn$ -th component of the matrix  $C$ . The proof is based on the following fact:

$$\begin{aligned} \sum_{k=0}^{N-1} a_{mk} \frac{1}{\sqrt{N}} e^{-i\frac{2\pi}{N}kl} \quad (C \text{ is a circulant matrix}) &= \sum_{k=0}^{N-1} c_k \frac{1}{\sqrt{N}} e^{-i\frac{2\pi}{N}(m+k)l} \\ &= \frac{1}{\sqrt{N}} e^{-i\frac{2\pi}{N}ml} \left( \sum_{k=0}^{N-1} c_k e^{-i\frac{2\pi}{N}kl} \right), \end{aligned}$$

where  $0 \leq m, l \leq N - 1$  and  $i^2 = -1$ . Based on this, one can easily verify that the Fourier series  $\{e^{-i\frac{2\pi}{N}kl} : k = 0, 1, 2, \dots, N - 1\}$ , for  $l = 0, 1, 2, \dots, N - 1$ , are the eigenvectors of the matrix  $C$  and the Fourier transform of the sequence  $\{\sqrt{N}c_0, \sqrt{N}c_1, \dots, \sqrt{N}c_{N-1}\}$  are the eigenvalues.

#### Proof of Theorem 3.5

Suppose the covariance matrix  $\Sigma$  can be diagonalized by a type of DCT. If the corresponding DCT matrix is  $C_N^\bullet$ , then we have  $C_N^\bullet \Sigma (C_N^\bullet)^T = \Omega_N$ , where  $\Omega_N$  is a diagonal matrix  $\Omega_N = \text{diag}\{\lambda_0, \lambda_1, \dots, \lambda_{N-1}\}$ . Equivalently, we have  $\Sigma = (C_N^\bullet)^T \Omega_N C_N^\bullet$ . From (3.11),

the  $ij$ th component of matrix  $\Sigma$  is

$$\begin{aligned}\Sigma_{ij} &= \sum_{k=0}^{N-1} \lambda_k \alpha_1^2(k) \alpha_2(i) \alpha_2(j) \frac{2}{N} \cos\left(\frac{\pi(k+\delta_1)(i+\delta_2)}{N}\right) \cos\left(\frac{\pi(k+\delta_1)(j+\delta_2)}{N}\right) \\ &= \alpha_2(i) \alpha_2(j) \frac{1}{N} \sum_{k=0}^{N-1} \lambda_k \alpha_1^2(k) \left[ \cos\frac{\pi(k+\delta_1)(i-j)}{N} + \cos\frac{\pi(k+\delta_1)(i+j+2\delta_2)}{N} \right].\end{aligned}$$

In the above equation, let the first term  $\frac{1}{N} \sum_{k=0}^{N-1} \lambda_k \alpha_1^2(k) \cos\frac{\pi(k+\delta_1)(i-j)}{N}$  be the  $ij$ th element of matrix  $\Sigma_1$ , and let the second term  $\frac{1}{N} \sum_{k=0}^{N-1} \lambda_k \alpha_1^2(k) \cos\frac{\pi(k+\delta_1)(i+j+2\delta_2)}{N}$  be the  $ij$ th element of matrix  $\Sigma_2$ . It is easy to see that the matrix  $\Sigma_1$  is Toeplitz, and the matrix  $\Sigma_2$  is Hankel. Inserting different values of  $\delta_1$  and  $\delta_2$ , we establish the Theorem 3.5.

### Proof of Theorem 3.6

Consider the  $z$ -transform of sequence  $\{h(n) : n \in \mathbb{Z}\}$  and sequence  $\{g(n) : n \in \mathbb{Z}\}$ :

$$\begin{aligned}H(z) &= \sum_{n \in \mathbb{Z}} h(n) z^n, \\ G(z) &= \sum_{n \in \mathbb{Z}} g(n) z^n.\end{aligned}$$

From (3.27), note that all the *even* terms in  $H(z)H(z^{-1})$  have zero coefficient except the constant term. Hence we have

$$H(z)H(z^{-1}) + H(-z)H(-z^{-1}) = 2. \quad (3.38)$$

Similarly from (3.28) we have

$$G(z)G(z^{-1}) + G(-z)G(-z^{-1}) = 2. \quad (3.39)$$

From (3.29), all the *even* terms in  $H(z)G(z^{-1})$  have zero coefficients, so we have

$$G(z)H(z^{-1}) + G(-z)H(-z^{-1}) = 0. \quad (3.40)$$

Equations (3.38) (3.39) and (3.40) together are equivalent to the matrix multiplication

$$\begin{pmatrix} H(z) & H(-z) \\ G(z) & G(-z) \end{pmatrix} \begin{pmatrix} H(z^{-1}) & H(-z^{-1}) \\ G(z^{-1}) & G(-z^{-1}) \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

Taking the determinant of both sides, we have

$$[H(z)G(-z) - G(z)H(-z)][H(z^{-1})G(-z^{-1}) - G(z^{-1})H(-z^{-1})] = 4. \quad (3.41)$$

If both sequence  $\{h(n) : n \in \mathbb{Z}\}$  and sequence  $\{g(n) : n \in \mathbb{Z}\}$  have finite length (thinking of the impulse responses of FIR filters), then the polynomial  $H(z)G(-z) - G(z)H(-z)$  must have only one nonzero term. Since if the polynomial  $H(z)G(-z) - G(z)H(-z)$  has more than two terms and simultaneously has finite length, (3.41) can never be equal to a constant.

On the other hand, we have

$$\begin{aligned} & [H(z^{-1}) + H(-z^{-1})][H(z)G(-z) - G(z)H(-z)] \\ = & H(z^{-1})H(z)G(-z) + H(-z^{-1})H(z)G(-z) \\ & - H(z^{-1})G(z)H(-z) - H(-z^{-1})G(z)H(-z) \\ \stackrel{(3.40)}{=} & H(z^{-1})H(z)G(-z) - H(z^{-1})H(z)G(z) \\ & + H(-z^{-1})G(-z)H(-z) - H(-z^{-1})G(z)H(-z) \\ \stackrel{(3.38)}{=} & 2[G(-z) - G(z)]. \end{aligned}$$

We know that  $H(z)G(-z) - G(z)H(-z)$  only has one nonzero term. The polynomial  $H(z^{-1}) + H(-z^{-1})$  can only have terms with even exponents and polynomial  $G(-z) - G(z)$  can only have terms with odd exponents, so from the previous equation, there must exist an integer  $k$ , such that

$$H(z)G(-z) - G(z)H(-z) = 2z^{2k+1},$$

and

$$[H(z^{-1}) + H(-z^{-1})]z^{2k+1} = G(-z) - G(z). \quad (3.42)$$

Similarly, we have

$$[H(z^{-1}) - H(-z^{-1})][H(z)G(-z) - G(z)H(-z)] = 2[G(-z) + G(z)].$$

Consequently,

$$[H(z^{-1}) - H(-z^{-1})]z^{2k+1} = G(-z) + G(z). \quad (3.43)$$

From (3.42) and (3.43), we have

$$G(z) = -z^{2k+1}H(-z^{-1}).$$

The above equation is equivalent to relation (3.30) in the Theorem.



## Chapter 4

# Combined Image Representation

This chapter is about combined image representation and sparse decomposition. First, in Section 4.1, we discuss the motivation for using combined image representation, the key being that we can obtain benefits from different image transforms. Because we have combined representations, we have an overcomplete system, or an overcomplete dictionary. Section 4.2 surveys research developments in finding sparse decompositions in an overcomplete dictionary. Section 4.3 explains the optimality of using the minimum  $\ell^1$  norm decomposition and explains the formulation we used in this thesis. Section 4.4 explains how we use Lagrange multipliers to transform a constrained optimization problem into an unconstrained optimization problem. Section 4.5 is about how to choose the parameters in our method. Section 4.6 points out that a homotopic method converges to the minimum  $\ell^1$  norm decomposition. Section 4.7 describes the Newton method. Section 4.8 surveys existing methods and softwares and explains some advantages of our approach. Section 4.9 gives a preview about the importance of iterative methods and why we use them. Section 4.10 gives more detail to the numerical solution of the problem. Finally, in Section 4.11, we make some general remarks. Section 4.12 contains all relevant proofs in this chapter.

### 4.1 Why Combined Image Representation?

Recently, many new methods for signal/image representation have been proposed, including wavelets, wavelet packets, cosine packets, brushlets, edgelets, and ridgelets. Typically, each of these is good for a specific class of features, but not for others. For example, for 1-D signal representation, wavelets are *effective* at representing signals made by impulses—where

“effective” means that there are only a few coefficients that have large amplitudes—while not effective at representing oscillatory signals. At the same time, the Fourier basis is good at representing oscillatory signals, but not at representing impulses. For 2-D images, 2-D wavelets are effective at representing point singularities and patches; edgelets are effective at representing linear singularities [50]. Different transforms are effective at representing different image features. An image is usually made of several features. Combining several transforms, we have more flexibility, hopefully enabling sparse representation.

After combining several transforms, we have an overcomplete system. How do we find a sparse decomposition in an overcomplete system? This is a *huge* topic that we are going to address in the next section, Section 4.2. Typically, finding a sparse decomposition in a combined dictionary is a much more computationally intensive job than implementing any single transform of them. Here we give a handwaving example. Suppose  $y$  is a vectorized image, and we want to decompose it in a dictionary made by 2-D discrete cosine basis and 2-D wavelet basis,

$$y = T_1 x_1 + T_2 x_2, \quad (4.1)$$

where  $T_1$  and  $T_2$  are matrices whose columns are vectorized basis functions of the 2-D DCT and the 2-D wavelet transform, respectively.  $x_1$  and  $x_2$  are coefficient vectors.  $y, x_1, x_2 \in \mathbb{R}^{N^2}$ . If there is only  $T_1$  or  $T_2$ , it takes an  $O(N^2)$  or  $O(N^2 \log N)$  algorithm to get the coefficient  $x_1$  or  $x_2$ . But if we want to find a sparse solution to the overcomplete system (4.1), say, find the minimum  $\ell^1$  norm solution to (4.1), then we need to solve linear programming (LP) problem,

$$\begin{aligned} & \text{minimize} && e^T(x_1^+ + x_1^- + x_2^+ + x_2^-), \\ & \text{subject to} && x_1^+, x_1^-, x_2^+, x_2^- \geq 0, \\ & && y = T_1(x_1^+ - x_1^-) + T_2(x_2^+ - x_2^-), \\ & && e = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \end{aligned}$$

which becomes much more complicated. We can solve it with the simplex method or the

interior point method. As we know, solving this LP problem in general takes more time than doing a 2-D DCT or a 2-D wavelet transform.

Additional information about why we choose a minimum  $\ell^1$  norm solution is in Section 4.2 and Section 4.3.

## 4.2 Sparse Decomposition

There is by now extensive research on finding sparse decompositions. These methods can be roughly classified into three categories:

1. greedy algorithms,
2. global optimization algorithms,
3. special structures.

A *global optimization algorithm* searches for a decomposition minimizing a specified objective function while satisfying some constraints. (Typically the objective function is convex and any local minimizer is also a global minimizer.) The basis pursuit method (BP) [25, 27] is a global optimization algorithm. In the noise free case, it minimizes  $\|x\|_1$  subject to  $\Phi x = y$ . Another example of the global optimization algorithm is the *method of frames* (MOF) [34], which minimizes  $\|x\|_2$  subject to  $\Phi x = y$ . Note that an ideal objective function would be the  $\ell^0$  norm of  $x$ , but that makes it a combinatorial optimization problem.

A *greedy algorithm* is a stepwise algorithm: at every step, the greedy algorithm takes one or several elements out of the dictionary into a linear superposition of the desired image. A well-known example is *Matching Pursuit* (MP) [102, 100]. The idea of MP is that at every step, the algorithm picks the atom that is most correlated with the residual. Some researchers give theoretical bounds for this greedy method, for example [112]. The newly published *high-resolution pursuit* [89] is another example of greedy algorithms.

Some algorithms utilize the *special structure* of a dictionary. For example, *best orthogonal basis* (BOB) [142] searches for an orthonormal basis that minimizes the additive entropy in a dictionary that has a binary tree structure. The dictionary can be, for example, cosine packets or wavelet packets.

Since a greedy algorithm is a stepwise algorithm, it runs the risk of being trapped in a bad sequence. Some examples are given in [27, 40]. Some numerical experiments, together

with some recent advances in theoretical study, show that a global optimization algorithm like BP is more stable in recovering the original sparse decomposition, if it exists. But BP is a computationally intensive method. The remainder of this thesis is mainly devoted to overcoming this barrier. In the next section, Section 4.3, we first explain the optimality of the minimum  $\ell^1$  norm decomposition and then give our formulation, which is a variation of the exact minimum  $\ell^1$  norm decomposition. This formulation determines the numerical problem that we try to solve.

### 4.3 Minimum $\ell^1$ Norm Solution

The minimum  $\ell^1$  norm solution means that in a decomposition  $y = \Phi x$ , where  $y$  is the desired signal/image,  $\Phi$  is a *flat* matrix with each column being an atom from an overcomplete dictionary and  $x$  is a coefficient vector, we pick the one that has the minimum  $\ell^1$  norm ( $\|x\|_1$ ) of the coefficient vector  $x$ .

A heuristic argument about the optimality of the minimum  $\ell^1$  norm decomposition is that it is the “best” convexification of the minimum  $\ell^0$  norm problem. Why? Suppose we consider all the convex functions that are supported in  $[-1, 1]$  and upper bounded by the  $\ell^0$  norm function and we solve

$$\underset{x}{\text{maximize}} f(x), \quad \text{subject to} \quad \begin{cases} f(x) \leq \|x\|_0, \\ \|x\|_\infty \leq 1, \\ f \text{ is convex.} \end{cases}$$

The solution of the above problem is  $\|x\|_1$ .

The ideas of using the minimum  $\ell^1$  norm solutions in signal estimation and recovery date back to 1965, in which Logan [96] described some so-called minimum  $\ell^1$  norm phenomena. Another good reference on this topic is [55]. The idea of the minimum  $\ell^1$  norm phenomenon is the following: a signal cannot be “sparse” in both time and Fourier (frequency) domain; if we know the signal is limited in one domain (e.g., frequency domain) and the signal is unknown in a relatively small set in another domain (e.g., time domain), then we may be able to perfectly recover the signal by solving the minimum  $\ell^1$  norm problem.

This phenomenon is connected to the uncertainty principle. But the conventional uncertainty principle is based on the  $\ell^0$  norm. In the continuous case, the  $\ell^0$  norm is the length of interval; in the discrete case, the  $\ell^0$  norm is the cardinality of a finite set. As we know,

$\ell^0$  norm is a nonconvex function while  $\ell^1$  norm is convex.

Some inspiring applications of the minimum  $\ell^1$  norm phenomenon are given in [55]:

- Recovering missing segments of a bandlimited signal. Suppose  $s$  is a bandlimited signal with frequency components limited in  $B$ . But  $s$  is missing in a time interval  $T$ . Given  $B$ , if  $|T|$  is small enough, then the minimizer of  $\|s - s'\|_1$  among all  $s'$  whose frequency components are limited in  $B$  is exactly  $s$ .
- Recovery of a “sparse” wide-band signal from narrow-band measurements. Suppose  $s$  is a signal that is “sparse” in the time domain. As we know, a time-sparse signal must occupy a wide band in the frequency domain. Suppose  $N_t \geq |s|$ . Suppose we can only observe a bandlimited fraction of  $s$ ,  $r = P_B s$ , here  $P_B$  functioning like a bandpass filter. We can perfectly recover  $s$  by finding the minimizer of  $\|r - P_B s'\|_1$  among all the  $s'$  satisfying  $|s'| \leq N_t$ . This phenomenon has application in several branches of applied science, where instrumental limitations make the available observation bandlimited. On the other hand, if we consider the symmetry between the time domain and the frequency domain, this is a dual of the previous case. We simply switch the positions of the time domain and the frequency domain.

Another recent advance in theory [43] has given an interesting result. Suppose that the overcomplete dictionary we consider is a combination of two complementary orthonormal bases. By “complementary” we mean that the maximum absolute value of the inner product of any two elements (one from each basis) is upper bounded by a small value. Suppose the observed signal  $y$  is made by a small number of atoms in the dictionary. Solving the minimum  $\ell^1$  norm problem will give us the same solution as solving the minimum  $\ell^0$  norm problem. More specifically, if a dictionary is made by two bases—Dirac basis and Fourier basis—and if the observation  $y$  is made by fewer than  $\sqrt{N}/2$  atoms from the dictionary, where  $N$  is the size of the signal, then the minimum  $\ell^1$  norm decomposition is the same as the minimum  $\ell^0$  norm decomposition.

$\|x\|_0$  denotes the number of nonzero elements in the vector  $x$ . The  $\ell^0$  norm is generally regarded as the measure of sparsity. So the minimum  $\ell^0$  norm decomposition is generally regarded as the sparsest decomposition. Note the minimum  $\ell^0$  norm problem is a combinatorial problem and in general is NP hard. But the minimum  $\ell^1$  norm problem is a convex optimization problem and can be solved by some polynomial-time optimization methods,

for example, linear programming. The previous result shows that we can attack a combinatorial optimization problem by solving a convex optimization problem; if the solution satisfies certain conditions, then the solution of the convex optimization problem is the same as the solution of the combinatorial optimization problem. This gives a new possibility to solve an NP hard problem.

More examples of identical minimum  $\ell^1$  norm decomposition and minimum  $\ell^0$  norm decomposition are given in [43].

An exact minimum  $\ell^1$  norm problem is

$$(eP_1) \quad \underset{x}{\text{minimize}} \quad \|x\|_1, \quad \text{subject to} \quad y = \Phi x.$$

We consider a problem whose constraint is based on the  $\ell^2$  norm:

$$(P_1) \quad \underset{x}{\text{minimize}} \quad \|x\|_1, \quad \text{subject to} \quad \|y - \Phi x\|_2 \leq \epsilon,$$

where  $\epsilon$  is a constant. Section 4.4 explains how to solve this problem.

## 4.4 Lagrange Multipliers

We explain how to solve  $(P_1)$  based on some insights from the interior point method. One key idea is to select a barrier function and then minimize the sum of the objective function and a multiplication of a positive constant and the barrier function. When the solution  $x$  approaches the boundary of the feasible set, the barrier function becomes infinite, thereby guaranteeing that the solution is always within the feasible set. Note that the subsequent optimization problem has become a nonconstrained optimization problem. Hence we can apply some standard methods—for example, the Newton method—to solve it.

A typical interior point method uses a logarithmic barrier function [113]. The algorithm in [25] is equivalent to using an  $\ell^2$  penalty function. Since the feasible set in  $(P_1)$  is the whole Euclidean space, the demand of restricting the solution in a feasible set is not essential. We actually solve the following problem

$$\underset{x}{\text{minimize}} \quad \|y - \Phi x\|_2^2 + \lambda \rho(x), \tag{4.2}$$

where  $\lambda$  is a scalar parameter and  $\rho$  is a convex separable function:  $\rho(x) = \sum_{i=1}^N \bar{\rho}(x_i)$ , where

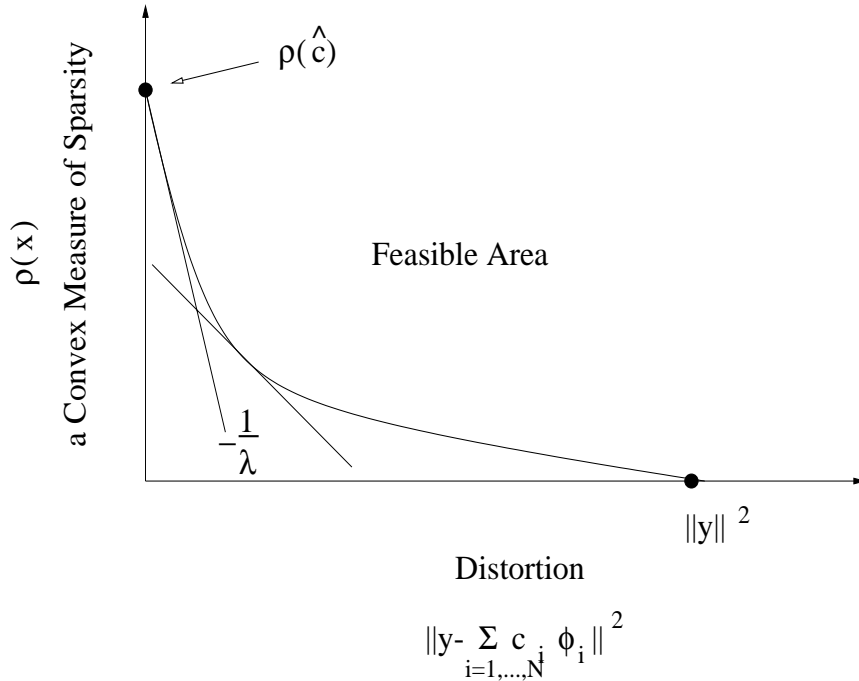


Figure 4.1: Quasi-sparsity and distortion curve.

$\bar{\rho}$  is a convex 1-D function. Note this is the idea of quadratic penalty-function method in solving the following optimization problem with exact constraints:

$$\underset{x}{\text{minimize}} \quad \rho(x), \quad \text{subject to} \quad y = \Phi x. \tag{4.3}$$

To better explain the connections we raise here, we introduce a concept called *Quasi-sparsity & distortion curve*. “Quasi-sparsity” refers to the small values in the quasi-sparsity measurement  $\rho(x)$ . Figure 4.1 gives a depiction. The horizontal axis is the distortion measure  $\|y - \Phi x\|_2^2$ . The vertical axis is a measure of quasi-sparsity, in our case  $\rho(x)$ . Allowing some abuse of the terminology “sparsity”, we call this plane a distortion-sparsity (D-S) plane. If there exists  $x$  such that  $(u, v) = (\|y - \Phi x\|_2^2, \rho(x))$ , then we say the point  $(u, v)$  on the D-S plane is feasible. We know  $\|y - \Phi x\|_2^2$  is a quadratic form. When  $\rho(x)$  is a convex function of  $x$ , all the feasible points on the D-S plane form a convex set; we call this convex set a *feasible area*. For a fixed value of distortion, there is a minimum achievable value for  $\rho(x)$ . If all the points like this form a continuous curve, we call it a Quasi-Sparsity

& Distortion (QSD) curve. Actually, the QSD curve is the lower boundary of the feasible area, as shown in Figure 4.1. Note in the figure, the notation  $\sum_{i=1,\dots,N} c_i \phi_i$  serves the same meaning as  $\Phi x$  in the above text.

A noteworthy phenomenon is that for fixed  $\lambda$ , the corresponding  $(u, v)$  point given by the solution of (4.2) is actually the tangent point of a straight line having slope  $-1/\lambda$  with the QSD curve. The tangent is the leftmost straight line having slope  $-1/\lambda$  and intersecting with the feasible area. Moreover, the QSD curve is the pointwise upper bound of all these tangents. We can see a similar argument on the *rate and distortion curve* (R&D curve) in Information Theory [10, 33].

There are two limiting cases:

1. When the distortion  $\|y - \Phi x\|_2^2$  is zero, the QSD curve intersects with the vertical axis. The intersection point, which has the coordinates  $(0, \rho(\hat{c}))$ , is associated with the solution  $\hat{c}$  to the exact constraint problem as in (4.3).
2. When the measure of sparsity  $\rho(x)$  is zero, because  $\rho(x)$  is convex, nonnegative, and symmetric about zero, we may think of  $x$  as an all-zero vector. Hence the distortion is equal to  $\|y\|^2$ . The corresponding point on the D-S plane is  $(\|y\|^2, 0)$ , and it is the intersection of the QSD curve with the horizontal axis.

## 4.5 How to Choose $\rho$ and $\lambda$

Since  $\rho(x) = \sum_{i=1}^N \bar{\rho}(x_i)$ , in order to determine  $\rho$ , we only need to determine  $\bar{\rho}$ . We choose  $\bar{\rho}$  as a convex,  $\ell^1$ -like and  $C^2$  function. We choose  $\bar{\rho}$  to be convex, so that the optimization problem has a global solution. We choose  $\bar{\rho}$  to be an  $\ell^1$ -like function, for the reasons mentioned in Section 4.3. We choose  $\bar{\rho}$  to be  $C^2$  so that the problem in (4.2) is tractable by standard Newton methods. Our choice of  $\bar{\rho}$  is

$$\bar{\rho}(x, \gamma) = |x| + \frac{1}{\gamma} e^{-\gamma|x|} - \frac{1}{\gamma}, \quad \text{for } \gamma > 0,$$

where  $\gamma$  is a controlling parameter. Note when  $\gamma \rightarrow +\infty$ ,  $\bar{\rho}(x, \gamma) \rightarrow |x|$ . The derivatives of  $\bar{\rho}$  have the form:

$$\frac{\partial}{\partial x} \bar{\rho}(x, \gamma) = \begin{cases} 1 - e^{-\gamma x}, & x \geq 0, \\ -1 + e^{\gamma x}, & x \leq 0, \end{cases} \quad (4.4)$$

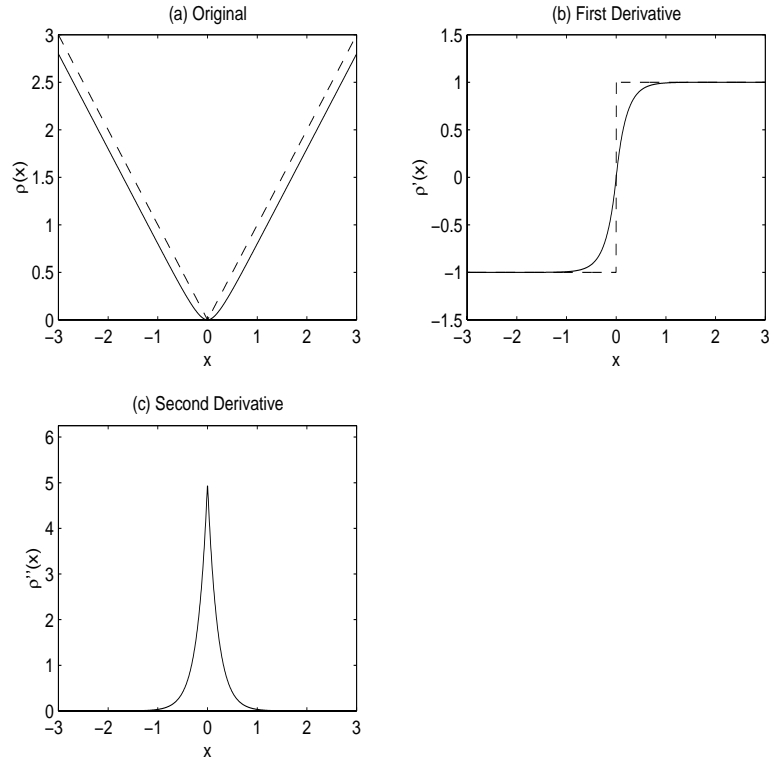


Figure 4.2: Function  $\bar{\rho}$  (as  $\rho$  in figures) and its first and second derivatives.  $\bar{\rho}$ ,  $\bar{\rho}'$  and  $\bar{\rho}''$  are solid curves. The dashed curve in figure (a) is the absolute value. The dashed curve in figure (b) is the signum function.

and

$$\frac{\partial^2}{\partial^2 x} \bar{\rho}(x, \gamma) = \gamma e^{-\gamma|x|}. \quad (4.5)$$

It is easy to verify that  $\bar{\rho}$  is  $C^2$ .

Figure 4.2 shows for fixed  $\gamma$ , the function  $\bar{\rho}$  and its first and second derivatives. Figure 4.3 shows the function  $\bar{\rho}$  at a neighborhood of the origin with different values of  $\gamma$ .

For fixed  $\gamma$  and function  $\bar{\rho}$ , the following result tells us how to choose  $\lambda$ .

**Proposition 4.1** *If  $\hat{x}$  is the solution to the problem in (4.2), then we have*

$$\|\Phi^T(y - \Phi\hat{x})\|_\infty \leq \frac{\lambda}{2}. \quad (4.6)$$

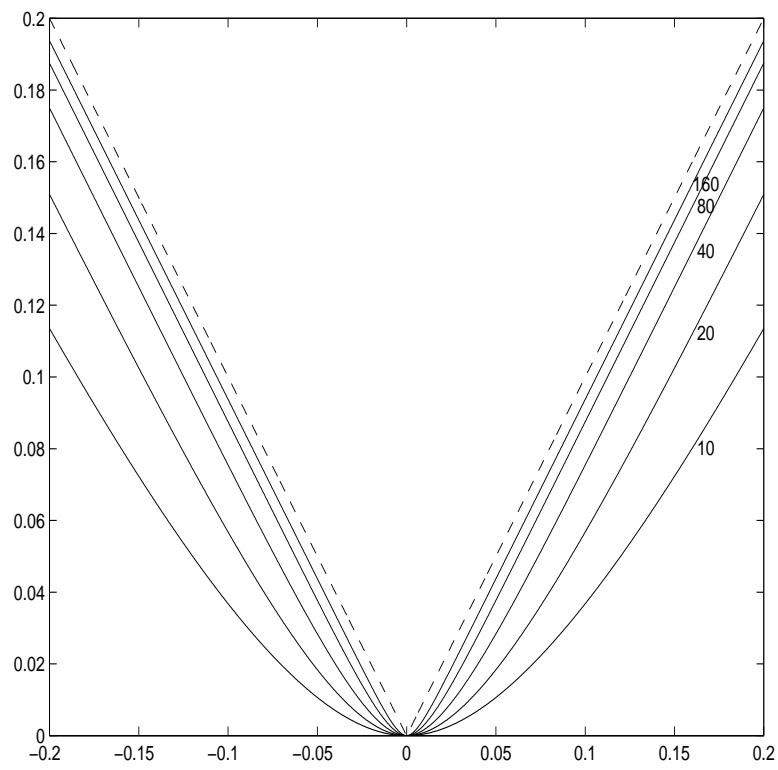


Figure 4.3: Function  $\bar{\rho}$  at a neighborhood of origin with different values of  $\gamma$ ,  $\gamma = 10, 20, 40, 80, 160$ . The dashed line is the absolute value.

A way to interpret the above result is that for the residual  $r = y - \Phi\hat{x}$ , the maximum amplitude of the analysis transform of the residual  $\|\Phi^T r\|_\infty$  is upper bounded by  $\frac{\lambda}{2}$ . Hence if  $\lambda$  is small enough and if  $\Phi^T$  is norm preserving—each column of  $\Phi$  has almost the same  $l_2$  norm—then the deviation of the reconstruction based on  $\hat{x}$ ,  $\Phi\hat{x}$ , from the image  $y$  is upper bounded by a small quantity (literally  $\frac{\lambda}{2}$ ) at each direction given by the columns of  $\Phi$ . So if we choose a small  $\lambda$ , the corresponding reconstruction cannot be much different from the original image.

After choosing  $\rho$  and  $\lambda$ , we have the Hessian and gradient of the objective function (later denoted by  $f(x)$ ). For fixed vector  $x_k = (x_1^k, x_2^k, \dots, x_N^k)^T \in \mathbb{R}^N$ , the gradient at  $x_k$  is

$$g(x_k) = -2\Phi^T y + 2\Phi^T \Phi x_k + \lambda \begin{pmatrix} \bar{\rho}'(x_1^k) \\ \vdots \\ \bar{\rho}'(x_N^k) \end{pmatrix}, \quad (4.7)$$

where  $x_i^k$  is the  $i$ -th element of vector  $x_k$ ,  $i = 1, 2, \dots, N$ , and the Hessian is the matrix

$$H(x_k) = 2\Phi^T \Phi + \lambda \begin{pmatrix} \bar{\rho}''(x_1^k) & & \\ & \ddots & \\ & & \bar{\rho}''(x_N^k) \end{pmatrix}, \quad (4.8)$$

where  $\bar{\rho}''$  is the second derivative of function  $\bar{\rho}$ .

## 4.6 Homotopy

Based on the previous choice of  $\rho$  and  $\bar{\rho}$ , when the parameter  $\gamma$  goes to  $+\infty$ ,  $\rho(x, \gamma) = \sum_{i=1}^N \bar{\rho}(x_i, \gamma)$  goes to function  $\|x\|_1$ . Note our ultimate goal is to solve the minimum  $\ell^1$  norm problem ( $P_1$ ). Considering the Lagrangian multiplier method, for a fixed  $\lambda$ , we solve

$$\underset{x}{\text{minimize}} \|y - \Phi x\|_2^2 + \lambda \|x\|_1. \quad (4.9)$$

When  $\gamma$  goes to  $+\infty$ , will the solution of the problem in (4.2) converge to the solution of the problem in (4.9)?

We prove the convergence under the following two assumptions. The first one is easy to satisfy. The second one seems too rigorous and may not be true for many situations that we

are interested in. We suspect that the convergence is still true when the second assumption is relaxed. We leave the analysis for future research.

**Assumption 1** For given  $y$ ,  $\Phi$  and  $\lambda$ , the solution to the problem in (4.9) exists and is unique.

Since the objective function is convex and its Hessian, as in (4.8), is always positive definite, it is easy to prove that the above assumption is true in most cases.

**Assumption 2**  $\Phi^T\Phi$  is a diagonally dominant matrix, which means that there exists a constant  $\epsilon > 0$  such that

$$|(\Phi^T\Phi)_{ii}| \geq (1 + \epsilon) \sum_{k \neq i} |(\Phi^T\Phi)_{ik}|, \quad i = 1, 2, \dots, N.$$

As we mentioned, this assumption is too rigorous in many cases. For example, if  $\Phi$  is a concatenation of the Dirac basis and the Fourier basis, this assumption does not hold.

**Theorem 4.1** For fixed  $\gamma$ , let  $x(\gamma)$  denote the solution to problem (4.2). Let  $x'$  denote the solution to problem (4.9). If the previous two assumptions are true, we have

$$\lim_{\gamma \rightarrow +\infty} x(\gamma) = x'. \quad (4.10)$$

From the above result, we can apply the following method. Starting with a small  $\gamma_1$ , we get the solution  $x(\gamma_1)$  of problem (4.2). Next we choose  $\gamma_2 > \gamma_1$ , set  $x(\gamma_1)$  as the initial guess, and apply an iterative method to find the solution (denoted by  $x(\gamma_2)$ ) of problem (4.2). We repeat this process, obtaining a parameter sequence  $\gamma_1, \gamma_2, \gamma_3, \dots$ , and a solution sequence  $x(\gamma_1), x(\gamma_2), x(\gamma_3), \dots$ . From Theorem 4.1, the sequence  $\{x(\gamma_i), i = 1, 2, \dots\}$  converges to the solution of problem (4.9). This method may save computing time because when  $\gamma$  is small, an iterative method takes a small number of steps to converge. After finding an approximate solution by using a small valued  $\gamma$ , we then take it as a starting point for an iterative method to find a more precise solution. Some iterations may be saved in the early stage.

## 4.7 Newton Direction

For fixed  $\gamma$ , we use Newton's method for convex optimization to solve problem (4.2). Starting from an initial guess  $x^{(0)}$ , at every step, Newton's then generates a new vector that is closer to the true solution:

$$x^{(i+1)} = x^{(i)} + \beta_i n(x^{(i)}), \quad i = 0, 1, \dots, \quad (4.11)$$

where  $\beta_i$  is a damping parameter ( $\beta_i$  is chosen by line search to make sure that the value of the objective function is reduced),  $n(x^{(i)})$  is the Newton direction, a function of the current guess  $x^{(i)}$ . Let  $f(x) = \|y - \Phi x\|_2^2 + \lambda \rho(x)$  denote the objective function in problem (4.2). The gradient and Hessian of  $f(x)$  are defined in (4.8) and (4.7). The Newton direction at  $x^{(i)}$  satisfies

$$\left[ H(x^{(i)}) \right] \cdot n(x^{(i)}) = -g(x^{(i)}). \quad (4.12)$$

This is a system of linear equations. We choose iterative methods to solve it, as discussed in the next section and also the next chapter.

## 4.8 Comparison with Existing Algorithms

We compare our method with the method proposed by Chen, Donoho and Saunders (CDS) [27]. The basic conclusion is that the two methods are very similar, but our method is simpler in derivation, requires fewer variables and is potentially more efficient in numerical computing. We start by reviewing the CDS approach, describe the difference between our approach and theirs and then discuss benefits of these changes.

Chen, Donoho and Saunders proposed a primal-dual log-barrier perturbed LP algorithm. Basically, they solve [27, equation (6.3), page 56]

$$\underset{x^o}{\text{minimize}} \quad c^T x^o + \frac{1}{2} \|\gamma x^o\|^2 + \frac{1}{2} \|p\|^2 \quad \text{subject to} \quad Ax^o + \delta p = y, \quad x^o \geq 0,$$

where

- $\gamma$  and  $\delta$  are normally small (e.g.,  $10^{-4}$ ) regularization parameters;
- $c = \lambda \mathbf{1}$ , where  $\lambda$  is the penalization parameter as defined in (4.2) and  $\mathbf{1}$  is an all-one

column vector;

- $x^o = (x_+^T, x_-^T)^T$ , where  $x_+$  and  $x_-$  are the positive and negative part of the column vector  $x$  that is the same as the “ $x$ ” in (4.2):  $x = x_+ - x_-$ ,  $x_+ \geq 0$  and  $x_- \geq 0$ ;
- $A = [\Phi, -\Phi]$  and  $\Phi$  is the same matrix as the one specified in (4.2);
- $y$  is equivalent to the “ $y$ ” in (4.2);
- $x^o \geq 0$  means that each entry of the vector  $x^o$  is greater than or equal to 0.

The main effort of this approach is to solve the following system of linear equations [27, equation (6.4), page 56]

$$(ADA^T + \delta^2 I)\Delta y = \mathbf{r} - AD(X^{-1}\mathbf{v} - \mathbf{t}), \quad (4.13)$$

where  $\mathbf{r}$ ,  $\mathbf{v}$  and  $\mathbf{t}$  are column vectors given in CDS paper,  $D = (X^{-1}Z + \gamma^2 I)^{-1}$ ,  $X$  and  $Z$  are diagonal matrices composed from primal variable  $x^o$  and dual slack variable  $\mathbf{z}$ . (For more specific description, we refer readers to the original paper.)

Recall in our approach, to obtain the Newton direction, we solve

$$[2\Phi^T\Phi + \lambda\rho''(x_k)]n = 2\Phi^T(y - \Phi x_k) - \lambda\rho'(x_k), \quad (4.14)$$

where  $\rho''(x_k) = \text{diag}\{\bar{\rho}''(x_1^k), \dots, \bar{\rho}''(x_N^k)\}$ ,  $\rho'(x_k) = (\bar{\rho}'(x_1^k), \dots, \bar{\rho}'(x_N^k))^T$  and  $n$  is the desired Newton direction that is equivalent to  $n(x^{(k)})$  in (4.12).

After careful examination of the matrices on the left hand sides of both (4.13) and (4.14), we observe that both of them are positive definite matrices having the same size. By this we say that the two approaches are similar. But it is hard to say which one may have a better eigenvalue distribution than the other.

When they are close to the solution, both algorithms become slow to converge. Note that the system of linear equations in (4.13) is for a perturbed LP problem. If both  $\gamma$  and  $\delta$  are extremely small, the eigenvalue distribution of the matrix  $ADA^T + \delta^2 I$  can be very bad, so that iterative methods converge slowly. The same discussion is also true for the matrix on the left hand side of (4.14). When the vector  $x_k$  has nonzero and spread entries, the eigenvalue distribution of the positive definite matrix  $2\Phi^T\Phi + \lambda\rho''(x_k)$  could be bad for any iterative methods.

The appealing properties of our approach are:

1. Avoiding introducing the dual variables as in the primal-dual log-barrier perturbed LP approach. We transfer the problem to a unconstrained convex optimization problem and apply Newton methods directly to this problem. It is not necessary to introduce the dual variables and we do not need to increase the size of the primal variable  $x$  by two (this was done in CDS approach). It is not necessary to consider the KKT conditions. Although the core problem—as in (4.14)—is very similar to the core problem—as in (4.13)—of CDS approach, the derivation is simpler and more direct.
2. Instead of using a standard conjugate gradients methods, we choose LSQR, which is analytically equivalent to CG algorithm but is numerically more stable.

Chen, Donoho and Saunders [26] have successfully developed software to carry out numerical experiments with problems having the size of thousands by tens of thousands. We choose to implement a new approach in the hope that it simplifies the algorithm and hopefully increase the numerical efficiency. A careful comparison of our new approach with the one previously implemented will be an interesting future research topic. At the same time, we choose LSQR instead of CG. (Chen *et al.* showed how to use LSQR when  $\delta > 0$  in (4.13), but their experiments used CG.)

## 4.9 Iterative Methods

In our problem, we choose an iterative solver to solve the system in (4.12) because the Hessian is a positive definite matrix with special structure to facilitate fast algorithms for matrix-vector multiplication. As we know, when there is a fast (low complexity) algorithm for matrix-vector multiplication and assuming the iterative solver takes a moderate number of iterations to converge, an iterative solver is an ideal tool to solve a system of linear equations [74, 78].

Here, we have a fast algorithm to multiply with the Hessian  $H(x)$  because, as in (4.8), the second term is a diagonal matrix, and we know it is an  $O(N)$  algorithm to multiply with a diagonal matrix. The  $\Phi$  in our model is a combination of transforms having fast algorithms (for example, 2-D wavelet transforms, 2-D DCT and edgelet-like transforms), so we have fast algorithms to multiply with  $\Phi$  and  $\Phi^T$  (which is simply the adjoint). Hence we have a fast algorithm for multiplying with the Hessian.

Choosing the “correct” iterative method is a big topic. The next chapter is dedicated to this topic. We choose a variation of the conjugate gradient method: LSQR [116, 117].

## 4.10 Numerical Issues

*Damped Newton direction.* To ensure that Newton's method converges, we implement a backtracking scheme to find the value for  $\beta_i$  in (4.11). It guarantees that at every iteration, the value of the objective function is reduced.

*Truncated CG.* In order to save computing time, in the early Newton iterations, we terminate the CG solver before it reaches high precision, because an inexact Newton direction does not hurt the precision of the final solution by Newton method [38, 39].

The algorithm is implemented in Matlab. Fast algorithms for wavelet and edgelet transforms are implemented in C and called by Matlab through a CMEX interface.

## 4.11 Discussion

### 4.11.1 Connection With Statistics

In statistics, we can find the same method being used in *model selection*, where we choose a subset of variables so that the model is still sufficient for prediction and inference. To be more specific, in linear regression models, we consider

$$y = X\beta + \varepsilon,$$

where  $y$  is the response,  $X$  is the model matrix with every column being values of a variable (predictor),  $\beta$  is the coefficient vector, and  $\varepsilon$  is a vector of IID random variables. Model selection in this setting means choosing a subset of columns of  $X$ ,  $X^{(0)}$ , so that for most of the possible responses  $y$ , we have  $y \approx X^{(0)}\beta^{(0)}$ , where  $\beta^{(0)}$  is a subvector of  $\beta$  with locations corresponding to the selected columns in  $X^{(0)}$ . The difference (or prediction error),  $y - X^{(0)}\beta^{(0)}$ , is negligible in the sense that it can be interpreted as a realization of the random noise vector  $\varepsilon$ .

Typically, people use penalized regression to select the model. Basically, we solve

$$(PR) \quad \underset{\beta}{\text{minimize}} \quad \|y - X\beta\|_2^2 + \lambda\rho(\beta),$$

which is exactly the problem we encountered in (4.2). After solving problem (PR), we can pick the  $i$ th column in  $X$  if  $\beta_i$  has a significant amplitude. When  $\rho(\beta) = \|\beta\|_1$ , the method (PR) is called LASSO by R. Tibshirani [133] and Basis Pursuit by Chen et al [27]. When

$\rho(\beta) = \|\beta\|_2^2$ , the method (*PR*) is called *ridge regression* by Hoerl and Kennard [81, 80].

### 4.11.2 Non-convex Sparsity Measure

An ideal measure of sparsity is usually nonconvex. For example, in (4.2), the number of nonzero elements in  $x$  is the most intuitive measure of sparsity. The  $\ell^0$  norm of  $x$ ,  $\|x\|_0$ , is equal to the number of nonzero elements, but it is *not* a convex function. Another choice of measure of sparsity is the logarithmic function; for  $x = (x_1, \dots, x_N)^T \in \mathbb{R}^N$ , we can have  $\rho(x) = \sum_{i=1}^N \log|x_i|$ . In sparse image component analysis, another nonconvex sparsity measure is used:  $\rho(x) = \sum_{i=1}^N \log(1 + x_i^2)$  [53].

Generally speaking, a nonconvex optimization problem is a combinatorial optimization problem, and hence it is NP hard. Some discussion about how to use reweighting methods to solve a nonconvex optimization problem is given in the next subsection.

### 4.11.3 Iterative Algorithm for Non-convex Optimization Problems

Sometimes, a *reweighted iterative method* can be used to find a *local* minimum for a non-convex optimization problem. Let's consider the following problem:

$$(LO) \quad \underset{x}{\text{minimize}} \sum_{i=1}^N \log|x_i|, \quad \text{subject to } y = \Phi x;$$

and its corresponding version with a Lagrangian multiplier  $\lambda$ ,<sup>1</sup>

$$(LO_\lambda) \quad \underset{x}{\text{minimize}} \|y - \Phi x\|_2^2 + \lambda \sum_{i=1}^N \log(|x_i| + \delta).$$

Note that the objective function of (*LO*) is not convex.

Let's consider a reweighted iterative algorithm: for  $\delta > 0$ ,

$$(RIA) \quad x^{(k+1)} = \underset{x}{\text{argmin}} \sum_{i=1}^N \frac{|x_i|}{|x_i^{(k)}| + \delta}, \quad \text{subject to } y = \Phi x;$$

---

<sup>1</sup>More precisely, ( $LO_\lambda$ ) is the Lagrangian multiplier version of the following optimization problem:

$$\underset{x}{\text{minimize}} \sum_{i=1}^N \log(|x_i| + \delta), \quad \text{subject to } \|y - \Phi x\| \leq \varepsilon.$$

Note when  $\delta$  and  $\varepsilon$  are small, it is close to (*LO*).

and its corresponding version with a Lagrangian multiplier  $\lambda$ ,

$$(RIA_\lambda) \quad x^{(k+1)} = \underset{x}{\operatorname{argmin}} \|y - \Phi x\|_2^2 + \lambda \sum_{i=1}^N \frac{|x_i|}{|x_i^{(k)}| + \delta}.$$

We know the following results.

**Theorem 4.2** *Suppose we add one more constraint on  $x$ :  $x_i \geq 0, i = 1, 2, \dots, N$ . The sequence generated by  $(RIA)$ ,  $\{x_i^{(k)}, k = 1, 2, 3, \dots\}$ , converges in the sense that the difference of sequential elements goes to zero:*

$$|x_i^{(k+1)} - x_i^{(k)}| \rightarrow 0, \quad \text{as } k \rightarrow +\infty.$$

We learned this result from [95].

**Theorem 4.3** *If the sequence  $\{x_i^{(k)}, k = 1, 2, 3, \dots\}$  generated by  $(RIA_\lambda)$  converges, it converges to a local minimum of  $(LO_\lambda)$ .*

Some related works can be found in [36, 106]. There is also some ongoing research, for example, the work being carried out by Boyd, Lobo and Fazel in the Information Systems Laboratories, Stanford.

## 4.12 Proofs

### 4.12.1 Proof of Proposition 4.1

When  $\hat{x}$  is the solution to the problem as in (4.2), the first order condition is

$$0 = \lambda \nabla \rho(x) + 2\Phi^T \Phi x - 2\Phi^T y,$$

where  $\nabla \rho(x)$  is the gradient vector of  $\rho(x)$ . Hence

$$\frac{1}{2} \lambda \nabla \rho(x) = \Phi^T (y - \Phi x). \quad (4.15)$$

Recall  $\rho(x) = \sum_{i=1}^N \bar{\rho}(x_i)$ . It's easy to verify that  $\forall x_i, |\bar{\rho}'(x_i)| \leq 1$ . Hence for the left-hand side of (4.15), we have  $\|\frac{1}{2} \lambda \nabla \rho(x)\|_\infty \leq \frac{\lambda}{2}$ . The bound (4.6) follows.  $\square$

### 4.12.2 Proof of Theorem 4.1

We will prove convergence first, and then prove that the limiting distribution is  $x'$ .

When  $\gamma$  takes all the real positive values,  $(x(\gamma), \gamma)$  forms a continuous and differentiable trajectory in  $\mathbb{R}^{N+1}$ . Let  $x_\gamma^i$  denote the  $i$ th element of  $x(\gamma)$ . By the first-order condition, we have

$$0 = -2\Phi^T(y - \Phi x(\gamma)) + \lambda \begin{pmatrix} \frac{\partial \bar{\rho}(x_\gamma^1, \gamma)}{\partial x} \\ \vdots \\ \frac{\partial \bar{\rho}(x_\gamma^N, \gamma)}{\partial x} \end{pmatrix}.$$

Taking  $\frac{d}{d\gamma}$  on both sides, we have

$$0 = 2\Phi^T \Phi \frac{dx(\gamma)}{d\gamma} + \lambda \begin{pmatrix} \frac{\partial^2 \bar{\rho}(x_\gamma^1, \gamma)}{\partial \gamma \partial x} + \frac{dx_\gamma^1}{d\gamma} \frac{\partial^2 \bar{\rho}(x_\gamma^1, \gamma)}{\partial x \partial x} \\ \vdots \\ \frac{\partial^2 \bar{\rho}(x_\gamma^N, \gamma)}{\partial \gamma \partial x} + \frac{dx_\gamma^N}{d\gamma} \frac{\partial^2 \bar{\rho}(x_\gamma^N, \gamma)}{\partial x \partial x} \end{pmatrix}.$$

Since

$$\frac{\partial^2 \bar{\rho}(x, \gamma)}{\partial \gamma \partial x} = x e^{-\gamma|x|},$$

and

$$\frac{\partial^2 \bar{\rho}(x, \gamma)}{\partial x \partial x} = \gamma e^{-\gamma|x|},$$

we have

$$-2\Phi^T \Phi \frac{dx(\gamma)}{d\gamma} = \lambda \begin{pmatrix} x_\gamma^1 e^{-\gamma|x_\gamma^1|} + \frac{dx_\gamma^1}{d\gamma} \gamma e^{-\gamma|x_\gamma^1|} \\ \vdots \\ x_\gamma^N e^{-\gamma|x_\gamma^N|} + \frac{dx_\gamma^N}{d\gamma} \gamma e^{-\gamma|x_\gamma^N|} \end{pmatrix}.$$

Based on Assumption 2, suppose that the  $k$ th entry of vector  $\frac{dx(\gamma)}{d\gamma}$  is negative and takes the maximum amplitude of the vector:

$$\left\| \frac{dx(\gamma)}{d\gamma} \right\|_\infty = -\frac{dx_\gamma^k}{d\gamma}.$$

Further supposing that  $(\Phi^T \Phi)_{kk} > 0, k = 1, 2, \dots$ , we have

$$\begin{aligned} x_\gamma^k e^{-\gamma|x_\gamma^k|} + \frac{dx_\gamma^k}{d\gamma} \gamma e^{-\gamma|x_\gamma^k|} &= \sum_{j=1}^N 2(\Phi^T \Phi)_{kj} \left( -\frac{dx_\gamma^j}{d\gamma} \right) \\ &\geq 2(\Phi^T \Phi)_{kk} \cdot \left( -\frac{dx_\gamma^k}{d\gamma} \right) - \sum_{j \neq k} |2(\Phi^T \Phi)_{kj}| \cdot \left( -\frac{dx_\gamma^k}{d\gamma} \right) \\ &\geq \frac{\epsilon}{1+\epsilon} 2(\Phi^T \Phi)_{kk} \cdot \left( -\frac{dx_\gamma^k}{d\gamma} \right). \end{aligned}$$

Hence

$$\begin{aligned} \left\| \frac{dx(\gamma)}{d\gamma} \right\|_\infty = -\frac{dx_\gamma^k}{d\gamma} &\leq \frac{x_\gamma^k e^{-\gamma|x_\gamma^k|}}{\frac{\epsilon}{1+\epsilon} 2(\Phi^T \Phi)_{kk} + \gamma e^{-\gamma|x_\gamma^k|}} \\ &\leq \frac{1}{2\sqrt{2} \sqrt{\frac{\epsilon}{1+\epsilon}} \sqrt{(\Phi^T \Phi)_{kk}}} \frac{x_\gamma^k}{\sqrt{\gamma}} e^{-\gamma|x_\gamma^k|/2} \\ &\leq \frac{1}{2\sqrt{2} \sqrt{\frac{\epsilon}{1+\epsilon}} \sqrt{(\Phi^T \Phi)_{kk}}} \frac{2}{\gamma^{3/2}} e^{-1}. \end{aligned}$$

The integration of the last term in the right-hand side of the above inequality is finite, so the integration of  $dx(\gamma)/d\gamma$  is upper bounded by a finite quantity. Hence  $x(\gamma)$  converges. When  $dx_\gamma^k/d\gamma$  is positive, the discussion is similar. This confirms convergence.

Now we prove the limiting vector  $\lim_{\gamma \rightarrow +\infty} x(\gamma)$  is  $x'$ . Let  $x(\infty) = \lim_{\gamma \rightarrow +\infty} x(\gamma)$ . Let  $f(x, \gamma)$  denote the objective function in (4.9). If  $x' \neq x(\infty)$  and by Assumption 1 the solution to problem (4.9) is unique, we have

$$\text{there exists a fixed } \epsilon > 0, \quad f(x', \infty) + \epsilon < f(x(\infty), \infty). \quad (4.16)$$

But at the same time, we have

$$f(x(\gamma), \gamma) \stackrel{1}{\leq} f(x', \gamma) \stackrel{2}{\leq} f(x', \infty) \stackrel{3}{\leq} f(x(\infty), \infty), \quad (4.17)$$

where inequality 1 is true because  $x(\gamma)$  is the minimizer at  $\gamma$ , inequality 2 is true because function  $\|x\|_1$  is always larger than  $\rho(x, \gamma)$  (see Figure 4.3), and inequality 3 is a special

case of (4.16). Consider

$$f(x(\gamma), \gamma) - f(x(\infty), \infty) = f(x(\gamma), \gamma) - f(x(\gamma), \infty) + f(x(\gamma), \infty) - f(x(\infty), \infty).$$

As  $\gamma \rightarrow +\infty$ ,  $f(x(\gamma), \gamma) - f(x(\gamma), \infty) \rightarrow 0$  because the two objective functions become closer and closer (see Figure 4.3); and  $f(x(\gamma), \infty) - f(x(\infty), \infty) \rightarrow 0$  because  $x(\infty)$  is the limit of  $x(\gamma)$ . Hence as  $\gamma$  goes to  $\infty$ ,  $f(x(\gamma), \gamma) - f(x(\infty), \infty) \rightarrow 0$ . Hence by (4.17),  $f(x', \infty) - f(x(\infty), \infty) \rightarrow 0$ . This contradicts (4.16), which contradiction is due to the assumption  $x' \neq x(\infty)$ . Hence we proved that the limiting distribution is  $x'$ .  $\square$

### 4.12.3 Proof of Theorem 4.2

Defining  $L(x) = \prod_{i=1}^N (x_i + \delta)$ , we have

1.  $L(x^{(k+1)})/L(x^{(k)}) \leq 1$ , because

$$L(x^{(k+1)})/L(x^{(k)}) = \prod_{i=1}^N \frac{x_i^{(k+1)} + \delta}{x_i^{(k)} + \delta} \leq \left( \frac{1}{N} \sum_{i=1}^N \frac{x_i^{(k+1)} + \delta}{x_i^{(k)} + \delta} \right)^N \leq 1.$$

2.  $L(x^{(k)})$  is lower bounded.

3. From the above two,  $L(x^{(k+1)})/L(x^{(k)}) \rightarrow 1$ .

4.  $L(x^{(k+1)})/L(x^{(k)}) \rightarrow 1$  implies  $\frac{x_i^{(k+1)} + \delta}{x_i^{(k)} + \delta} \rightarrow 1$  for  $i = 1, 2, \dots, N$ , because if  $\frac{x_i^{(k+1)} + \delta}{x_i^{(k)} + \delta} = 1 + \varepsilon$ , then

$$L(x^{(k+1)})/L(x^{(k)}) \leq (1 + \varepsilon) \left( 1 - \frac{\varepsilon}{N-1} \right)^{N-1} \stackrel{\text{def.}}{=} f(\varepsilon).$$

We can check that  $f(0) = 0$ ,  $f'(0) = 0$ , and  $f''(\varepsilon) < 0$  for  $|\varepsilon| < 1$ . Hence  $f(\varepsilon) \rightarrow 1$  implies  $\varepsilon \rightarrow 0$ . Hence  $L(x^{(k+1)})/L(x^{(k)}) \rightarrow 1$  implies  $f(\varepsilon) \rightarrow 1$ , which implies  $\varepsilon \rightarrow 0$ , which is equivalent to  $\frac{x_i^{(k+1)} + \delta}{x_i^{(k)} + \delta} \rightarrow 1$ .

5. From all the above, we proved Theorem 4.2.

$\square$

#### 4.12.4 Proof of Theorem 4.3

We only need to check that the stationary point, denoted by  $x^{(*)}$ , of the algorithm  $(RIA_\lambda)$  satisfies the first-order condition (FOC) of the optimization problem  $(LO_\lambda)$ .

If  $x^{(*)}$  is a stationary point of  $(RIA_\lambda)$ , then

$$0 = 2\Phi^T(\Phi x^{(*)} - y) + \lambda \begin{pmatrix} \text{sign}(x_1^{(*)})/(|\text{sign}(x_1^{(*)})| + \delta) \\ \vdots \\ \text{sign}(x_N^{(*)})/(|\text{sign}(x_N^{(*)})| + \delta) \end{pmatrix},$$

where  $x_i^{(*)}$  denotes the  $i$ th component of  $x^{(*)}$ . It is easy to check that the above equality is also the FOC of a local minimum of  $(LO_\lambda)$ .  $\square$

In fact, we can verify that

$$\frac{\text{sign}(x)}{|\text{sign}(x)| + \delta} = \frac{\text{sign}(x)}{1 + \delta}.$$

## Chapter 5

# Iterative Methods

This chapter discusses the algorithms we use to solve for the Newton direction (see (4.12)) in our sparse representation problem. We choose an iterative method because there is a fast algorithm to implement the matrix-vector multiplication. Since our matrix is Hermitian (moreover symmetric), we basically choose between CG and MINRES. We choose LSQR, which is a variation of the CG, because our system is at least positive semidefinite and LSQR is robust against rounding error caused by finite-precision arithmetic.

In Section 5.1, we start with an overview of the iterative methods. Section 5.2 explains why we favor LSQR and how to apply it to our problem. Section 5.3 gives some details on MINRES. Section 5.4 contains some discussion.

### 5.1 Overview

#### 5.1.1 Our Minimization Problem

We wish to solve the following problem:

$$\underset{x}{\text{minimize}} f(x) = \|y - \Phi x\|_2^2 + \lambda \rho(x),$$

where  $y$  is the vectorized analyzed image,  $\Phi$  is a (flat) matrix with each column a vectorized basis function of a certain image analysis transform (e.g., a vectorized basis function for the 2-D DCT or the 2-D wavelet transform),  $x$  is the coefficient vector, and  $\rho$  is a separable convex function,  $\rho(x) = \sum_{i=1}^N \bar{\rho}(x_i)$ , where  $\bar{\rho}$  is a convex 1-D function.

We apply a damped Newton method. To find the Newton direction, the following system

of linear equations must be solved:

$$H(x_k) \cdot d = g(x_k), \quad (5.1)$$

where  $H(x_k)$  and  $g(x_k)$  are the Hessian (as in (4.8)) and the gradient (as in (4.7)) of  $f(x)$  at  $x_k$  as defined in the previous chapter.

We have the following observations:

- [O1] If  $\Phi$  is a concatenation of several transform matrices and each of them has a fast algorithm, then the matrix-vector multiplications with  $\Phi$ ,  $\Phi^T$  and  $\Phi^T\Phi$  have fast algorithms.
- [O2] There is a closed form for  $\bar{\rho}''(x_k^i)$  and  $\bar{\rho}'(x_k^i)$ ,  $i = 1, 2, 3, \dots$ , so there is a low complexity  $O(N)$  algorithm to generate the diagonal matrix made by  $\bar{\rho}''(x_k^i)$  in the Hessian and the vector made by  $\bar{\rho}'(x_k^i)$  in the gradient.
- [O3] Based on [O1] and [O2], there is a low-complexity algorithm to compute the gradient vector  $g(x_k)$ .
- [O4]  $\Phi^T\Phi$  and  $H(x_k)$  are dense matrices, but  $H(x_k)$  is positive definite for  $0 \leq \gamma < +\infty$ .
- [O5] There is a low-complexity algorithm to multiply with the Hessian  $H(x_k)$ .

### 5.1.2 Iterative Methods

We choose an iterative method to solve (5.1) because the main work involves matrix-vector multiplication, for which we have fast algorithms, and some vector inner products. For other methods like Cholesky factorization, because of [O4], in general there will be no low-complexity algorithms.

Adopting a general notation, we consider solving a system of linear equations

$$Ax = b. \quad (5.2)$$

When  $A$  is symmetric and positive definite, there are two important iterative methods: CG and MINRES.

**CG** *Conjugate gradient* (CG) methods minimize the  $A$ -norm of the error,  $\|e_k\|_A = \langle A^{-1}b - x_k, b - Ax_k \rangle^{1/2}$ , where  $e_k$  is the error vector at step  $k$ ,  $x_k$  is the solution estimate at

the  $k$ -th iteration.

**MINRES** The *minimum residual* (MINRES) method minimize the Euclidean norm of the residual,  $b - Ax_k$ , at step  $k$ .

In general, [78, Page 92], when the matrix is Hermitian, MINRES is preferred *in theory* because of the following inequality relation [78, Page 94]:

$$\|r_k^C\| = \frac{\|r_k^M\|}{\sqrt{1 - (\|r_k^M\|/\|r_{k-1}^M\|)^2}} \geq \|r_k^M\|,$$

where  $r_k^C$  is the residual at step  $k$  of the CG method and  $r_k^M$  is the residual at step  $k$  in MINRES. In words, at the same step, the Euclidean norm of the residual from CG is always larger than the Euclidean norm of the residual from MINRES.

In practical numerical computing, CG and MINRES will *not* perform as well as we predict under exact arithmetic. We choose a variation of CG—LSQR—which is proven to be more robust in finite-precision computing [117, 116]. For least-squares problems where  $A = B^T B$  and  $b = B^T c$ , applying LSQR to  $\min \|Bx - c\|_2^2$  is analytically equivalent to CG on the normal equations  $B^T Bx = B^T c$ , so in the following discussion about theoretical result, we will only mention CG rather than LSQR.

### 5.1.3 Convergence Rates

A powerful technique in analyzing the convergence rate for both CG and MINRES is the minimax polynomial of eigenvalues. Here we summarize the key results:

[C1] The  $A$ -norm of the error in the CG algorithm for a Hermitian and positive definite matrix  $A$ , and the Euclidean norm of the residual in the MINRES algorithm for a general Hermitian matrix, are minimized over the spaces

$$e_0 + \text{SPAN}\{Ae_0, A^2e_0, \dots, A^k e_0\}$$

and

$$r_0 + \text{SPAN}\{Ar_0, A^2r_0, \dots, A^k r_0\},$$

respectively. These two spaces are called Krylov subspaces, and the corresponding

methods are called Krylov subspace approximations [78, Page 49].

[C2] At step  $k$ , the CG error vector and the MINRES residual vector can be written as

$$\begin{aligned} e_k &= p_k^C(A)e_0, \\ r_k &= p_k^M(A)r_0, \end{aligned}$$

where  $p_k^C$  and  $p_k^M$  are two polynomials with degree no higher than  $k$  that take value 1 at the origin. Moreover,

$$\begin{aligned} \|e_k\|_A &= \min_{p_k} \|p_k(A)e_0\|_A, \\ \|r_k\| &= \min_{p_k} \|p_k(A)r_0\|, \end{aligned}$$

where  $p_k$  is a  $k$ th-degree polynomial with value 1 at the origin.

[C3] Suppose for a Hermitian and positive semidefinite matrix  $A$ ,  $A = U\Lambda U^T$  is its eigen-decomposition, where  $U$  is an orthogonal matrix and  $\Lambda$  is a diagonal matrix. Suppose  $\Lambda = \text{diag}\{\lambda_1, \dots, \lambda_N\}$ . We have sharp bounds for the norms of the error and the residual in CG and MINRES:

$$\|e_k\|_A / \|e_0\|_A \leq \min_{p_k} \max_{i=1,2,\dots,N} |p_k(\lambda_i)|, \quad \text{for CG;} \quad (5.3)$$

$$\|r_k\| / \|r_0\| \leq \min_{p_k} \max_{i=1,2,\dots,N} |p_k(\lambda_i)|, \quad \text{for MINRES.} \quad (5.4)$$

In (5.3) and (5.4), if the eigenvalues are tightly clustered around a single point (away from the origin), then the right-hand sides are more likely to be minimized; hence, iterative methods tends to converge quickly. On the other hand, if the eigenvalues are widely spread, especially if they lie on the both sides of the origin, then the values on the right-hand sides of both inequalities are difficult to be minimized; hence, iterative methods may converge slowly.

Note that the above results are based on exact arithmetic. In finite-precision computation, these error bounds are in general not true, because the round-off errors that are due to finite precision may destroy assumed properties in the methods (e.g., orthogonality). For further discussion, we refer to Chapter 4 of [78] and the references therein.

### 5.1.4 Preconditioner

Before we move into detailed discussion, we would like to point out that the content of this section applies to both CG and MINRES.

When the original matrix  $A$  does not have a good eigenvalue distribution, a preconditioner may help. In our case, we only need to consider the problem as in (5.2) for Hermitian matrices. We consider solving the preconditioned system

$$L^{-1}AL^{-H}y = L^{-1}b, \quad x = L^{-H}y, \quad (5.5)$$

where  $L$  is a preconditioner. When the matrix-vector multiplications associated with  $L^{-H}$  and  $L^{-1}$  have fast algorithms and  $L^{-1}AL^{-H}$  has a *good* eigenvalue distribution, the system in (5.2) can be solved in fewer iterations.

Before giving a detailed discussion about preconditioners, we need to restate our setting. Here  $A$  is the Hessian at step  $k$ :  $A = H(x_k)$ . To simplify the discussion, we assume that  $\Phi$  is a concatenation of several orthogonal matrices:

$$\Phi = [T_1, T_2, \dots, T_m],$$

where  $T_1, T_2, \dots, T_m$  are  $n \times n$  orthogonal square matrices. The number of columns of  $\Phi$  is equal to  $N = mn^2$ . Hence from (4.8),

$$\frac{1}{2}A = \begin{bmatrix} I + D_1 & T_1^T T_2 & \cdots & T_1^T T_m \\ T_2^T T_1 & I + D_2 & & \\ \vdots & & \ddots & \vdots \\ T_m^T T_1 & & \cdots & I + D_m \end{bmatrix}, \quad (5.6)$$

where

$$D_i = \frac{1}{2}\lambda \begin{pmatrix} \bar{\rho}''(x_k^{1+(i-1)n^2}) & & & \\ & \ddots & & \\ & & \bar{\rho}''(x_k^{n^2+(i-1)n^2}) & \\ & & & \ddots \end{pmatrix}, \quad i = 1, 2, \dots, m,$$

are diagonal matrices. To simplify (5.6), we consider  $A$  left and right multiplied by a block

diagonal matrix and its transpose as follows:

$$\begin{aligned}\tilde{A} &= \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_m \end{pmatrix} \frac{1}{2}A \begin{pmatrix} T_1^T & & & \\ & T_2^T & & \\ & & \ddots & \\ & & & T_m^T \end{pmatrix} \\ &= \begin{pmatrix} I + S_1 & I & \cdots & I \\ I & I + S_2 & & \\ \vdots & & \ddots & \vdots \\ I & & \cdots & I + S_m \end{pmatrix},\end{aligned}$$

where  $S_i = T_i D_i T_i^T$ ,  $i = 1, 2, \dots, m$ . In the remainder of this section, we consider what a good preconditioner for  $\tilde{A}$  should be.

We considered three possible preconditioners:

1. a preconditioner from complete Cholesky factorization,
2. a preconditioner from sparse incomplete factorization of the inverse [9],
3. a diagonal block preconditioner [4, 5, 31].

The main result is that we found the preconditioner 1 and preconditioner 2 are not “optimal”. Here “optimal” means that the matrix-vector multiplication with the matrix associated with the preconditioner  $L^{-H}$  should still have fast algorithms, and these fast algorithms should be based on fast algorithms for matrix-vector multiplication for the matrices  $T_1, T_2, \dots, T_m$ . So the block diagonal preconditioner is the only one we are going to use. We describe the results about the block diagonal preconditioner here, and postpone the discussion about the preconditioners in case 1 and 2 to Section 5.4.

The optimal block diagonal preconditioner for  $\tilde{A}$  is

$$\begin{pmatrix} (I + S_1)^{-1/2} & & & \\ & (I + S_2)^{-1/2} & & \\ & & \ddots & \\ & & & (I + S_m)^{-1/2} \end{pmatrix}. \quad (5.7)$$

A striking result is due to Demmel [78, Page 168, Theorem 10.5.3]. The key idea is that

among all the block diagonal preconditioners, the one that takes the Cholesky factorizer of the diagonal submatrix as its diagonal submatrix is *nearly* optimal. Here “nearly” means that the resulting condition number cannot be larger than  $m$  times the best achievable condition number by using a block diagonal preconditioner. Obviously, we have fast algorithms to multiply with matrix  $(I + S_i)^{-1/2}$ ,  $i = 1, 2, \dots, m$ .

## 5.2 LSQR

### 5.2.1 What is LSQR?

LSQR [117, 116] solves the following two least-squares (LS) problems, depending on whether the damping parameter  $\alpha$  is zero or not.

[N] When the damping parameter is equal to zero ( $\alpha = 0$ ), solve  $Ax = b$  or minimize $_x \|b - Ax\|^2$ .

[R] When the damping parameter is *not* equal to zero ( $\alpha \neq 0$ ):

$$\text{minimize}_x \left\| \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \alpha I \end{pmatrix} x \right\|_2^2.$$

Here [N] is a nonregularized problem and [R] is a regularized problem.

The problem in (5.1) can be rewritten as follows:

$$\left[ 2\Phi^T\Phi + \lambda \begin{pmatrix} \bar{\rho}''(x_1^k) & & \\ & \ddots & \\ & & \bar{\rho}''(x_N^k) \end{pmatrix} \right] d = 2\Phi^T(y - \Phi x_k) - \lambda \begin{pmatrix} \bar{\rho}'(x_1^k) \\ \vdots \\ \bar{\rho}'(x_N^k) \end{pmatrix},$$

where  $d$  is the Newton direction that we want to solve for and the remaining variables are defined in the previous chapter. The above equation is equivalent to solving an LS problem:

$$\text{(LS)} : \quad \text{minimize}_d \left\| \begin{bmatrix} \Phi \\ D(x_k) \end{bmatrix} d - \begin{bmatrix} y - \Phi x_k \\ -D^{-1}(x_k)g(x_k) \end{bmatrix} \right\|_2,$$

where

$$D(x_k) = \sqrt{\lambda/2} \begin{pmatrix} \bar{\rho}''(x_1^k) & & \\ & \ddots & \\ & & \bar{\rho}''(x_N^k) \end{pmatrix}^{1/2},$$

and

$$g(x_k) = \frac{\lambda}{2} \begin{pmatrix} \bar{\rho}'(x_1^k) \\ \vdots \\ \bar{\rho}'(x_N^k) \end{pmatrix}.$$

Note that this is the [N] case. Recall  $\bar{\rho}'$  and  $\bar{\rho}''$  are defined in (4.4) and (4.5).

We can transfer problem (LS) into a damped LS problem by solving for a shifted variable  $\bar{d}$ :  $\delta\bar{d} = D(x_k)d + D^{-1}(x_k)g(x_k)$ . The problem (LS) becomes

$$(\text{dLS}) : \quad \underset{\bar{d}}{\text{minimize}} \left\| \begin{bmatrix} \Phi D^{-1}(x_k)\delta \\ \delta I \end{bmatrix} \bar{d} - \begin{bmatrix} y - \Phi(x_k - D^{-2}(x_k)g(x_k)) \\ 0 \end{bmatrix} \right\|_2.$$

This is the [R] case. This method is also called diagonal preconditioning. Potentially, it will turn the original LS problem into one with clustered singular values, so that LSQR may take fewer iterations. LSQR also works with shorter vectors (as it handles the  $\delta I$  term implicitly).

### 5.2.2 Why LSQR?

Since the Hessian in (5.1) is at least semidefinite, we prefer a conjugate-gradients type method. LSQR is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties. Particularly when the Hessian is ill-conditioned—which is very likely to happen when the iteration gets closer to the convergence point—LSQR is more stable than the standard CG. But LSQR costs slightly more storage and work per iteration.

### 5.2.3 Algorithm LSQR

A formal description of LSQR is given in [117, page 50]. We list it here for the convenience of readers.

Algorithm LSQR: to minimize  $\|b - Ax\|_2$ .

1. Initialize.

$$\beta_1 u_1 = b, \alpha_1 v_1 = A^T u_1, w_1 = v_1, x_0 = 0, \bar{\phi}_1 = \beta_1, \bar{\rho}_1 = \alpha_1.$$

2. For  $i = 1, 2, 3, \dots$

(a) Continue the bidiagonalization.

- i.  $\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$
- ii.  $\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$ .

(b) Construct and apply next orthogonal transformation.

- i.  $\rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$
- ii.  $c_i = \bar{\rho}_i / \rho_i$
- iii.  $s_i = \beta_{i+1} / \rho_i$
- iv.  $\theta_{i+1} = s_i \alpha_{i+1}$
- v.  $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$
- vi.  $\phi_i = c_i \bar{\phi}_i$
- vii.  $\bar{\phi}_{i+1} = s_i \bar{\phi}_i$ .

(c) Update  $x, w$ .

- i.  $x_i = x_{i-1} + (\phi_i / \rho_i) w_i$
- ii.  $w_{i+1} = v_{i+1} - (\theta_{i+1} / \rho_i) w_i$ .

(d) Test for convergence. Exit if some stopping criteria have been met.

### 5.2.4 Discussion

There are two possible dangers in the previous approaches (LS) and (dLS). They are both caused by the existence of large entries in the vector  $D^{-1}(x_k)g(x_k)$ . The first danger occurs in (LS), when  $D^{-1}(x_k)g(x_k)$  is large, the right-hand side is large even though the elements of  $d$  will be small as Newton's method converges. Converting (5.1) to (LS) is a

somewhat unstable transformation. The second danger occurs in (dLS): when an entry of  $D^{-1}(x_k)g(x_k)$  is large, because  $d = D(x_k)^{-1}\bar{d} - D^{-2}(x_k)g(x_k)$ , there must be “catastrophic cancellation” in some of the elements of  $d$  as the iterative method converges.

As we know, the  $j$ th entry of  $D^{-1}(x_k)g(x_k)$  is

$$\begin{aligned}\sqrt{\lambda/2}\bar{\rho}'(x_j^k)/\sqrt{\bar{\rho}''(x_j^k)} &= \sqrt{\lambda/2}\frac{1}{\gamma}\text{sign}(x_j^k)\left(e^{-\gamma|x_j^k|} - 1\right)/e^{-\gamma|x_j^k|/2} \\ &= \sqrt{\lambda/2}\frac{1}{\gamma}\text{sign}(x_j^k)\left(e^{-\frac{\gamma}{2}|x_j^k|} - e^{\frac{\gamma}{2}|x_j^k|}\right).\end{aligned}$$

Since  $\gamma$  usually is large in our problem, when  $|x_j^k|$  is significantly nonzero, the  $j$ th entry of  $D^{-1}(x_k)g(x_k)$  is big. It is unavoidable to have significantly nonzero  $|x_j^k|$ . But hopefully the images that we consider have intrinsic sparse decompositions, hence the proportion of significantly nonzero entries in  $|x_j^k|$  is small and the numerical Newton’s direction is still accurate enough, so that our iterative algorithm will still converges to the minimum.

Solving either (LS) or (dLS) gives an approach to finding a Newton’s direction. It is hard to tell from theory which method works better. In our numerical experiments of image decompositions, we choose to solve (LS).

## 5.3 MINRES

In this section, we describe the MINRES algorithm and its preconditioned version because MINRES has a nice theoretical property in reducing the residual norm monotonically, and it avoids the large numbers involved in transforming (5.1) to (LS) or (dLS). We did not implement MINRES, but a comparison with LSQR on our problem might be an interesting research topic.

### 5.3.1 Algorithm MINRES

A formal description of MINRES is given on page 44 of [78]. We list it for the convenience of the readers.

Algorithm: MINRES

1. Given  $x_0$ , compute  $r_0 = b - Ax_0$  and set  $q_1 = r_0/\|r_0\|$ . Initialize  $\xi = (1, 0, \dots, 0)^T$ ,  $\beta = \|r_0\|$ .

2. For  $k = 1, 2, \dots$ ,

(a) Compute  $q_{k+1}$ ,  $\alpha_k = T(k, k)$  and  $\beta_k = T(k+1, k) = T(k, k+1)$  using the Lanczos algorithm:

$$\begin{aligned} \tilde{q}_{k+1} &= Aq_k - \beta_{k-1}q_{k-1}. \\ \text{Set } \alpha_k &= \langle \tilde{q}_{k+1}, q_k \rangle, \\ \tilde{q}_{k+1} &\leftarrow \tilde{q}_{k+1} - \alpha_k q_k. \\ \beta_k &= \|\tilde{q}_{k+1}\|, \\ q_{k+1} &= \tilde{q}_{k+1}/\beta_k. \end{aligned}$$

(b) Work on the last column of  $T$ , that is:

If  $k > 2$ , then

$$\begin{pmatrix} T(k-2, k) \\ T(k-1, k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-2} & s_{k-2} \\ -\bar{s}_{k-2} & c_{k-2} \end{pmatrix} \begin{pmatrix} 0 \\ T(k-1, k) \end{pmatrix}.$$

If  $k > 1$ , then

$$\begin{pmatrix} T(k-1, k) \\ T(k, k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-1} & s_{k-1} \\ -\bar{s}_{k-1} & c_{k-1} \end{pmatrix} \begin{pmatrix} T(k-1, k) \\ T(k, k) \end{pmatrix}.$$

Set

$$\begin{aligned} a &= |T(k, k)| / (|T(k, k)| + |T(k+1, k)|), \\ b &= 1 - a, \\ c_k &= a / \sqrt{a^2 + b^2}, \\ \bar{s}_k &= c_k T(k+1, k) / T(k, k). \end{aligned}$$

Apply the  $k$ -th rotation to  $\xi$  and to the last column of  $T$ :

$$\begin{aligned} \begin{pmatrix} \xi(k) \\ \xi(k+1) \end{pmatrix} &\leftarrow \begin{pmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{pmatrix} \begin{pmatrix} \xi(k) \\ 0 \end{pmatrix}, \\ T(k, k) &\leftarrow c_k T(k, k) + s_k T(k+1, k), \\ T(k+1, k) &\leftarrow 0. \end{aligned}$$

- (c) Compute  $p_{k-1} = [q_k - T(k-1, k)p_{k-2} - T(k-2, k)p_{k-3}] / T(k, k)$ , where undefined terms are zeros for  $k < 2$ .
- (d) Set  $x_k = x_{k-1} + a_{k-1}p_{k-1}$ , where  $a_{k-1} = \beta\xi(k)$ .

### 5.3.2 Algorithm PMINRES

MINRES can be used with a block diagonal preconditioner as specified in (5.7). A preconditioned MINRES is called PMINRES. Going back to (5.5), we have

$$L^{-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} (I + S_1)^{-1/2} & & & \\ & (I + S_2)^{-1/2} & & \\ & & \ddots & \\ & & & (I + S_m)^{-1/2} \end{pmatrix} \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_m \end{pmatrix}$$

And then

$$\begin{aligned} M &= LL^H \\ &= 2 \begin{pmatrix} I + D_1 & & & \\ & I + D_2 & & \\ & & \ddots & \\ & & & I + D_m \end{pmatrix}. \end{aligned}$$

Note that the preconditioner  $M$  is actually a diagonal matrix.

As an interesting reference, we list the preconditioned MINRES (also in [78, Page 122]) algorithm here.

#### Algorithm: PMINRES

1. Given  $x_0$ , compute  $r_0 = b - Ax_0$  and solve  $Mz_0 = r_0$ . Set  $\beta = \langle r_0, z_0 \rangle^{1/2}$ ,  $q_1 = r_0/\beta$ , and  $w_1 = z_0/\beta$ . Initialize  $\xi = (1, 0, \dots, 0)^T$ .
2. For  $k = 1, 2, \dots$ ,
  - (a) Compute  $q_{k+1}, w_{k+1}$ ,  $\alpha_k = T(k, k)$  and  $\beta_k = T(k+1, k) = T(k, k+1)$

using the preconditioned Lanczos algorithm:

$$\begin{aligned} \text{Set } v_k &= Aw_k - \beta_{k-1}q_{k-1}, \\ \alpha_k &= \langle v_k, w_k \rangle, \\ v_k &\leftarrow v_k - \alpha_k q_k, \\ \text{Solve } M\tilde{w}_{k+1} &= v_k, \end{aligned}$$

Set  $q_{k+1} = v_k/\beta_k$  and  $w_{k+1} = \tilde{w}_{k+1}/\beta_k$ , where  $\beta_k = \langle v_k, \tilde{w}_{k+1} \rangle^{1/2}$ .

(b) Work on the last column of  $T$ , that is:

If  $k > 2$ , then

$$\begin{pmatrix} T(k-2, k) \\ T(k-1, k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-2} & s_{k-2} \\ -\bar{s}_{k-2} & c_{k-2} \end{pmatrix} \begin{pmatrix} 0 \\ T(k-1, k) \end{pmatrix}.$$

If  $k > 1$ , then

$$\begin{pmatrix} T(k-1, k) \\ T(k, k) \end{pmatrix} \leftarrow \begin{pmatrix} c_{k-1} & s_{k-1} \\ -\bar{s}_{k-1} & c_{k-1} \end{pmatrix} \begin{pmatrix} T(k-1, k) \\ T(k, k) \end{pmatrix}.$$

Set

$$\begin{aligned} a &= |T(k, k)| / (|T(k, k)| + |T(k+1, k)|), \\ b &= 1 - a, \\ c_k &= a / \sqrt{a^2 + b^2}, \\ \bar{s}_k &= c_k T(k+1, k) / T(k, k). \end{aligned}$$

Apply the  $k$ -th rotation to  $\xi$  and to the last column of  $T$ :

$$\begin{aligned} \begin{pmatrix} \xi(k) \\ \xi(k+1) \end{pmatrix} &\leftarrow \begin{pmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{pmatrix} \begin{pmatrix} \xi(k) \\ 0 \end{pmatrix}, \\ T(k, k) &\leftarrow c_k T(k, k) + s_k T(k+1, k), \\ T(k+1, k) &\leftarrow 0. \end{aligned}$$

- (c) Compute  $p_{k-1} = [w_k - T(k-1, k)p_{k-2} - T(k-2, k)p_{k-3}] / T(k, k)$ , where undefined terms are zeros for  $k < 2$ .
- (d) Set  $x_k = x_{k-1} + a_{k-1}p_{k-1}$ , where  $a_{k-1} = \beta\xi(k)$ .

## 5.4 Discussion

### 5.4.1 Possibility of Complete Cholesky Factorization

We argue that a preconditioner based on a complete Cholesky factorization of  $\tilde{A}$  is not optimal, because the resulting preconditioner may not have fast matrix-vector multiplication.

Suppose we have the Cholesky factorization  $\tilde{A} = LL^T$ , here  $L$  denotes a lower triangular matrix with elements (could be block matrices)  $a_{ij}$ :

$$L = \begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ \dots & & & \ddots \end{pmatrix}.$$

Here the dimension is not important. We have

$$\begin{aligned} a_{11}a_{11}^T &= I + S_1, \\ a_{11}a_{12}^T &= I, \\ &\dots \\ a_{22}a_{22}^T + a_{21}a_{21}^T &= I + S_2, \\ &\dots \end{aligned}$$

Obviously,

$$\begin{aligned} a_{11} &= (I + S_1)^{1/2}, \\ a_{12} &= (I + S_1)^{-1/2}, \\ \text{and } a_{22} &= (I + S_2 - (I + S_1)^{-1})^{-1/2}. \end{aligned}$$

$a_{22}$  is from the Cholesky factorization of the Schur complement of the first  $2 \times 2$  block.

From

$$\begin{aligned}(I + S_1)^{1/2} &= T_1(I + D_1)^{1/2}T_1^T, \\ (I + S_1)^{-1/2} &= T_1(I + D_1)^{-1/2}T_1^T,\end{aligned}$$

and there are fast algorithms to implement matrix-vector multiplication with matrix  $T_1$ ,  $T_1^T$ ,  $(I + D_1)^{-1/2}$  and  $(I + D_1)^{1/2}$ , so there are fast algorithms to multiply with  $a_{11}$  and  $a_{12}$ . But for  $a_{22}$ , none of  $T_1$ , inverse of  $T_1$ ,  $T_2$  and inverse of  $T_2$  can simultaneously diagonalize  $I + S_2$  and  $(I + S_1)^{-1}$ . Hence there is no trivial fast algorithm to multiply with matrix  $a_{22}$ . In general, a complete Cholesky factorization will destroy the structure of matrices from which we can have fast algorithms. The fast algorithms of matrix-vector multiplication are so vital for solving large-scale problems with iterative methods (and an intrinsic property of our problem is that the size of data is huge) that we do not want to sacrifice the existence of fast algorithms. Preconditioning may reduce the number of iterations, but the amount of computation within each iteration is increased significantly, so overall, the total amount of computing may increase. Because of this philosophy, we stop plodding in the direction of complete Cholesky factorization.

### 5.4.2 Sparse Approximate Inverse

The idea of a sparse approximate inverse (SAI) is that if we can find an approximate eigendecomposition of the inverse matrix, then we can use it to precondition the linear system. More precisely, suppose  $Z$  is an orthonormal matrix and at the same time the columns of  $Z$ , denoted by  $z_i$ ,  $i = 1, 2, \dots, N$ , are  $\tilde{A}$ -conjugate orthogonal to each other:

$$Z = [z_1, z_2, \dots, z_N], \quad \text{and} \quad Z\tilde{A}Z^T = D,$$

where  $D$  is a diagonal matrix. We have  $\tilde{A} = Z^T D Z$  and  $\tilde{A}^{-1} = Z^T D^{-1} Z$ . If we can find such a  $Z$ , then  $(D^{-1/2} Z)\tilde{A}(D^{-1/2} Z)^T \approx I$ , so  $(D^{-1/2} Z)$  is a good preconditioner.

We now consider its block matrix analogue. Actually in the previous subsection, we have already argued that when we have the block matrix version of a preconditioner, each element of the block matrix should be able to be associated with a fast algorithm that is based on matrix multiplication with matrices  $T_1, T_2, \dots, T_m$  and matrix-vector multiplication with diagonal matrices. Unfortunately, a preconditioner derived by using the idea of

eigendecomposition of the inverse of  $\tilde{A}$  may not satisfy this criterion. To see this, let's look at a  $2 \times 2$  block matrix. Suppose

$$\tilde{A} = \begin{pmatrix} S_1 + I & I \\ I & S_2 + I \end{pmatrix}$$

can be factorized as

$$\tilde{A} = \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} \begin{pmatrix} \lambda_{11}^T & \lambda_{21}^T \\ \lambda_{12}^T & \lambda_{22}^T \end{pmatrix},$$

where matrix

$$\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix}$$

is an orthogonal matrix and  $D_1$  and  $D_2$  are diagonal matrices. We have

$$\tilde{A}^{-1} = \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} \begin{pmatrix} D_1^{-1} & \\ & D_2^{-1} \end{pmatrix} \begin{pmatrix} \lambda_{11}^T & \lambda_{21}^T \\ \lambda_{12}^T & \lambda_{22}^T \end{pmatrix}.$$

At the same time,

$$\begin{pmatrix} (S_1 + I)^{-1} & \\ & (S_2 + I - (S_1 + I)^{-1})^{-1} \end{pmatrix} = \begin{pmatrix} I & (S_1 + I)^{-1} \\ & I \end{pmatrix} \tilde{A}^{-1} \begin{pmatrix} I & \\ & (S_1 + I)^{-1} \end{pmatrix}.$$

Hence

$$(S_2 + I - (S_1 + I)^{-1})^{-1} = \lambda_{21} D_1^{-1} \lambda_{21}^T + \lambda_{22} D_2^{-1} \lambda_{22}^T.$$

So if there were fast algorithms to multiply with matrices  $\lambda_{21}$ ,  $\lambda_{22}$  and their transposes, then there would be a fast algorithm to multiply with matrix  $(S_2 + I - (S_1 + I)^{-1})^{-1}$ . But we know that in general, there is no fast algorithm to multiply with this matrix (see also the previous subsection). So in general, such a eigendecomposition will not give a block preconditioner that has fast algorithms to multiply with its block elements. Hence we proved that at least in the  $2 \times 2$  case, an eigendecomposition of inverse approach is not favored.

## Chapter 6

# Simulations

Section 6.1 describes the dictionary that we use. Section 6.2 describes our testing images. Section 6.3 discusses the decompositions based on our approach and its implication. Section 6.4 discusses the decay of amplitudes of coefficients and how it reflects the sparsity in representation. Section 6.5 reports a comparison with Matching Pursuit. Section 6.6 summarizes the computing time. Section 6.7 describes the forthcoming software package that is used for this project. Finally, Section 6.8 talks about some related efforts.

### 6.1 Dictionary

The dictionary we choose is a combination of an orthonormal 2-D wavelet basis and a set of edgelet-like features.

2-D wavelets are tensor products of two 1-D wavelets. We choose a type of 1-D wavelets that have a minimum size support for a given number of vanishing moments but are as symmetrical as possible. This class of 1-D wavelets is called “Symmlets” in WaveLab [42]. We choose the Symmlets with 8 vanishing moments and size of the support being 16. An illustration of some of these 2-D wavelets is in Figure 3.6.

Our “edgelet dictionary” is in fact a collection of edgelet features. See the discussion of Sections 3.3.1–3.3.3. In Appendix B, we define a collection of linear functionals  $\tilde{\lambda}_e[x]$  operating on  $x$  belonging to the space of  $N \times N$  images. These linear functionals are associated with the evaluation of an approximate Radon transform as described in Appendix B. In effect, the Riesz representers of these linear functionals,  $\{\tilde{\psi}_e(k_1, k_2) : 0 \leq k_1, k_2 < N\}$ ,

defined by the identity,

$$\tilde{\lambda}_e[I] = \sum_{k_1, k_2} I_{k_1, k_2} \tilde{\psi}_e(k_1, k_2),$$

gives a collection of “thickened edgelets”, i.e., line segments digitally sampled and of thickness a few pixels wide. Our “edgelet dictionary” is precisely this collection of representers. We will call this an edgelet dictionary even though edgelets have been previously defined in Section 3.3.1 as line segments in  $\mathbb{R}^2$  rather than vectors  $\{\tilde{\psi}_e(k_1, k_2) : 0 \leq k_1, k_2 < N\}$ ; we hope this abuse of terminology will not confuse most readers. An illustration of some of these representers can be found in Figure B.3. An obvious drawback of this set of features is that they have roughly the same width. This property stops the set from being tight and makes it incapable of representing fine scale image components. Developing a similar set of linear features with various width will be an interesting research topic. See Section 7.2.

The two sets are chosen because they possess some interesting phenomena in the images and there are fast algorithms to carry out their discrete version transforms. At this stage, we are mainly interested in point singularities and linear singularities in an image.

## 6.2 Images

We test our approach on four images shown in Figure 6.1:

- **Car**: “In The Car”, a painting by Roy Lichtenstein, 1963;
- **Pentagon**: a pentagon;
- **Overlap**: overlapping point singularity and linear singularity;
- **Separate**: separated point singularity and linear singularity.

They were chosen because:

- [1] They have the features that we want to test with our dictionary. Our dictionary is made by 2-D wavelets and edgelet-like features. 2-D wavelets resemble point singularities and edgelet-like features resemble linear singularities. These features are the key image components that we are interested in, and our approach should find them.

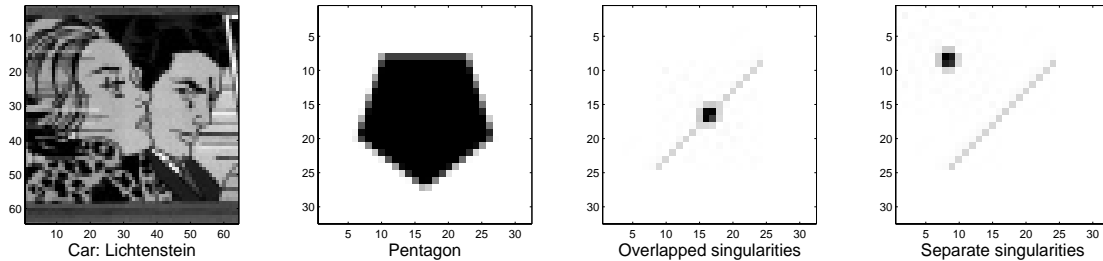


Figure 6.1: Four test images.

- [2] They are simple but sufficient to examine whether our basic assumption is true—that different transforms will automatically represent the corresponding features in a sparse image decomposition.

### 6.3 Decomposition

A way to test whether this approach works is to see how it decomposes the image into parts associated with included transforms. Our approach presumably provides a *global* sparse representation. Because of the global sparsity, if we reconstruct part of an image by using only coefficients associated with a certain transform, then this partial reconstruction should be a superposition of a few atoms (from the dictionary made by representers of the transform). So we expect to see features attributable to the associated transforms in these partial reconstructions.

We consider decomposing an image into two parts—wavelet part and edgelet part. In principle, we expect to see points and patches in the wavelet part and lines in the edgelet part. Figure 6.2 shows decompositions of the four testing images. The first row is for **Car**. The second, third and fourth row are for **Pentagon**, **Overlap** and **Separate** respectively. In each row, from the left, the first squared sub-image is the original, the second is the wavelet part of the image, the third is the edgelet part of the image, and the last is the superposition of the wavelet part and the edgelet part. With appropriate parameter  $\lambda$ , the

fourth sub-image should be close to the original.

We have some interesting observations:

1. Overall, we observe the wavelet parts possess features resembling points (fine scale wavelets) and patches (coarse scale scaling functions), while the edgelet parts possess features resembling lines. This is most obvious in **Car** and least obvious in **Pentagon**. The reason could be that **Pentagon** does not contain many linear features. (Boundaries are not lines.)
2. We observe some artifacts in the decompositions. For example, in the wavelet part of **Car**, we see a lot of fine scale features that are not similar to points, but are similar to line segments. This implies that they are made by many small wavelets. The reason for this is the intrinsic disadvantage of the edgelet-like transform that we have used. As in Figure B.3, the representers of our edgelet-like transform have a fixed width. This prevents it from efficiently representing narrow features. The same phenomena emerge in **Overlap** and **Separate**. A way to overcome it is to develop a transform whose representers have not only various locations, lengths and orientations, but also various widths. This will be an interesting topic of future research.

## 6.4 Decay of Coefficients

As we discussed in Chapter 2, one can measure sparsity of representation by studying the decay of the coefficient amplitudes. The faster the decay of coefficient amplitudes, the sparser the representation.

We compare our approach with two other approaches. One is an approach using merely the 2-D DCT; the other is an approach using merely the discrete 2-D wavelet transform. The reasons we choose to compare with these transforms:

1. The 2-D DCT, especially the one localized to  $8 \times 8$  blocks, has been applied in an important industry standard for image compression and transmission known as JPEG.
2. The 2-D wavelet transform is a modern alternative to 2-D DCT. In some developing industry standards (e.g., draft standards for JPEG-2000), the 2-D wavelet transform has been adopted as an option. It has been proven to be more efficient than 2-D DCT in many cases.

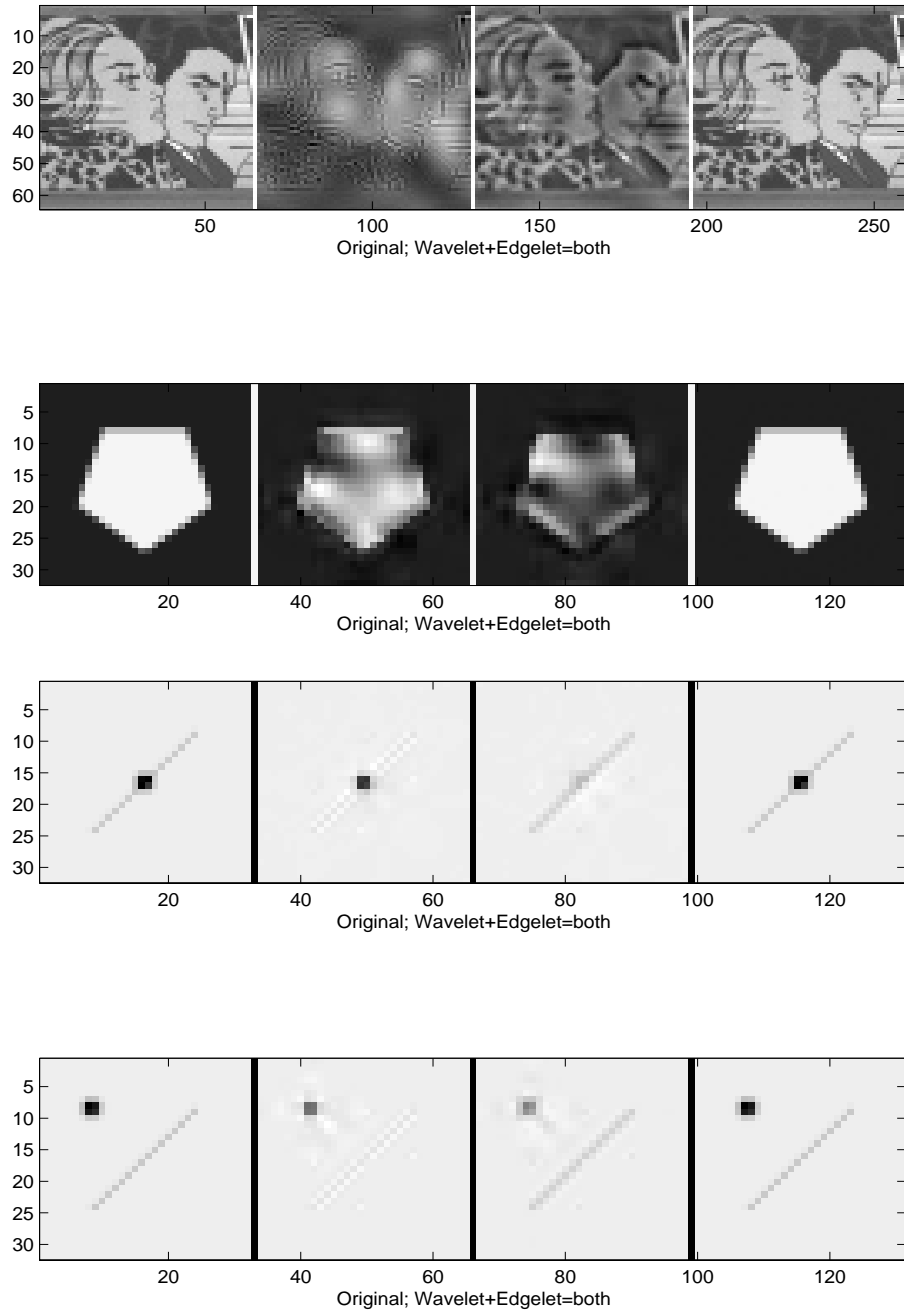


Figure 6.2: Decompositions.

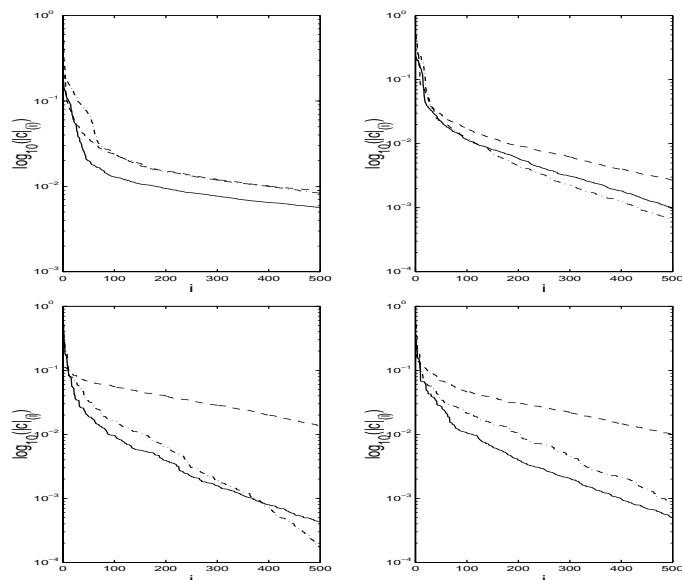


Figure 6.3: Decaying amplitudes of coefficients.

The goal of this thesis is to find a sparse representation of an image. We aim to see if our combined approach will lead to a sparser representation than these existing approaches.

Figure 6.3 shows the decay of the amplitudes of coefficients from three approaches on four images. On each plot, the horizontal axis is the order index of the amplitudes of the coefficients and the vertical axis is the logarithm of the amplitude (in base 10). The amplitudes are sorted from largest to smallest. The dashed lines “- -” illustrate the DCT-only approach; the solid lines denote our combined (wavelets+edgelet-like features) approach; the dash and dotted lines “- . . . -” illustrate the wavelet-only approach. From left to right and upper to lower, these plots give results for **Car**, **Pentagon**, **Overlap** and **Separate**.

We have the following observations:

1. Our combined approach tends to give the sparsest representation (in the sense of the fastest decay of the amplitudes) among all the three approaches. This is particularly clear in the case of **Car** and **Separate**. But in some cases the wavelet-only approach provides very competitive results—for example, for **Pentagon** and **Overlap**. Actually, in **Pentagon** and **Overlap**, we observe that the curve associated with the wavelet-only

approach ultimately falls well below the curve associated with the combined approach. (This seems to imply that the wavelet-only approach gives a sparser asymptotic representation.)

Particularly, it is important to remember that amplitudes are plotted on a logarithmic scale. A big drop at the beginning can not be shown significantly. The advantage of combined representation at very high levels of compression is difficult to discuss in this display. Note that the curve associated with the wavelet-only approach falls below the curve associated with the combined approach when the order index is large. If we study carefully the first few largest amplitudes, the curve associated with the combined approach goes down faster. This fact is evident for `Overlap`. As we know, in the compression or sparse representation, the decay of the first few biggest amplitudes is the most important factor, so we think our combined approach still gives better results, even for `Pentagon` and `Overlap`.

2. The DCT only approach always gives the least sparse coefficients. As we have mentioned, the DCT is good for images with homogeneous components. Unfortunately, in our examples, none of them seems to have a high proportion of homogeneous components. This may explain why here DCT is far from optimal.
3. For more “realistic” images, as in the case of `Car`, the wavelet-only approach works only as well as the DCT-only approach, but our combined approach is significantly better. This may imply that our approach is better-suited for natural images than existing methods. Of course more careful study and extensive experiments are required to verify this statement.

## 6.5 Comparison with Matching Pursuit

Our approach is a global approach, in the sense that we minimize a global objective function. Our method is computationally expensive. A potentially cheaper method is Matching Pursuit (MP) [102]. MP is a greedy algorithm. In a Hilbert space, MP at every step picks up the atom that is the most correlated with the residual at that step. But MP runs the danger of being trapped by unfortunate choices at early steps into badly suboptimal decompositions [27, 40, 25]. We examine the decay of the amplitudes of coefficients for both MP and our approach.

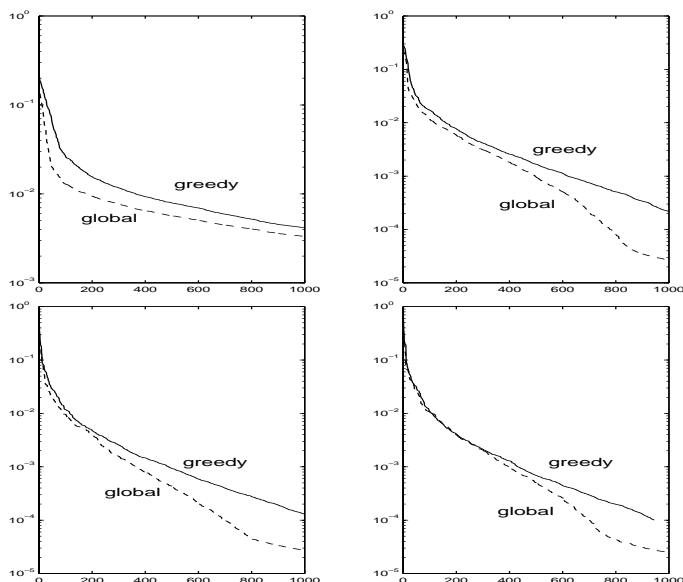


Figure 6.4: A global algorithm vs. Matching Pursuit.

Figure 6.4 shows the decay of the amplitudes of the coefficients from both of the two approaches. In these plots, the solid lines always correspond to the MP, and these dashed lines correspond to our (minimizing the  $\ell^1$  norm) approach. From upper row to lower row, left to right, the plots are for **Car**, **Pentagon**, **Overlap** and **Separate**.

We have the following observations:

1. In all four cases, our global approach always provides a sparser representation than MP does: the decay of the amplitudes of coefficients for our approach is faster than for MP. This validates our belief that a global optimization scheme should often provide a sparser representation than one from a greedy algorithm.
2. If we adopt the idea that the decay of the amplitudes of coefficients at the beginning is important, then our global approach shows the largest advantage in the example of a realistic image (case **Car**). This is promising because it may imply that our approach is more *adaptable* to real images.
3. We observe that our approach achieves a slightly greater advantage for **Overlap** than for **Separate**. As we know, MP is good for separated features but not for overlapped ones. This belief is confirmed here.

## 6.6 Summary of Computational Experiments

We ran our experiments on an SGI Power Challenger server with 196 MHz CPU. In the algorithm in Chapter 4, we start with  $\gamma = 10^2$  and stop with  $\gamma = 10^5$ . Table 6.1 gives the number of LSQR iterations and the execution time of one iteration in each case. Each LSQR iteration includes one analysis transform and one synthesis transform of an image. Since the number of LSQR iterations is machine independent, it is a good criterion for comparison. The last column lists the total execution time.

Image	Size	No. of Iterations	Execution time of each iteration (seconds)	Total execution time (hours)
Car	$64 \times 64$	226881	0.80	50.2
Pentagon	$32 \times 32$	83928	0.33	7.6
Overlap	$32 \times 32$	104531	0.40	11.7
Separate	$32 \times 32$	266430	0.39	28.9

Table 6.1: Table of Running Times

In the above simulations, the tolerance parameter for LSQR is chosen to be  $10^{-10}$  [117]. The minimum tolerance for Newton’s method is chosen to be  $10^{-7}$ . (The Newton iteration will stop if the norm of the gradient vector is less than  $10^{-7}$ .)

## 6.7 Software

We plan to develop a software package (perhaps named “ImageAtomizer”, following Shaobing Chen’s “Atomizer” package). At present it includes the following directories:

- **Datasets**—data files of images and of Matlab functions providing access to them.
- **MEXSource**—CMEX source code for the direct edgelet transform, the fast approximate edgelet transform, 2-D DCT, and more.
- **Methods**—Matlab functions carrying out our global minimum  $\ell^1$  norm algorithm, Matching Pursuit, Method of Frames, together with some supporting files.
- **Transforms**—Matlab functions for the curvelet transform, DCT, edgelet, etc.

- **Utilities**—supporting files, for example, functions to read images and functions to analyze results.
- **Workouts**—scripts to produce results for my thesis and some conference papers.

This package depends heavily on WaveLab [42] and we may integrate it as part of WaveLab.

## 6.8 Scale of Efforts

This project was started near the end of 1997.<sup>1</sup> Extensive efforts have been spent in the development of fast code for the edgelet transform and the implementation and adaptation of iterative algorithms. The code that carries out the edgelet-like transform is completely written in C, which should give us a tremendous gain in speed for numerical computing. The formulation of our approach to the minimum  $\ell^1$  norm problem is new. It is the first time that LSQR has been implemented for this problem. (Chen, Donoho and Saunders mention LSQR in their paper, but did not implement it [27].)

---

<sup>1</sup>An unofficial version: We started this project about two years ago. As usual, it was mixed with countless useful and useless diversions and failures. During the past two years, I have written tens of thousands of lines of C code and thousands of Matlab functions. There were many exciting and sleep-deprived nights. Unfortunately, only a small proportion of the effort became the work that is presented in this thesis.

## Chapter 7

# Future Work

In the future, there are three promising directions that we should explore. Section 7.1 discusses more experiments that we should try but, due to the time constraints, we have not yet performed. Section 7.2 discusses how to apply the filtering idea from multiresolution analysis and the idea from monoscale orthonormal ridgelets to design a system with representers having various widths. Section 7.3 talks about how to use the *block coordinate relaxation* to accelerate our iterative algorithm.

### 7.1 Experiments

So far we have experimented on four images. We are restricted by the fact that each experiment takes a long time to compute. To see if our approach gives sparse atomic decomposition in more general cases, we should test it on more images. Some sets of images that may be useful:

- Textures images [2];
- Aerials images [2];
- Some images favored by the image processing community, for example, “barbara”, “lenna”, etc. [2];
- Some military images—for example, sensor data [1];
- Biological images, for example, some microscopic images of tissues.

These images possess a variety of features. The results of a sparse decomposition can be used to determine which class of features is dominant in an image, and then to tell which transform is well-suited to processing the image.

We are going to experiment on different dictionaries. We have experimented on a dictionary that is a combination of 2-D wavelets and edgelet-like features. Some other possible combinations are:

- {2-D DCT + 2-D wavelets + edgelets}. This dictionary contains elements for homogeneous image components, point singularities and linear singularities. It should provide sparser decomposition, but will increase the computational cost.
- {2-D DCT + 2-D wavelets + curvelets}. It is similar to the idea for the previous dictionary, but with a better-designed curvelets set in place of the edgelets set, it may lead to an improvement in the computational efficiency.
- {2-D DCT + curvelets}. Comparing with the previous result, we may be able to tell how important the wavelet components are in the image, by knowing how many wavelets we need in sparsely representing the desired image. We can explore the same idea for 2-D DCT and curvelets, respectively.

There are some open questions. We hope to gain more insights (and hopefully answer the questions) via more computational experiments and theoretical analysis. Some of these open questions are:

- We want to explore the limit of our approach in finding a sparse atomic decomposition. For example, if the desired image can be made by a few atoms from a dictionary, will our approach find the sparsest atomic decomposition in the dictionary?
- For most of natural images (e.g., those images we have seen in our ordinary life), can they be represented by a few components from a dictionary made by 2-D DCT, 2-D wavelets, edgelets and curvelets? If not, what are other transforms we should bring in (or develop)?
- We think our sparse image representation approach can be used as a tool to preprocess a class of images. Based on the sparse representation results, we can decide which image transform should be chosen for the specific class of images. We are going to explore this idea. This gives a way of doing method selection and it has applications in image search, biological image analysis, etc.

## 7.2 Modifying Edgelet Dictionary

By inspection of Figure 6.2, one sees that the existing edgelet dictionary can be further improved. Indeed, in those figures, one sees that the wavelet component of the reconstruction is carrying a significant amount of the burden of edge representation. It seems that the terms in our edgelet dictionary do not have a width matched to the edge width, and that in consequence, many fine-scale wavelets are needed in the representation. In future experiments, we would try using edgelet features that have a finer width at fine scales and coarser width at coarse scales. This intuitive discussion matches some of the concerns in the paper [54].

The idea in [54] is to construct a new tight frame intended for efficient representation of 2-D objects with singularities along curves. The frame elements exhibit a range of dyadic positions, scales and angular orientations. The useful frame elements are highly directionally selective at fine scales. Moreover, the width of the frame elements scales with length according to the square of length.

The frame construction combines ideas from ridgelet and wavelet analysis. One tool is the monoscale ridgelet transform. The other tool is the use of multiresolution filter banks.

The frame coefficients are obtained by first separating the object into special multiresolution subbands  $f_s$ ,  $s \geq 0$  by applying filtering operation  $\Delta_s f = \Psi_{2s} * f$  for  $s \geq 0$ , where  $\Psi_{2s}$  is built from frequencies in an annulus extending from radius  $2^{2s}$  to  $2^{2s+2}$ . To the  $s$ -th subband one applies the monoscale ridgelet transform at scale  $s$ . Note that the monoscale ridgelet transform at scale index  $s$  is composed with multiresolution filtering near index  $2s$ . The  $(s, 2s)$  pairing makes the useful frame elements highly anisotropic at fine scales.

The frame gives rise to a near-optimal atomic decomposition of objects which have discontinuities along a closed  $C^2$  curve. Simple thresholding of frame coefficients gives rise to new methods of approximation and smoothing that are highly anisotropic and provably optimal.

We refer readers to the original paper cited at the beginning of this section for more details.

## 7.3 Accelerating the Iterative Algorithm

The idea of block coordinate relaxation can be found in [124, 125]. A proof of the convergence can be found in [135]. We happen to know an independent work in [130]. The idea

of block coordinate relaxation is the following: consider minimizing an objective function that is a sum of a residual sum of square and a  $\ell^1$  penalty term on coefficients  $x$ ,

$$(B1) \quad \underset{x}{\text{minimize}} \quad \|y - \Phi x\|_2^2 + \lambda \|x\|_1.$$

Suppose the matrix  $\Phi$  can be split into several orthogonal and square submatrices,  $\Phi = [\Phi_1, \Phi_2, \dots, \Phi_k]$ ; and  $x$  can be split into some subvectors accordingly,  $x = (x_1^T, x_2^T, \dots, x_k^T)^T$ . Problem (B1) is equivalent to

$$(B2) \quad \underset{x_1, x_2, \dots, x_k}{\text{minimize}} \quad \|y - \sum_{i=1}^k \Phi_i x_i\|_2^2 + \lambda \sum_{i=1}^k \|x_i\|_1.$$

When  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k$  are fixed but not  $x_i$ , the minimizer  $\tilde{x}_i$  of (B2) for  $x_i$  is

$$\tilde{x}_i = \eta_{\lambda/2} \left( \Phi_i^T \left( y - \sum_{\substack{l=1 \\ l \neq i}}^k \Phi_l x_l \right) \right),$$

where  $\eta_{\lambda/2}$  is a soft-thresholding operator (for  $x \in \mathbb{R}$ ):

$$\eta_{\lambda/2}(x) = \begin{cases} \text{sign}(x)(|x| - \lambda/2), & \text{if } |x| \geq \lambda/2, \\ 0, & \text{otherwise.} \end{cases}$$

We may iteratively solve problem (B2) and at each iteration, we rotate the soft-thresholding scheme through all subsystems. (Each subsystem is associated with a pair  $(\Phi_i, x_i)$ .) This method is called *block coordinate relaxation* (BCR) in [124, 125]. Bruce, Sardy and Tseng [124] report that in their experiments, BCR is faster than the interior point method proposed by Chen, Donoho and Saunders [27]. They also note that if some subsystems are not orthogonal, then BCR does not apply.

Motivated by BCR, we may split our original optimization problem into several subproblems. (Recall that our matrix  $\Phi$  is also a combination of submatrices associated with some image transforms.) We can develop another iterative method. In each iteration, we solve each subproblem one by one by assuming that the coefficients associated with other subsystems are fixed. If a subproblem corresponds to an orthogonal matrix, then the solution is simply a result of soft-thresholding of the analysis transform of the residual image. If a subproblem does not correspond to an orthogonal matrix, moreover, if it corresponds to a

overcomplete system, then we use our approach (that uses Newton method and LSQR) to solve it. Compared with the original problem, each subproblem has a smaller size, so hopefully this splitting approach will give us faster convergence than our previously presented global approach (that minimizes the objective function as whole).

Some experiments with BCR and its comparison with our approach is an interesting topic for future research.



# Appendix A

## Direct Edgelet Transform

This chapter is about the implementation of edgelet transform described in [50]. We give a review of edgelets in Section A.1, then some examples in Section A.2, and finally some details in Section A.3.

### A.1 Introduction

The edgelet [50] transform was developed to represent needle-like features in images. Edgelets are 2-D objects taking various scales, locations and orientations. If we consider an image as a function on a unit square  $[0, 1] \times [0, 1]$ , an edgelet system is constructed as follows:

- [E1] Partition the unit square into dyadic sub-squares, so that the sides of these squares take values at  $1/2, 1/4, 1/8, \dots$
- [E2] On each dyadic subsquare, put equally-spaced vertices on the boundary, starting from corners. We require each side equally partitioned by these vertices, and we generally assume that there are dyadic and integral number of vertices on each side, so the distance between two neighbor vertices should be a dyadic value too.
- [E3] For a line segment that connects two vertices as in [E2], if it does not coincide with a boundary, then it is called an *edgelet*.
- [E4] The edgelet system is a collection of all the edgelets as in [E3].

An  $N \times N$  digital image can be considered as a sample of a continuous 2-D function in a unit square. The value of each pixel is equal to the sample value at points  $(i/N, j/N)$ ,

$i, j = 1, 2, \dots, N$ . It's natural to assume that a vertex mentioned in [E2] must be located at a pixel. The cardinality of an edgelet system has  $O(N^2 \log_2 N)$ . More details are given in Section A.3.2.

An *edgel* is a line segment connecting a pair of pixels in an image. Note if we take all the possible edgels in an  $N \times N$  image, we have  $O(N^4)$  of them. Moreover, for any edgel, it's proven in [50] that it takes at most  $O(\log_2 N)$  edgelets to approximate it within a distance  $1/N + \delta$ , where  $\delta$  is a constant.

The coefficients of the edgelet transform are simply the integration of the 2-D function along these edgelets.

There is a fast algorithm to compute an approximate edgelet transform [50]. For an  $N \times N$  image, the complexity of the fast algorithm is  $O(N^2 \log_2 N)$ . The fast edgelet transform will be the topic of the next chapter.

This transform has been implemented in C and it is callable via a Matlab MEX function. It can serve as a benchmark for testing other transforms, which are designed to capture linear features in images.

## A.2 Examples

Before we present details, let's first look at some examples. The key idea of developing this transform is hoping that if the original image is made by a few needle-like components, then this transform will give a small number of significant coefficients, and the rest of the coefficients will be relatively small. Moreover, if we apply the adjoint transform to the coefficients selected by keeping only these with significant amplitudes, then the reconstructed image should be close to the original.

To test the above idea, we select four images:

[**Huo**] a Chinese character,  $64 \times 64$ ;

[**Sticky**] a sticky figure,  $128 \times 128$ ;

[**WoodGrain**] an image of wood grain,  $512 \times 512$ ;

[**Lenna**] the lenna image,  $512 \times 512$ .

[**Huo**] was selected because it is made by a few lines, so it is an ideal testing image. [**Sticky**] has a patch in the head. We apply an edge filter to this image before we apply the

edgelet transform. [WoodGrain] is a natural image, but it has significant linear features in it. [Lenna] is a standard testing image in image processing. As for [Sticky], we apply an edge filter to [Lenna] before doing the edgelet transform. The images in the above list are roughly ranked by the abundance (of course this could be subjective) of linear features.

Figure A.1, A.2, A.3 and A.4 show the numerical results. From the reconstructions based on partial coefficients—Figure A.1 (b), (c), (e) and (f), Figure A.2 (d), (e) and (f), Figure A.3 (b), (c), (e) and (f), Figure A.4 (d), (e) and (f)—we see that the significant edgelet coefficients capture the linear features of the images. The following table shows the percentages of the coefficients being used in the reconstructions. Note all the percentages

	Recon. 1	Recon. 2	Recon. 3	Recon. 4
[Huo]	0.75 %	1.50 %	2.99 %	5.98 %
[Sticky]	0.03 %	0.08 %	0.14 %	NA
[WoodGrain]	0.54 %	1.09 %	2.17 %	4.35 %
[Lenna]	0.23 %	0.47 %	0.93 %	NA

Table A.1: Percentages of edgelet coefficients being used in the reconstructions.

in the above table are small, say, less than 6%. The larger the percentage is, the better the reconstruction captures the linear features in the images.

### A.2.1 Edge Filter

Here we design an edge filter. Let  $A$  represent the original image. Define 2-D filters  $D_1$ ,  $D_2$  and  $D_3$  as

$$D_1 = \begin{pmatrix} - & + \\ - & + \end{pmatrix}, \quad D_2 = \begin{pmatrix} - & - \\ + & + \end{pmatrix}, \quad D_3 = \begin{pmatrix} - & + \\ + & - \end{pmatrix}.$$

Let “ $\star$ ” denote 2-D convolution. The notation  $[\cdot]^2$  means square each element of the matrix in the brackets. Let “+” be an elementwise addition operator. The edge filtered image of  $A$  is defined as

$$A^E = [A \star D_1]^2 + [A \star D_2]^2 + [A \star D_3]^2.$$

As we have explained, for the [Sticky] and [Lenna] image, we first apply an edge filter.

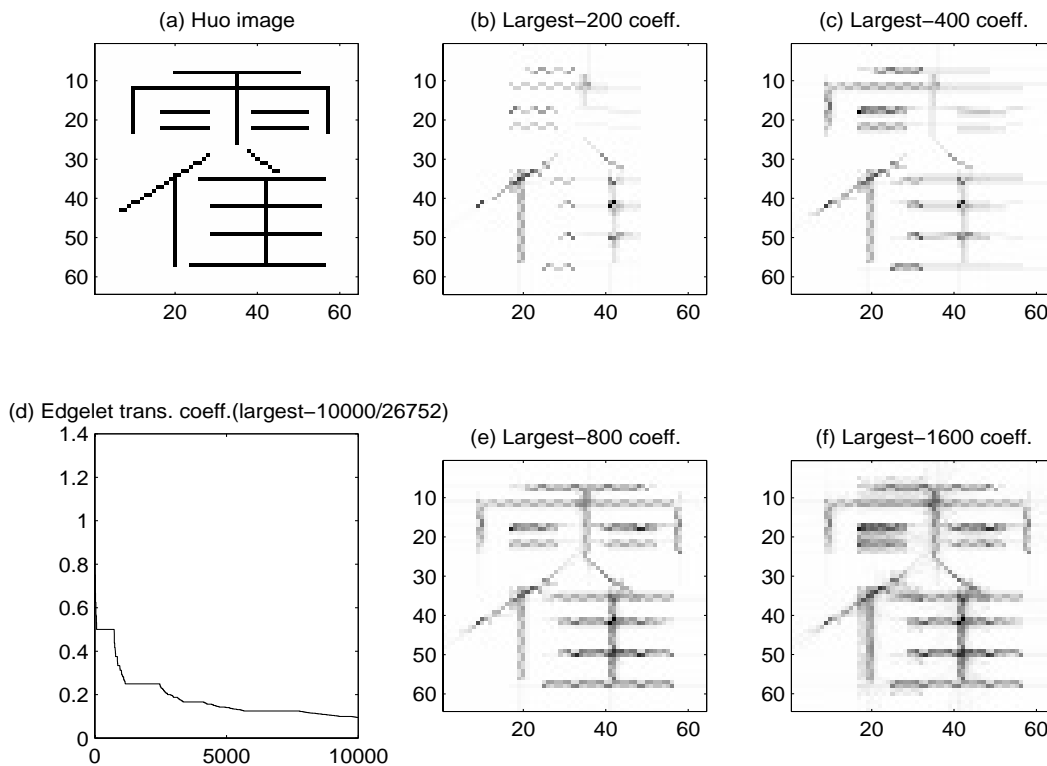


Figure A.1: Edgelet transform of the Chinese character “Huo”: (a) is the original; (d) is the sorted coefficients; (b), (c), (e) and (f) are reconstructions based on the largest 200, 400, 800, 1600 coefficients, respectively.

The reason to do this is that both of them show some patchy patterns in the original images. The filtered images show more obvious edge features. See Figure A.2 (b) and Figure A.4 (b).

### A.3 Details

A *direct* way of computing edgelet coefficients is explained. It is direct in the sense that every single edgelet coefficient is computed by a direct method; the coefficient is a weighted sum of the pixel values (intensities) that the edgelet trespasses. For an  $N \times N$  image, the order of the complexity of the direct method is  $O(N^3 \log N)$ .

Section A.3.1 gives the definition of the *direct* edgelet transform. Section A.3.2 calculates the cardinality of the edgelet system. Section A.3.3 specifies the ordering. Section A.3.4

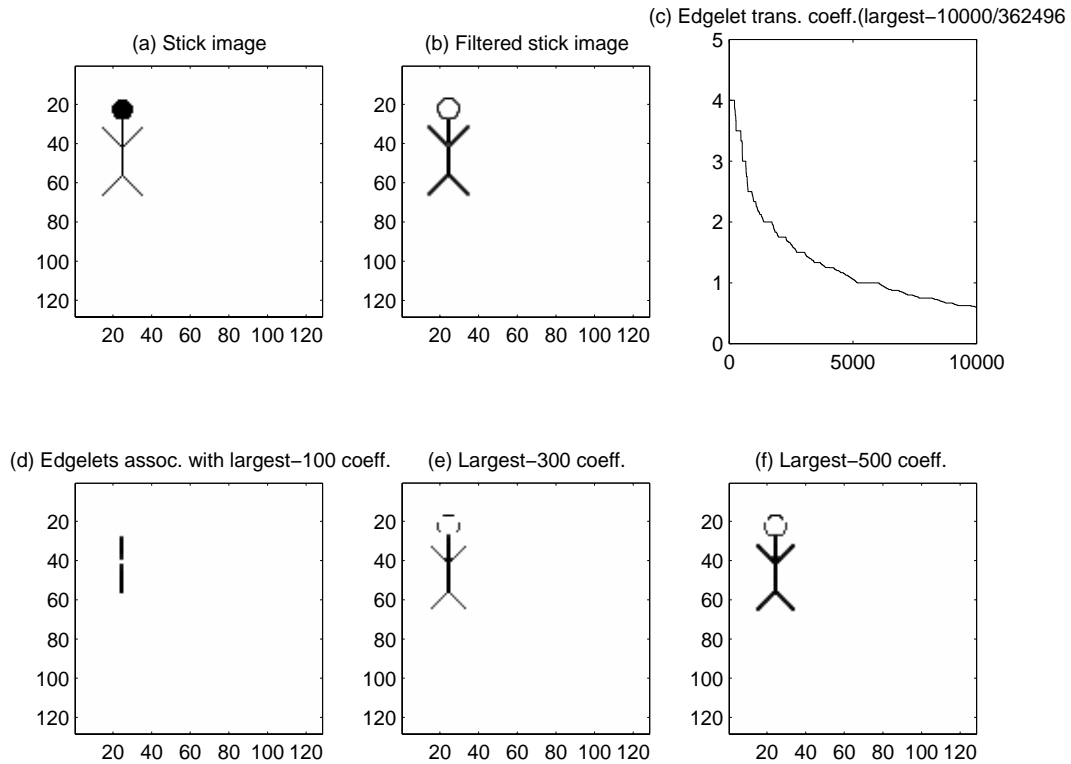


Figure A.2: Edgelet transform of the sticky image: (a) is the original; (b) is the filtered image; (c) is the sorted coefficients; (d), (e) and (f) are the reconstructions based on the largest 100, 300 and 500 coefficients, respectively.

explains how to compute a single edgelet coefficient. Section A.3.5 shows how to do the adjoint transform.

### A.3.1 Definition

In this subsection, we define what a *direct* edgelet transform is.

For continuous functions, the edgelet transform is defined in [50]. But in modern computing practice, an image is digitalized: it is a discrete function, or a matrix. There are many possibilities to realize the edgelet transform for a matrix, due to many ways to interpolate a matrix as a continuous 2-D function.

We explain a way of interpolation. Suppose the image is a squared image having  $N$  rows and  $N$  columns. The total number of pixels is  $N^2$ . We assume  $N$  is a power of 2,  $N = 2^n$ . Let the matrix  $I = I(i, j)_{1 \leq i, j \leq N} \in \mathbf{R}^{N \times N}$  correspond to the digital image,

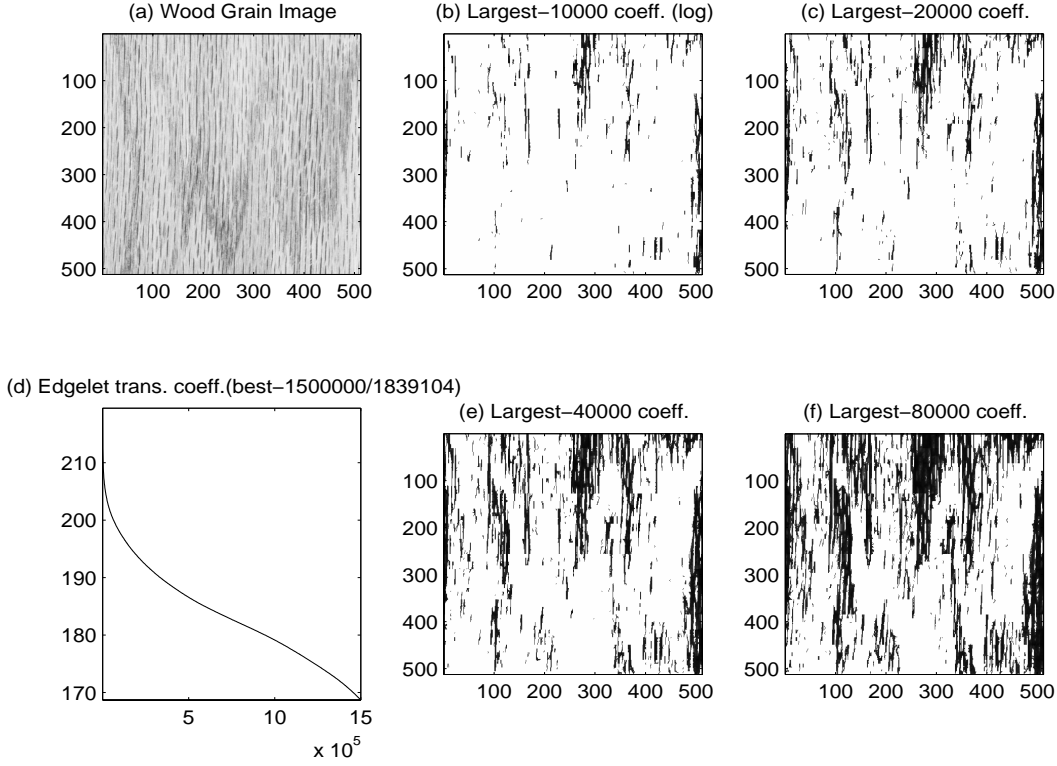


Figure A.3: Edgelet transform of the wood grain image: (a) is the original; (d) is the sorted coefficients; (b), (c), (e) and (f) are reconstructions based on the largest  $1 \times 10^4$ ,  $2 \times 10^4$ ,  $4 \times 10^4$ ,  $8 \times 10^4$  coefficients, respectively.

so when the image is a gray-scale image,  $I(i, j)$  is the gray scale at point  $(i, j)$ . A 2-D function  $f$  is defined in the following way: for  $i, j = 1, 2, \dots, N$ , if  $x$  and  $y$  fall in the cell  $((i-1)/N, i/N] \times ((j-1)/N, j/N]$ , then  $f(x, y) = I(i, j)$ . Suppose  $e$  is an edgelet as described in Section A.1. The edgelet transform of  $I$  corresponding to  $e$  is the integration of the function  $f$  divided by the length of  $e$ :

$$E(I, e) = \frac{\int_e f}{\text{length}(e)}.$$

Note this integration is a weighted sum of  $I(i, j)$  where the edgelet  $e$  intersects the square  $((i-1)/N, i/N] \times ((j-1)/N, j/N]$ . The weight of  $I(i, j)$  depends on the fraction of  $e$  in the cell  $((i-1)/N, i/N] \times ((j-1)/N, j/N]$ .

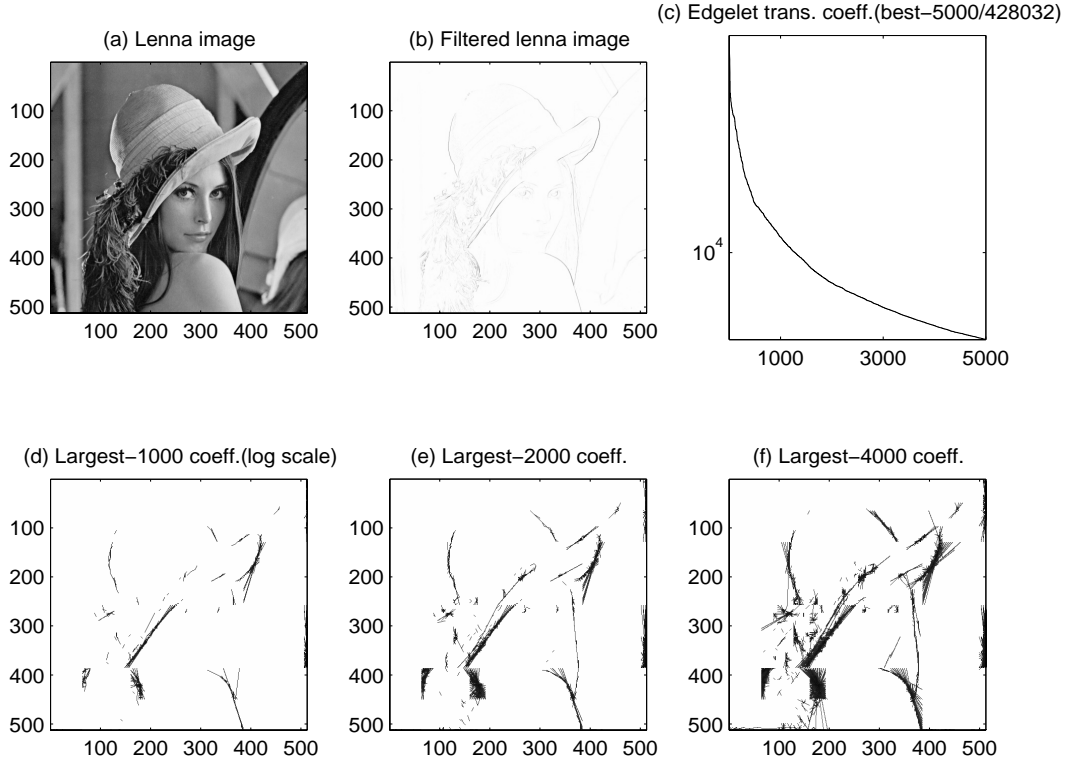


Figure A.4: Edgelet transform of Lenna image: (a) is the original Lenna image; (b) is the filtered version; (c) is the sorted largest 5,000 coefficients out of 428032. (d), (e) and (f) are the reconstructions based on the largest 1000, 2000 and 4000 coefficients, respectively.

### A.3.2 Cardinality

In this subsection, we discuss the size of edgelet coefficients.

For fixed scale  $j$  ( $l \leq j \leq n$ ), the number of dyadic squares is

$$N/2^j \times N/2^j = 2^{n-j} \times 2^{n-j} = 2^{2n-2j}.$$

In each  $2^j \times 2^j$  dyadic square, every side has  $1 + 2^{j-l}$  vertices; hence the total number of edgelets is

$$\frac{1}{2} \left\{ 4 \cdot (2 \cdot 2^{j-l} - 1) + 4 \cdot (2^{j-l} - 1)(3 \cdot 2^{j-l} - 1) \right\} = 6 \cdot 2^{j-l} \cdot 2^{j-l} - 4 \cdot 2^{j-l}.$$

So the number of edgelets at scale  $j$  is

$$\#\{\text{edgelets at scale } j\} = 6 \cdot 2^{2n-2l} - 4 \cdot 2^{2n-j-l}.$$

If we only consider the scale between  $s_l$  and  $s_u$ ,  $l \leq s_l \leq s_u \leq n$ , the number of edgelets is

$$\begin{aligned} \#\{\text{edgelets}\} &= 6(s_u - s_l + 1) \cdot 2^{2n-2l} - 4 \sum_{j=s_l}^{s_u} 2^{2n-j-l} \\ &= 6(s_u - s_l + 1) \cdot 2^{2n-2l} - 4 \cdot 2^{2n-l} (2^{-(s_l-1)} - 2^{-s_u}). \end{aligned}$$

If  $s_l = l + 1$ ,  $s_u = n$ , the total number of edgelets is

$$\begin{aligned} \#\{\text{edgelets}\} &= 6(n - l) \cdot 2^{2n-2l} - 4 \sum_{j=l+1}^n 2^{2n-j-l} \\ &= 6(n - l) \cdot 2^{2n-2l} - 4 \cdot 2^{2n-l} (2^{-l} - 2^{-n}) \\ &= 6(n - l) \cdot 2^{2n-2l} - 2^{2n-2l+2} + 2^{n-l+2} \\ &= 6(n - l) \left(N/2^l\right)^2 - 4 \left(N/2^l\right)^2 + 4 \left(N/2^l\right). \end{aligned}$$

### A.3.3 Ordering

In this subsection, we discuss how to order the edgelet coefficients. The ordering has three layers:

1. order the scales;
2. within a scale, order dyadic squares;
3. within a dyadic square, order edgelets.

There is a natural way to order scales. The scale could go from the lowest  $s_l$  to the highest  $s_u$ ,  $s_l \leq s_u$ .

In the next two subsections, we describe how we order dyadic squares within a scale and how we order edgelets within a dyadic square.

### Ordering of Dyadic Squares

For scale  $j$ , we have  $2^{n-j} \times 2^{n-j}$  number of dyadic squares. The following table gives a natural way to order them.

1	2	3	...	$2^{n-j}$
$2^{n-j} + 1$	$2^{n-j} + 2$	$2^{n-j} + 3$	...	$2 \cdot 2^{n-j}$
$2 \cdot 2^{n-j} + 1$	$2 \cdot 2^{n-j} + 2$	$2 \cdot 2^{n-j} + 3$	...	$3 \cdot 2^{n-j}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$(2^{n-j} - 1) 2^{n-j} + 1$	$(2^{n-j} - 1) 2^{n-j} + 2$	$(2^{n-j} - 1) 2^{n-j} + 3$	...	$(2^{n-j})^2$

### Ordering of Edgelets Within a Dyadic Square

We start with ordering the vertices on the boundary. We will see that it gives an ordering for edgelets within this square.

For every dyadic square, the vertices on the boundary can be ordered by: starting from the left upper corner, labeling the first vertex to the right by 1, clockwise labeling the rest of vertices by integers from 2 to  $4 \cdot 2^{j-l}$ , where  $j$  is the scale of the dyadic square. Obviously  $4 \cdot 2^{j-l}$  is the total number of vertices on this dyadic square. When  $n = 3$  and  $l = 1$ , Figure A.5 illustrates the ordering of vertices for a dyadic square at scale  $j = 3$ .

The ordering of vertices gives a natural ordering of edgelets. In a  $2^j \times 2^j$  square, there are  $4 \cdot 2^{j-l}$  vertices. Since each edgelet is determined by two vertices, let a pair  $(k, l)$  denote the edgelet connecting vertex  $k$  and vertex  $l$ . We order the edgelets by the following two rules:

1. Index  $k$  is always less than index  $l$ :  $k < l$ .
2. Fixing first index  $k$ , let the second index  $l$  increase. When  $l$  hits the upper limit, increase index  $k$  by one, and reset index  $l$  to the lowest possible value restricted by the previous rule. Repeat this until none of the indices can be increased.

Let  $K = 2^{j-l}$ . The Table A.2 shows our ordering.

### Ordering for All the Edgelets

Combining the ordering in all three steps, we get an ordering for the edgelets. The following system enumerates all edgelets.

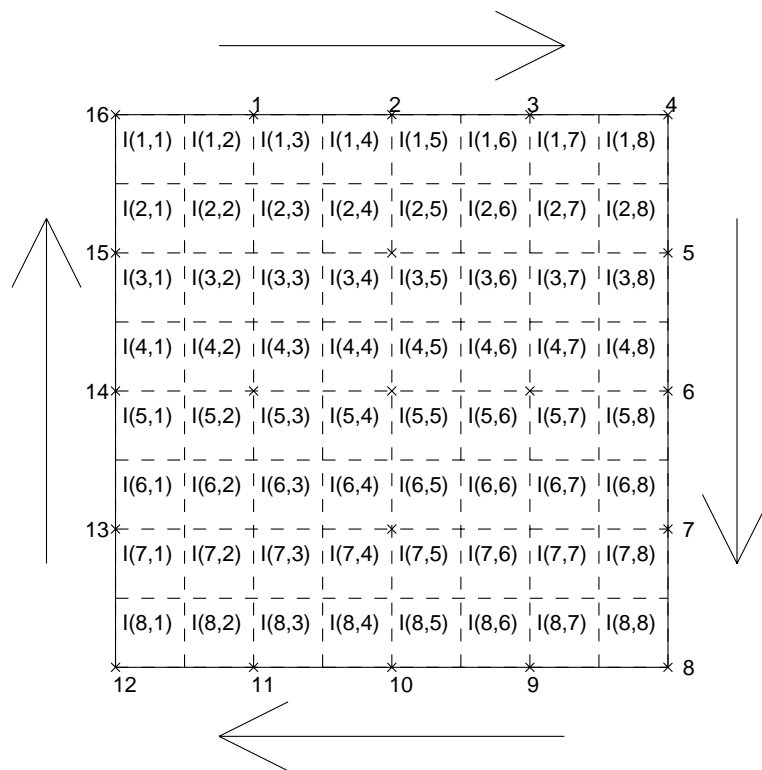


Figure A.5: Vertices at scale  $j$ , for a  $8 \times 8$  image with  $l = 1$ . The arrows shows the trend of ordering. Integers outside are the labels of vertices.

**For** scale  $j = s_l$  **to**  $s_u$ ,

**For** dyadic squares index  $d = 1$  **to**  $2^{2n-2j}$ ,

**For** edgelet index  $e = 1$  **to**  $6 \cdot 2^{2j-2l} - 4 \cdot 2^{j-l}$  (recall  $l$  is the coarsest scale)

            edgelet associated with  $(j, d, e)$ ;

**End;**

**End;**

**End.**

### A.3.4 Computing a Single Edgelet Coefficient

This section describes a *direct* way of computing edgelet coefficients. Basically, we compute them one by one, taking no more than  $O(N)$  operations each. There are  $O(N^2 \log N)$  edgelet coefficients, so the overall complexity of a direct algorithm is no higher than  $O(N^3 \log N)$ .

$(1, K + 1),$	$(1, K + 2),$	$(1, K + 3),$	$\dots$	$(1, 4K - 1),$
$(2, K + 1),$	$(2, K + 2),$	$(2, K + 3),$	$\dots$	$(2, 4K - 1),$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$(K - 1, K + 1),$	$(K - 1, K + 2),$	$(K - 1, K + 3),$	$\dots$	$(K - 1, 4K - 1),$
$(K, 2K + 1),$	$(K, 2K + 2),$	$(K, 2K + 3),$	$\dots$	$(K, 4K - 1),$
$(K + 1, 2K + 1),$	$(K + 1, 2K + 2),$	$(K + 1, 2K + 3),$	$\dots$	$(K + 1, 4K),$
$(K + 2, 2K + 1),$	$(K + 2, 2K + 2),$	$(K + 2, 2K + 3),$	$\dots$	$(K + 2, 4K),$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$(2K - 1, 2K + 1),$	$(2K - 1, 2K + 2),$	$(2K - 1, 2K + 3),$	$\dots$	$(2K - 1, 4K),$
$(2K, 3K + 1),$	$(2K, 3K + 2),$	$(2K, 3K + 3),$	$\dots$	$(2K, 4K),$
$(2K + 1, 3K + 1),$	$(2K + 1, 3K + 2),$	$(2K + 1, 3K + 3),$	$\dots$	$(2K + 1, 4K),$
$(2K + 2, 3K + 1),$	$(2K + 2, 3K + 2),$	$(2K + 2, 3K + 3),$	$\dots$	$(2K + 2, 4K),$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$(3K - 1, 3K + 1),$	$(3K - 1, 3K + 2),$	$(3K - 1, 3K + 3),$	$\dots$	$(3K - 1, 4K).$

Table A.2: Order of edgelets within a square.

We have the following algorithm: for matrix  $I$ , and an edgelet with end points  $(x_1, y_1)$  and  $(x_2, y_2)$ , where  $x_1, y_1, x_2$  and  $y_2$  are integers, we have

**Compute  $\mathbf{T}(I, \{x_1, y_1, x_2, y_2\})$**

1. **Case one:** If the edge is horizontal ( $x_1 = x_2$ ), then

$$\mathbf{T}(I, \{x_1, y_1, x_2, y_2\}) = \frac{1}{2|y_2 - y_1|} \sum_{i=x_1}^{x_1+1} \sum_{j=y_1+1}^{y_2} I(i, j).$$

2. **Case two:** If the edge is vertical ( $y_1 = y_2$ ), then

$$\mathbf{T}(I, \{x_1, y_1, x_2, y_2\}) = \frac{1}{2|x_2 - x_1|} \sum_{i=x_1+1}^{x_2} \sum_{j=y_1}^{y_1+1} I(i, j).$$

3. **Case three:** The edge is neither horizontal nor vertical. We must have

$x_1 < x_2$ . Let

$$\Delta = \frac{y_2 - y_1}{x_2 - x_1}.$$

If  $\Delta > 0$  ( $y_2 > y_1$ ), then

$$\mathbf{T}(I, \{x_1, y_1, x_2, y_2\}) = \frac{1}{(y_2 - y_1)} \sum_{i=x_1+1}^{x_2} \sum_{j=\lceil y_1+(i-x_1-1)\Delta \rceil}^{\lceil y_1+(i-x_1)\Delta \rceil} \omega[j, y_1 + (i - x_1 - 1)\Delta, y_1 + (i - x_1)\Delta] I(i, j);$$

else  $\Delta < 0$  ( $y_2 < y_1$ ),

$$\mathbf{T}(I, \{x_1, y_1, x_2, y_2\}) = \frac{1}{|y_2 - y_1|} \sum_{i=x_1+1}^{x_2} \sum_{j=\lceil y_1+(i-x_1)\Delta \rceil}^{\lceil y_1+(i-x_1-1)\Delta \rceil} \omega[j, y_1 + (i - x_1)\Delta, y_1 + (i - x_1 - 1)\Delta] I(i, j);$$

where  $\omega[j, x, y]$  is a function:

- when  $\lceil x \rceil = \lceil y \rceil$ , where  $\lceil x \rceil$  and  $\lceil y \rceil$  are the smallest integers bigger than or equal to  $x$  and  $y$ ,

$$\omega[j, x, y] = \begin{cases} (y - x) & j = \lceil x \rceil, \\ 0 & \text{otherwise;} \end{cases}$$

- when  $\lceil x \rceil < \lceil y \rceil$ ,

$$\omega[j, x, y] = \begin{cases} (j - x) & j = \lceil x \rceil, \\ 1 & \lceil x \rceil < j < \lceil y \rceil, \text{ (may be empty),} \\ y - \lceil y \rceil + 1 & j = \lceil y \rceil, \\ 0 & \text{otherwise.} \end{cases}$$

Figure A.6 gives an illustration of weights in edgelet transform. For a single edgelet from  $(x_1, y_1)$  to  $(x_2, y_2)$ , the weight of pixel  $(k, l)$  is equal to the length of the thick line at the bottom of the square associated with this pixel, divided by the quantity  $|y_2 - y_1|$ .

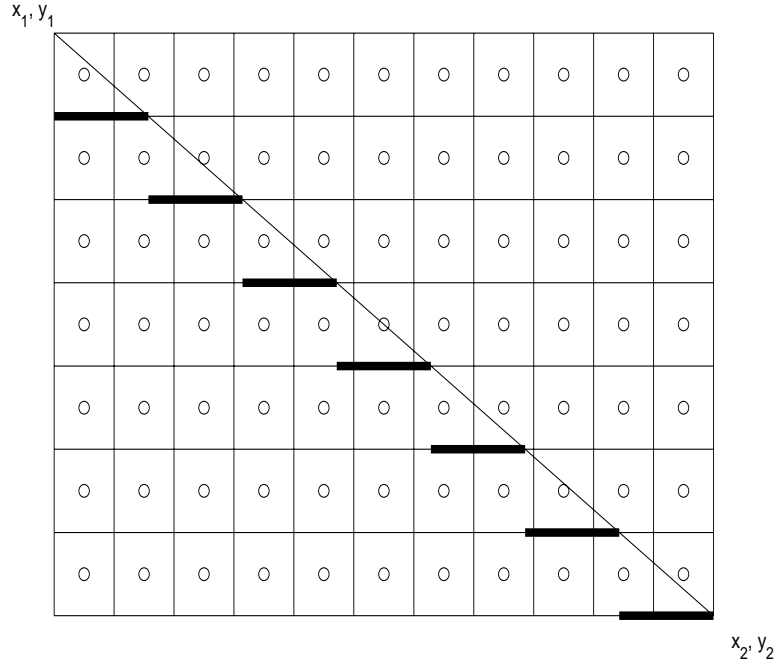


Figure A.6: Weights of pixels for one edgelet coefficient.

### A.3.5 Adjoint of the Direct Edgelet Transform

In this section, we derive the adjoint edgelet transform. We start with reviewing the edgelet transform in a symbolic way. If  $\alpha$  is an index of the image— $\alpha = (i, j)$  and for an image  $I$ ,  $I(\alpha)$  is the intensity at pixel  $\alpha$ —and  $e$  is an index of the edgelet, the coefficient of the edgelet transform at  $e$  is actually a weighted sum:

$$\mathbf{T}(I, e) = \sum_{\alpha} \omega(\alpha, e) I(\alpha).$$

Let  $\mathbf{T}^*$  denote the adjoint edgelet transform. By definition, for any  $x \in \mathbf{R}^{N \times N}$  in the image domain and any  $y$  in the edgelet transform domain, we must have

$$\langle \mathbf{T}x, y \rangle = \langle x, \mathbf{T}^*y \rangle. \tag{A.1}$$

Note both sides of equation (A.1) are linear functions of  $y$ . Let  $y = \delta_e$ , which means that

$y$  is equal to one if and only if it is at the position corresponding to  $e$ , and zero elsewhere. Note  $\delta_e$  is a generalized version of the Dirac function. Equation (A.1) becomes

$$\forall e, \quad \langle \mathbf{T}x, \delta_e \rangle = \langle x, \mathbf{T}^* \delta_e \rangle. \quad (\text{A.2})$$

The left-hand side of the above equation can be written as

$$\langle \mathbf{T}x, \delta_e \rangle = \mathbf{T}(x, e) = \sum_{\alpha} \omega(\alpha, e)x(\alpha).$$

Substituting the above into (A.2), we have

$$\mathbf{T}^* \delta_e(\alpha) = \omega(\alpha, e). \quad (\text{A.3})$$

This gives the formula for the adjoint edgelet transform.

Actually, in linear algebra, this is obvious: if the forward transform corresponds to the transform matrix, then the adjoint transform corresponds to the transpose of the transform matrix.

### A.3.6 Discussion

The following observations make a fast algorithm for the edgelet transform possible:

- We can utilize inter-scale relationships. Some edgelets at coarse scales are just a linear combination of other edgelets at a finer scale. So its coefficient is a linear combination of other edgelet coefficients. An analogue of it is the 2-scale relationship in orthogonal wavelets.
- We may use the special pattern of the edgelet transform. This idea is similar to the idea in [18].
- The edgelet transform is actually the Radon transform restricted in a subsquare. It is possible to do approximate fast Radon transform via fast Fourier transforms.

We avoid further discussion on this topic.

## Appendix B

# Fast Edgelet-like Transform

In this chapter, we present a fast algorithm to approximate the edgelet transform in discrete cases. Note the result after the current transform is *not* exactly the result after a direct edgelet transform as we presented in the previous chapter. They are close, in the sense that we can still consider the coefficients after this transform are approximate integrations along some line segments, but the line segments are *not* exactly the edgelets we described in the previous chapter.

It is clear that in order to have a fast algorithm, it is necessary to modify the original definition of edgelets. In many cases, there is a trade off between the simplicity or efficiency of the algorithm and the loyalty to the original definition. The same is true here. In this chapter, we show that we can change the system of the edgelets a little, so that a fast ( $O(N^2 \log N)$ ) algorithm is feasible, and the transform still captures the linear features in an image.

The current algorithm is based on three key foundations:

1. Fourier slice theorem,
2. fast Fourier transform (FFT),
3. fast X-interpolation based on fractional Fourier transform.

In the continuous case, the idea presented here has been extensively developed in [48, 50, 52, 51, 49]. This approach is related to unpublished work on fast approximate Radon transforms by Averbuch and Coifman, and to published work in the field of Synthetic Aperture Radar (SAR) tomography and medical tomography. These connections to

published work came to light only in the final stages of editing this thesis. The first Matlab version of the algorithm being presented was coded by David Donoho. The author made several modifications to the algorithm, and also some analysis is presented at the end of this chapter.

The rest of the chapter is organized as following: In Section B.1, we introduce the Fourier slice Theorem and the continuous Radon transform. Note both are for continuous functions. Section B.2 describes in detail the main algorithm—an algorithm for fast edgelet-like transform. The tools necessary to derive the adjoint of this transform are presented in Section B.3. Some discussion about miscellaneous properties of the fast edgelet-like transform are in Section B.4, including storage, computational complexity, effective region, and ill-conditioning. Finally, we present some examples in Section B.5.

## B.1 Transforms for 2-D Continuous Functions

### B.1.1 Fourier Slice Theorem

The Fourier slice theorem is the key for us to utilize the fast Fourier transform to implement the fast Radon transform. The basic idea is that for a 2-D continuous function, if we do a 2-D Fourier transform of it, then sampling along a straight line traversing the origin, the result is the same as the result of projecting the original function onto the same straight line then taking 1-D Fourier transform.

We will just sketch the idea of a proof. Suppose  $f(x, y)$  is a continuous function in 2-D, where  $(x, y) \in \mathbb{R}^2$ . We use  $\mathbf{f}$  to denote the continuous interpolation of the image  $I$ , and  $f$  to denote a general 2-D function. Let  $\hat{f}(\xi, \eta)$  denote its 2-D Fourier transform. Then we have

$$\hat{f}(\xi, \eta) = \int_y \int_x f(x, y) e^{-2\pi\sqrt{-1}x\xi} e^{-2\pi\sqrt{-1}y\eta} dx dy. \quad (\text{B.1})$$

Taking polar coordinates in both the original domain and the Fourier domain, we have

$$\xi = \rho \cos \theta, \quad \eta = \rho \sin \theta, \quad x = \rho' \cos \theta', \quad y = \rho' \sin \theta'.$$

Let  $\hat{f}_\theta^{(s)}(\rho)$  stand for the sampling of the function  $\hat{f}$  along the line  $\{(\rho \cos \theta, \rho \sin \theta) : \rho \in$

$\mathbb{R}^+, \theta$  is fixed}. Then we have

$$\hat{f}_\theta^{(s)}(\rho) = \hat{f}(\rho \cos \theta, \rho \sin \theta).$$

Let  $f_\theta^{(p)}(\rho')$  be the projection of  $f$  onto the line having angle  $\theta'$  in the original domain:

$$f_\theta^{(p)}(\rho') d\rho' = \int_{x \cos \theta' + y \sin \theta' = \rho'} f(x, y) dx dy. \quad (\text{B.2})$$

Later we see that this is actually the continuous Radon transform. Substituting them into equation (B.1), we have

$$\begin{aligned} \hat{f}_\theta^{(s)}(\rho) &= \hat{f}(\rho \cos \theta, \rho \sin \theta) \\ &= \int \int_{x, y} f(x, y) e^{-2\pi i(x\rho \cos \theta + y\rho \sin \theta)} dx dy \\ &= \int_{\rho'} \int_{x \cos \theta + y \sin \theta = \rho'} f(x, y) dx dy e^{-2\pi i\rho\rho'} \\ &= \int_{\rho'} f_\theta^{(p)}(\rho') e^{-2\pi i\rho\rho'} d\rho'. \end{aligned}$$

Note the last term is exactly the 1-D Fourier transform of the projection function  $f_\theta^{(p)}(\rho')$ . This gives the Fourier slice theorem.

### B.1.2 Continuous Radon Transform

The Radon transform is essentially a projection. Suppose  $f$  is a 2-D continuous function. Let  $(x, y)$  be the Cartesian coordinates, and let  $(\rho, \theta)$  be the Polar coordinates, where  $\rho$  is the radial parameter and  $\theta$  is the angular parameter. The Radon transform of function  $f$  is defined as

$$\mathbb{R}(f, \rho, \theta) = \int_{\rho = x \cos \theta + y \sin \theta} f(x, y).$$

Recalling function  $f_\theta^{(p)}(\cdot)$  in (B.2) is defined as a projection function, we have

$$\mathbb{R}(f, \rho, \theta) = f_\theta^{(p)}(\rho).$$

From the Fourier slice theorem and the above equality, we have

$$\hat{f}_\theta^{(s)}(\rho) = \int_{\rho'} \mathbb{R}(f, \rho', \theta) e^{-2\pi i \rho \rho'} d\rho'.$$

Taking the inverse Fourier transform, we have

$$\mathbb{R}(f, \rho, \theta) = \int_{\rho'} \hat{f}_\theta^{(s)}(\rho') e^{2\pi i \rho \rho'} d\rho'.$$

Thus in dimension 2, the Radon transform of  $f$  is the inverse 1-D Fourier transform of a specially sampled (along a straight line going through the origin) 2-D Fourier transform of the original function  $f$ . Since there are fast algorithms for the Fourier transform, we can have fast algorithms for the Radon transform.

## B.2 Discrete Algorithm

To some extent, the edgelet transform can be viewed as the Radon transform restricted to a small square. There is a fast way to do the Radon transform. For an  $N$  by  $N$  image, we can find an  $O(N^2 \log N)$  algorithm by using the fast Fourier transform. The key idea in developing a fast approximate edgelet transform is to do a discrete version of the Radon transform. A discrete version of the Radon transform is the algorithm we presented in this section.

The Radon transform was originally defined for continuous functions. The Fourier slice theorem is based on continuous functions also. Our algorithm has to be based on discrete data, say, a matrix. A natural way to transfer a matrix to a continuous function is to view the matrix as a sampling from a continuous function. We can calculate the analytical Radon transform of that continuous function, then sample it to get the corresponding discrete result.

The first question arising is how to interpolate the data. The second question is, because the Radon transform uses polar coordinates, and a digital image is generally sampled on a grid in the Cartesian coordinate, how do we switch the coordinates and still preserve the fast algorithms.

Section B.2.1 gives an outline of the algorithm. Section B.2.2 describes some issues in interpolation. Section B.2.3 is about a fast algorithm to switch from Cartesian coordinates to polar coordinates. Section B.2.4 presents the algorithm.

### B.2.1 Synopsis

We think of the Radon transform for an image as the result of the following five steps:

#### Outline

image	
↓	(1) Interpolate the discrete data to a continuous 2-D function.
$f(x, y)$	
↓	(2) Do 2-D continuous time Fourier transform.
$\hat{f}(x, y)$	
↓	(3) Switch from Cartesian to polar coordinates.
$\hat{f}(\rho, \theta)$	
↓	(4) Sample at fractional frequencies according to a grid in the polar coordinate system.
$\hat{f}(\rho_i, \theta_j)$	
↓	(5) Do 1-D inverse discrete Fourier transform.
Radon transform	

In subsection B.2.2, we describe the sampling idea associated with steps (1), (2), (4) and (5). In subsection B.2.3, we describe a fast way to sample in Frequency domain.

### B.2.2 Interpolation: from Discrete to Continuous Image

We view image  $\{I(i, j) : i = 1, 2, \dots, N; j = 1, 2, \dots, N\}$  as a set of sampled values of a continuous function at grid points  $\{(i, j) : i = 1, 2, \dots, N; j = 1, 2, \dots, N\}$ , and the continuous function is defined as

$$\begin{aligned}
 f(x, y) &= \sum_{\substack{i=1,2,\dots,N; \\ j=1,2,\dots,N.}} I(i, j)\rho(x-i)\rho(y-j) \\
 &= \left( \sum_{\substack{i=1,2,\dots,N; \\ j=1,2,\dots,N.}} I(i, j)\delta(x-i)\delta(y-j) \right) \star (\rho(x)\rho(y)),
 \end{aligned}$$

where  $\rho$  is the interpolating kernel function. We assume  $\rho$  is equal to one at the origin and zero at all the other integers:

$$\rho(0) = 1, \text{ and } \rho(i) = 0, \text{ for } i = 1, 2, \dots, N,$$

notation  $\star$  stands for the convolution, and function  $\delta(\cdot)$  is the Dirac function at point 0.

The 2-D Fourier transform of  $f(x, y)$  is

$$\hat{f}(\xi, \eta) = \left( \sum_{\substack{i=1,2,\dots,N; \\ j=1,2,\dots,N}} I(i, j) e^{-\sqrt{-1} \cdot 2\pi \xi i} e^{-\sqrt{-1} \cdot 2\pi \eta j} \right) \hat{\rho}(\xi) \hat{\rho}(\eta), \quad (\text{B.3})$$

where  $\hat{\rho}(\cdot)$  is the Fourier transform of function  $\rho(\cdot)$ . Note there are two parts in  $\hat{f}(\xi, \eta)$ , the first part denoted by  $F(\xi, \eta)$ ,

$$F(\xi, \eta) = \sum_{\substack{i=1,2,\dots,N; \\ j=1,2,\dots,N}} I(i, j) e^{-\sqrt{-1} \cdot 2\pi \xi i} e^{-\sqrt{-1} \cdot 2\pi \eta j},$$

is actually the 2-D Fourier transform of  $I$ . Note  $F(\xi, \eta)$  is a periodic function with period one for both  $\xi$  and  $\eta$ :  $F(\xi + 1, \eta) = F(\xi, \eta)$  and  $F(\xi, \eta + 1) = F(\xi, \eta)$ . If we sample  $\xi$  and  $\eta$  at points  $\frac{1}{N}, \frac{2}{N}, \dots, 1$ , then we have the discrete Fourier transform (DFT). We know there is an  $O(N \log N)$  algorithm to implement.

Function  $\hat{\rho}(\cdot)$  typically has finite support. In this paper, we choose the support to have length equal to one, so that the support of  $\hat{\rho}(\xi) \hat{\rho}(\eta)$  forms a unit square. We did not choose a support wider than one for some reason we will mention later.

From equation (B.3),  $\hat{f}(\xi, \eta)$  is the periodic function  $F(\xi, \eta)$  truncated by  $\hat{\rho}(\xi) \hat{\rho}(\eta)$ . The shape of  $\hat{\rho}(\xi)$  and  $\hat{\rho}(\eta)$  determines the property of function  $\hat{f}(\xi, \eta)$ .

Section B.6.1 gives three examples of interpolation functions and some related discussion.

In this paper, we will choose the raised cosine function as our windowing function.

### B.2.3 X-interpolation: from Cartesian to Polar Coordinate

In this section, we describe how to transfer from Cartesian coordinates to polar coordinates. As in the synopsis, we do the coordinate switch in the Fourier domain.

From equation (B.3), function  $\hat{f}(\xi, \eta)$  is just a multiplication of function  $F(\xi, \eta)$  with a windowing function. If we know the function  $F(\xi, \eta)$ , the function  $\hat{f}(\xi, \eta)$  is almost a direct

extension. In the following, we treat  $\hat{f}(\xi, \eta)$  as a box function (e.g., the indicator function of the unit square).

Without loss of generality, we redefine  $F(\xi, \eta)$  as

$$F(\xi, \eta) = \sum_{\substack{i=0,1,\dots,N-1; \\ j=0,1,\dots,N-1.}} I(i+1, j+1) e^{-2\pi\sqrt{-1}\xi i} e^{-2\pi\sqrt{-1}\eta j}.$$

Note the range of the index of  $I$  is changed. This is to follow the convention in DFT. By applying the FFT, we can get the following matrix by an  $O(N^2 \log^2 N)$  algorithm:

$$\left( F\left(\frac{k}{N}, \frac{l}{N}\right) \right)_{\substack{k=0,1,\dots,N-1 \\ l=0,1,\dots,N-1}},$$

because for  $0 \leq k, l \leq N-1$ ,

$$F\left(\frac{k}{N}, \frac{l}{N}\right) = \sum_{\substack{i=0,1,\dots,N-1 \\ j=0,1,\dots,N-1}} I(i+1, j+1) e^{-2\pi\sqrt{-1}\frac{k}{N}i} e^{-2\pi\sqrt{-1}\frac{l}{N}j}.$$

This is the 2-D discrete Fourier transform.

For Radon transform, instead of sampling at Cartesian grid point  $(k/N, l/N)$ , we need to sample at polar grid points. We develop an X-interpolation approach, which is taking samples at grid points in polar coordinate. So after X-interpolation, we get the following  $N \times 2N$  matrix:

$$\left[ \begin{array}{l} \left( F\left(\frac{k}{N} - \frac{1}{2}, s(k) + l \cdot \Delta(k) - \frac{1}{2}\right) \right)_{\substack{k=0,1,\dots,N-1 \\ l=0,1,\dots,N-1}}, \dots \\ \left( F\left(s(k) + l \cdot \Delta(k) - \frac{1}{2}, \frac{N-k}{N} - \frac{1}{2}\right) \right)_{\substack{k=0,1,\dots,N-1 \\ l=0,1,\dots,N-1}} \end{array} \right], \quad (\text{B.4})$$

where

$$s(k) = \frac{k}{N}, \quad \Delta(k) = \frac{1}{N} \left(1 - \frac{2k}{N}\right). \quad (\text{B.5})$$

Figure B.1 illustrates the idea of X-interpolation.

There is a fast way to compute the matrix in (B.4), using an idea from [6].

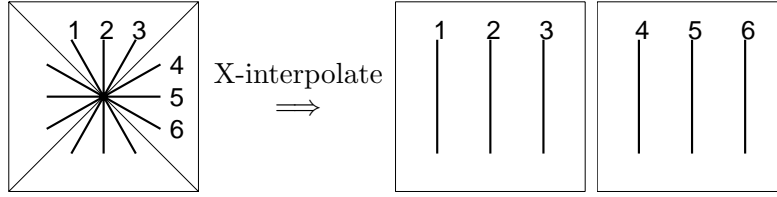


Figure B.1: X-interpolation.

For  $0 \leq k, l \leq N - 1$ , we have

$$\begin{aligned}
& F\left(\frac{k}{N} - \frac{1}{2}, s(k) + l \cdot \Delta(k) - \frac{1}{2}\right) \\
&= \sum_{\substack{i=0,1,\dots,N-1; \\ j=0,1,\dots,N-1.}} I(i+1, j+1) e^{-2\pi\sqrt{-1}(\frac{k}{N}-\frac{1}{2})i} e^{-2\pi\sqrt{-1}(s(k)+l\cdot\Delta(k)-\frac{1}{2})j} \\
&= \sum_{j=0,1,\dots,N-1} \left( \underbrace{\sum_{i=0,1,\dots,N-1} I(i+1, j+1) e^{-2\pi\sqrt{-1}(\frac{k}{N}-\frac{1}{2})i}}_{\text{define as } g(k, j)} \right) e^{-2\pi\sqrt{-1}(s(k)+l\cdot\Delta(k)-\frac{1}{2})j} \\
&= \sum_{j=0,1,\dots,N-1} g(k, j) e^{-2\pi\sqrt{-1}(s(k)+l\cdot\Delta(k)-\frac{1}{2})j} \\
&= e^{-2\pi\sqrt{-1}\Delta(k)\frac{l^2}{2}} \cdot \underbrace{\sum_{j=0,1,\dots,N-1} g(k, j) e^{-2\pi\sqrt{-1}(j\cdot s(k)-\frac{1}{2}j+\Delta(k)\frac{j^2}{2})}}_{\text{convolution}} \cdot e^{\pi\sqrt{-1}\Delta(k)(l-j)^2}.
\end{aligned}$$

The summation in the last expression is actually a convolution.

The matrix  $(g(k, j))_{\substack{k=0,1,\dots,N-1 \\ j=0,1,\dots,N-1}}$  is the 1-D Fourier transform of the image matrix  $I(i+1, j+1)_{\substack{i=0,1,\dots,N-1 \\ j=0,1,\dots,N-1}}$  by column. We can compute matrix  $(g(k, j))_{\substack{k=0,1,\dots,N-1 \\ j=0,1,\dots,N-1}}$  with  $O(N^2 \log N)$  work. For fixed  $k$ , computing function value  $F(\frac{k}{N}, s(k) + l \cdot \Delta(k))$  is basically a convolution. To compute the  $k$ th row in matrix  $(F(\frac{k}{N}, s(k) + l \cdot \Delta(k)))_{\substack{k=0,1,\dots,N-1 \\ l=0,1,\dots,N-1}}$ , we utilize the Toeplitz structure. We can compute the  $k$ th row in  $O(N \log N)$  time, and hence computing the first half of the matrix in (B.4) has  $O(N^2 \log N)$  complexity.

For the second half of the matrix in (B.4), we have the same result:

$$\begin{aligned}
& F\left(s(k) + l \cdot \Delta(k) - \frac{1}{2}, \frac{N-k}{N} - \frac{1}{2}\right) \\
&= \sum_{\substack{i=0,1,\dots,N-1; \\ j=0,1,\dots,N-1.}} I(i+1, j+1) e^{-2\pi\sqrt{-1}(s(k)+l\cdot\Delta(k)-\frac{1}{2})i} e^{-2\pi\sqrt{-1}(\frac{N-k}{N}-\frac{1}{2})j} \\
&= \sum_{i=0,1,\dots,N-1} \underbrace{\left( \sum_{j=0,1,\dots,N-1} I(i+1, j+1) e^{-2\pi\sqrt{-1}\frac{j}{N}} e^{-2\pi\sqrt{-1}(\frac{N-1-k}{N}-\frac{1}{2})j} \right)}_{\text{define as } h(i, N-1-k)} \\
&\quad \cdot e^{-2\pi\sqrt{-1}(s(k)+l\cdot\Delta(k)-\frac{1}{2})i} \\
&= \sum_{i=0,1,\dots,N-1} h(i, N-1-k) e^{-2\pi\sqrt{-1}\left(i\cdot s(k)-\frac{1}{2}i+\Delta(k)\left[\frac{l^2}{2}+\frac{i^2}{2}-\frac{1}{2}(l-i)^2\right]\right)} \\
&= e^{-2\pi\sqrt{-1}\Delta(k)\frac{l^2}{2}} \cdot \underbrace{\sum_{i=0,1,\dots,N-1} h(i, N-1-k) e^{-2\pi\sqrt{-1}\left(i\cdot s(k)-\frac{1}{2}i+\Delta(k)\frac{i^2}{2}\right)}}_{\text{convolution}} \cdot e^{\pi\sqrt{-1}\Delta(k)(l-i)^2}.
\end{aligned}$$

The matrix  $(h(i, k))_{\substack{i=0,1,\dots,N-1 \\ k=0,1,\dots,N-1}}$  is an assembly of the 1-D Fourier transform of rows of the image matrix  $(I(i+1, j+1))_{\substack{i=0,1,\dots,N-1 \\ j=0,1,\dots,N-1}}$ . For the same reason, the second half of the matrix in (B.4) can be computed with  $O(N^2 \log N)$  complexity.

The discrete Radon transform is the 1-D inverse discrete Fourier transform of columns of the matrix in (B.4). Obviously the complexity at this step is no higher than  $O(N^2 \log N)$ , so the overall complexity of the discrete Radon transform is  $O(N^2 \log N)$ .

In order to make each column of the matrix in (B.4) be the DFT of a real sequence, for any fixed  $l, 0 \leq l \leq N-1$ , and  $k = 1, 2, \dots, N/2$ , we need to have

$$F\left(\frac{k}{N} - \frac{1}{2}, s(k) + l \cdot \Delta(k) - \frac{1}{2}\right) = \overline{F\left(\frac{N-k}{N} - \frac{1}{2}, s(N-k) + l \cdot \Delta(N-k) - \frac{1}{2}\right)},$$

and

$$F\left(s(k) + l \cdot \Delta(k) - \frac{1}{2}, \frac{N-k}{N} - \frac{1}{2}\right) = \overline{F\left(s(N-k) + l \cdot \Delta(N-k) - \frac{1}{2}, \frac{k}{N} - \frac{1}{2}\right)}.$$

It is easy to verify that for  $s(\cdot)$  and  $\Delta(\cdot)$  in (B.5), the above equations are satisfied.

### B.2.4 Algorithm

Here we give the algorithm to compute the fast edgelet-like transform. Note when the original image is  $N$  by  $N$ , our algorithm generates an  $N \times 2N$  matrix. Recall that  $I(i, j)$  denotes the image value at pixel  $(i, j)$ ,  $1 \leq i, j \leq N$ .

#### Computing the discrete Radon transform via Fourier transform

A. Compute the first half of the DRT matrix (size is  $N$  by  $N$ ):

1. **For**  $j = 0$  **to**  $N - 1$  **step** 1,

take DFT of  $(j + 1)$ th column:

$$\begin{aligned} & [g(0, j), g(1, j), \dots, g(N - 1, j)] \\ & = DFT_N([I(1, j + 1), I(2, j + 1), \dots, I(N, j + 1)]); \end{aligned}$$

**End;**

2. **For**  $k = 0$  **to**  $N - 1$  **step** 1,

(a) pointwise multiply row  $k + 1$ ,  $[g(k, 0), g(k, 1), \dots, g(k, N - 1)]$ ,

$$\text{with complex sequence } \left\{ e^{-2\pi\sqrt{-1}\left(j \cdot s(k) + \Delta(k) \frac{j^2}{2}\right)} : j = 0, 1, \dots, N - 1 \right\};$$

(b) convolve it with complex sequence

$$\left\{ e^{\pi\sqrt{-1}\Delta(k)(t)^2} : t = 0, 1, \dots, N - 1 \right\};$$

(c) pointwise multiply the sequence with complex sequence

$$\left\{ e^{-2\pi\sqrt{-1}\Delta(k) \frac{l^2}{2}} : l = 0, 1, \dots, N - 1 \right\};$$

we get another row vector:  $\left\{ F\left(\frac{k}{N}, s(k) + l \cdot \Delta(k)\right) : l = 0, 1, \dots, N - 1 \right\}$ ;

**End;**

3. **For**  $l = 0$  **to**  $N - 1$  **step** 1,

taper the  $(l + 1)$ th column by the raised cosine, then take inverse 1-D DFT;

**End;**

B. Compute the second half of the DRT matrix (size is  $N$  by  $N$ ):

1. **For**  $i = 0$  **to**  $N - 1$  **step** 1,

**For**  $j = 0$  **to**  $N - 1$  **step** 1,

$$\text{modulate: } g(i, j) = I(i + 1, j + 1)e^{-2\pi\sqrt{-1}\frac{j}{N}};$$

- End;**
- End;**
2. **For**  $i = 0$  **to**  $N - 1$  **step** 1,  
 take 1-D DFT of the  $(i + 1)$ th row:  

$$[g(i, 0), g(i, 1), \dots, g(i, N - 1)]$$

$$= DFT_N([I(i + 1, 1), I(i + 1, 2), \dots, I(i + 1, N)]);$$
- End;**
3. flip matrix  $(g(i, k))_{0 \leq i, k \leq N-1}$  by columns:  

$$h(i, k) = g(i, N - 1 - k);$$
4. **For**  $k = 0$  **to**  $N - 1$  **step** 1,  
 (a) pointwise multiply  $(k + 1)$ th column,  $\{h(0, k), h(1, k), \dots, h(N - 1, k)\}$ ,  
 with complex sequence  $\left\{ e^{-2\pi\sqrt{-1}\left(i \cdot s(k) + \Delta(k) \frac{i^2}{2}\right)} : i = 0, 1, \dots, N - 1 \right\}$ ;  
 (b) convolve it with complex sequence  

$$\left\{ e^{\pi\sqrt{-1}\Delta(k)t^2} : t = 0, 1, \dots, N - 1 \right\};$$
  
 (c) pointwise multiply the sequence with complex sequence  

$$\left\{ e^{-2\pi\sqrt{-1}\Delta(k) \frac{l^2}{2}} : l = 0, 1, \dots, N - 1 \right\};$$
- we get another column vector:
- $$\left\{ F(s(k) + l \cdot \Delta(k), \frac{N - 1 - k}{N}) : l = 0, 1, \dots, N - 1 \right\};$$
- End;**
5. take transpose;
6. **For**  $l = 0$  **to**  $N - 1$  **step** 1,  
 taper the  $(l + 1)$ th column by the raised cosine function, then take  
 inverse 1-D DFT;
- End;**

### B.3 Adjoint of the Fast Transform

First of all, note that the previous fast algorithm is a linear transform, and so is every step in it. It is obvious that if we take the adjoint of each step, and do them in reverse order,

then we get the adjoint of the fast edgelet-like transform.

Note the superscript  $*$  denotes the adjoint transform. Table B.1 gives some transforms and their adjoints.

Transform	Adjoint transform
$x \mapsto \begin{bmatrix} \mathbf{T}_1(x) \\ \mathbf{T}_2(x) \end{bmatrix};$	$x \mapsto \mathbf{T}_1^*(x) + \mathbf{T}_2^*(x);$
taking 1-D DFT by row	taking inverse 1-D DFT by row
taking 1-D DFT by column	taking inverse 1-D DFT by column
reverse rows	reverse rows
reverse columns	reverse columns
matrix transpose (rotation)	matrix transpose (rotation)

Table B.1: Pairs of transforms and their adjoint

To find the adjoint of step 2 of the algorithm in computing the first half, which is also the adjoint of step 3 in computing the second half, we develop the following linear algebra point of view. Let  $x = (x_1, x_2, \dots, x_N)^T$  be a column vector. For fixed  $k$ , after taking step two (or three for the second half), we get column vector  $y = (y_1, y_2, \dots, y_N)^T$ . Vectors  $x$  and  $y$  satisfy the following equation:

$$y = \begin{pmatrix} e^{-2\pi\sqrt{-1}\Delta(k)\frac{0^2}{2}} \\ \ddots \\ e^{-2\pi\sqrt{-1}\Delta(k)\frac{(N-1)^2}{2}} \end{pmatrix} \begin{pmatrix} e^{\pi\sqrt{-1}\Delta(k)\cdot 0^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 2^2} & \dots & e^{\pi\sqrt{-1}\Delta(k)\cdot (N-1)^2} \\ e^{\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 0^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 1^2} & \dots & e^{\pi\sqrt{-1}\Delta(k)\cdot (N-2)^2} \\ e^{\pi\sqrt{-1}\Delta(k)\cdot 2^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot 0^2} & & \\ \vdots & & & \ddots & \vdots \\ e^{\pi\sqrt{-1}\Delta(k)\cdot (N-1)^2} & e^{\pi\sqrt{-1}\Delta(k)\cdot (N-2)^2} & & \dots & e^{\pi\sqrt{-1}\Delta(k)\cdot 0^2} \end{pmatrix} \begin{pmatrix} e^{-2\pi\sqrt{-1}(0\cdot s(k)+\Delta(k)\frac{0^2}{2})} \\ \ddots \\ e^{-2\pi\sqrt{-1}((N-1)\cdot s(k)+\Delta(k)\frac{(N-1)^2}{2})} \end{pmatrix} x.$$

The adjoint transform of this step corresponds to multiplying with the adjoint of the

above matrix, which is

$$\begin{pmatrix} e^{2\pi\sqrt{-1}(0\cdot s(k)+\Delta(k)\frac{0^2}{2})} & & & & \\ & \ddots & & & \\ & & e^{2\pi\sqrt{-1}((N-1)\cdot s(k)+\Delta(k)\frac{(N-1)^2}{2})} & & \\ \left( \begin{array}{cccccc} e^{-\pi\sqrt{-1}\Delta(k)\cdot 0^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 2^2} & \dots & e^{-\pi\sqrt{-1}\Delta(k)\cdot (N-1)^2} \\ e^{-\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 0^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 1^2} & \dots & e^{-\pi\sqrt{-1}\Delta(k)\cdot (N-2)^2} \\ e^{-\pi\sqrt{-1}\Delta(k)\cdot 2^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 1^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot 0^2} & & \\ \vdots & & & \ddots & \vdots \\ e^{-\pi\sqrt{-1}\Delta(k)\cdot (N-1)^2} & e^{-\pi\sqrt{-1}\Delta(k)\cdot (N-2)^2} & & \dots & e^{-\pi\sqrt{-1}\Delta(k)\cdot 0^2} \end{array} \right) \\ \left( \begin{array}{cccccc} e^{2\pi\sqrt{-1}\Delta(k)\frac{0^2}{2}} & & & & \\ & \ddots & & & \\ & & e^{2\pi\sqrt{-1}\Delta(k)\frac{(N-1)^2}{2}} & & \end{array} \right) \end{pmatrix}.$$

From all the above, we can derive the adjoint of the transform.

## B.4 Analysis

### B.4.1 Storage and Computational Complexity

In this section, we estimate some computing parameters.

Let  $c_N$  denote the number of operations to implement an  $N$ -point fast DFT. It is well known that  $c_N = O(N \log N)$ . First, let's consider the complexity of computing the first half of the DRT matrix. The first step is  $N$  times  $N$ -point DFTs with complexity  $Nc_N$ . In the second step, the convolution can be done via three times  $2N$ -point DFT, so the total computational effort will be  $2N^2$  multiplications and  $3N$  times  $2N$ -point DFTs, which has complexity  $3Nc_{2N} \approx 12Nc_N$ . The third step is  $N$  inverse  $N$ -point DFTs. From all above, the total complexity is  $2N^2 + 14Nc_N$ .

In computing the second half of the DRT matrix, except the flip step and transpose step, the rest of the algorithm is the same as for the first half, so the computing effort has the same complexity.

The overall effort to implement the fast transform for an  $N$  by  $N$  matrix is roughly  $28N$  times the effort of computing an  $N$ -point discrete Fourier transform ( $28Nc_N$ ), which is also

28 times the effort of doing the 2-D fast Fourier transform for an  $N$  by  $N$  matrix.

The storage needed for the previous algorithm is proportional to  $N^2$ .

### B.4.2 Effective Region

The way we sample in the 2-D Fourier domain is actually equivalent to sampling in the region surrounded by a dashed curve in Figure B.2. A column of the output of the fast algorithm is an equally-spaced sample of every straight line passing through the origin within this region. The dashed curve is made by four half circles. The reason that the region looks like this is simple: dilation in the Fourier domain is equivalent to shrinkage in the time domain with the same factor.

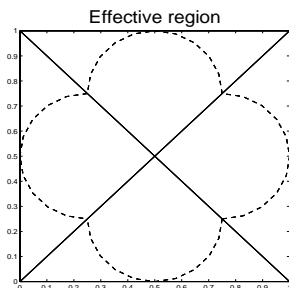
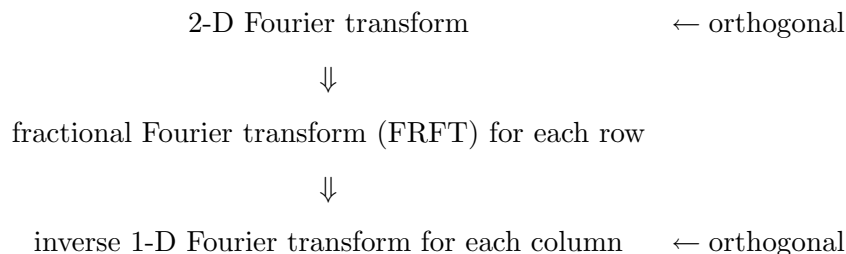


Figure B.2: Effective region.

### B.4.3 Ill-conditioning of the Fast X-interpolation Transform

The fast X-interpolation transform can be divided into three steps: if we only consider the first half of the matrix, we have



Note the matrix associated with the fractional Fourier transform may have a large condition number. We leave further discussion as future research.

## B.5 Examples

### B.5.1 Basic Elements for the Fast Edgelet-like Transform

For any linear transform, we can regard the coefficients as inner products of the input signal with given basic elements. If it is an isometric transform, then these basic elements form an orthonormal basis. Note our fast edgelet-like transform is redundant. Hence the corresponding set of basic elements does not form a basis.

In Figure B.3, we plot some basic elements of our transform. Note we actually apply our transform for square images with different size (scale). In the first (second, third) row, the squared images have sides  $\frac{1}{4}(\frac{1}{2}, 1)$  of the side of the original image. We intentionally renormalize the basic elements so that each of them should have  $\ell^2$  norm roughly equal to 1.

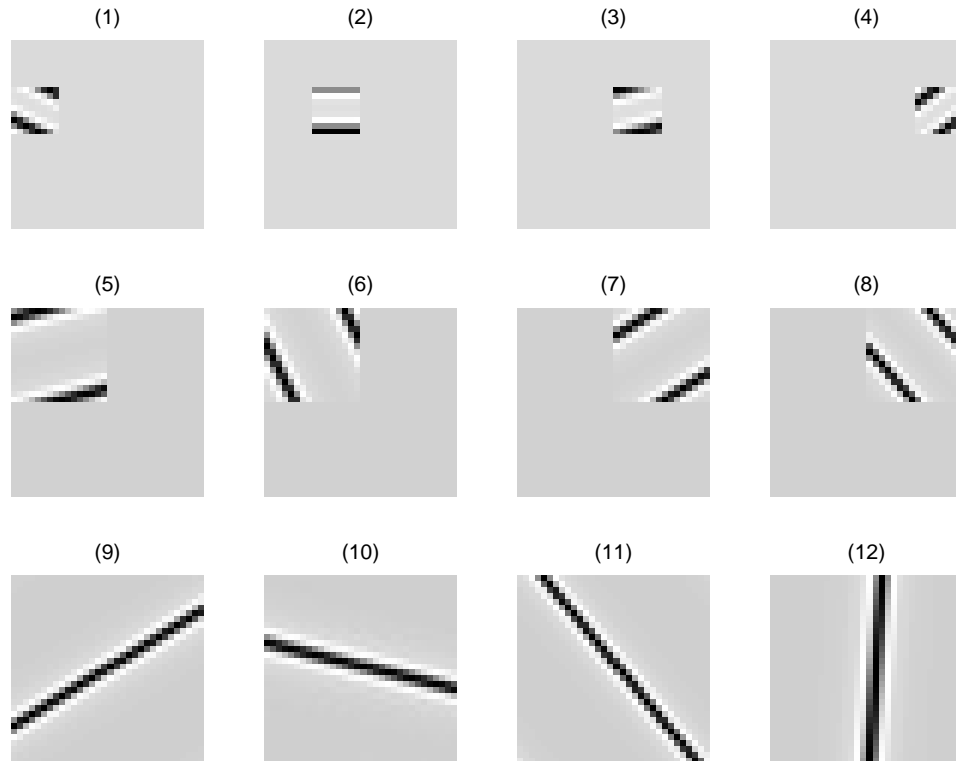


Figure B.3: Basic elements of the fast edgelet-like transform.

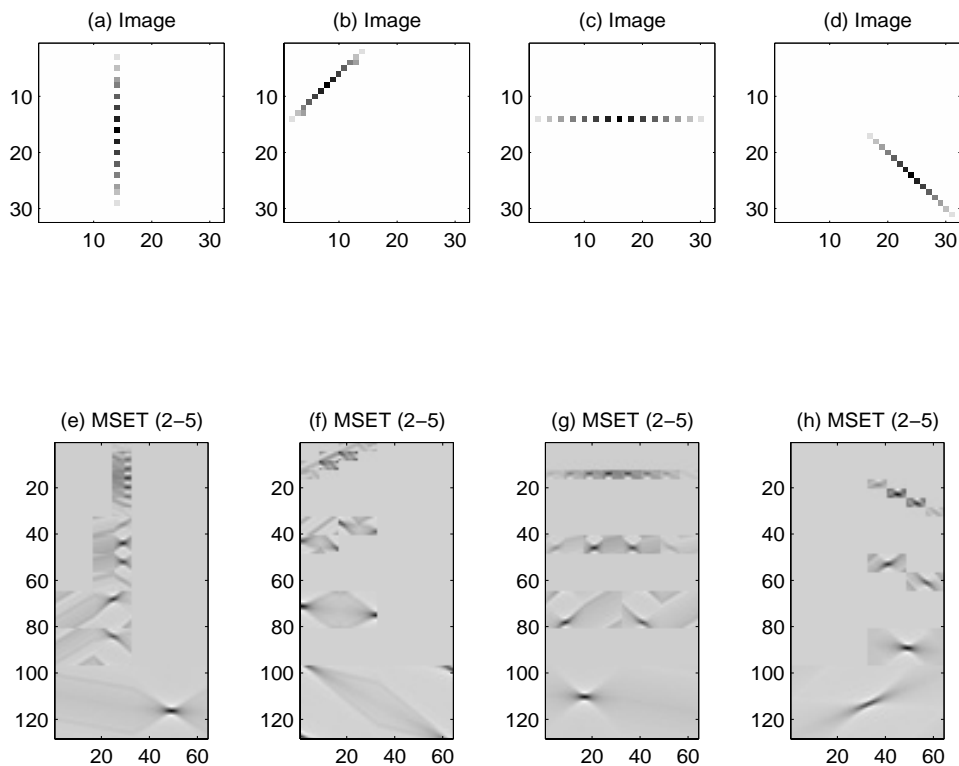


Figure B.4: Multiscale fast edgelet-like transform of artificial needle-like images.

### B.5.2 Edgelet-like Transform for Some Artificial Images

Since this algorithm is designed to capture the linear features in an image, it will be interesting to see how it works on some artificial images made by a single linear singularity.

The first row of Figure B.4 shows some needle-like images. The second row is their *multiscale* fast edgelet-like transforms. To explain the images of the coefficients, we need to explain how we arrange the output of our algorithm. Suppose we only do the transform for the whole image (scale-0 transform). Then as in B.2.4, the input  $I$  is mapped to a coefficient matrix with two components  $[E_1^1, E_1^2]$ . When we divide the image into  $2 \times 2$  block images (scale-1 transform), each subimage is mapped to a coefficient sub-matrix with two components:

$$\begin{bmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{bmatrix} \rightarrow \begin{bmatrix} E_{11}^1 & E_{11}^2 & E_{12}^1 & E_{12}^2 \\ E_{21}^1 & E_{21}^2 & E_{22}^1 & E_{22}^2 \end{bmatrix}.$$

And so on. Note for a fixed scale, the output matrix has the same number of rows, but the number of columns expands by 2. The second row in Figure B.4 is illustrations of output by taking four scales. So the number of rows are expanded by 4. The dark points are associated with coefficients with large amplitudes. Obviously there are few coefficients with significantly large amplitudes and all the rest are small. This matches our original goal to design this transformation.

### B.5.3 Edgelet-like Transform for Some Real Images

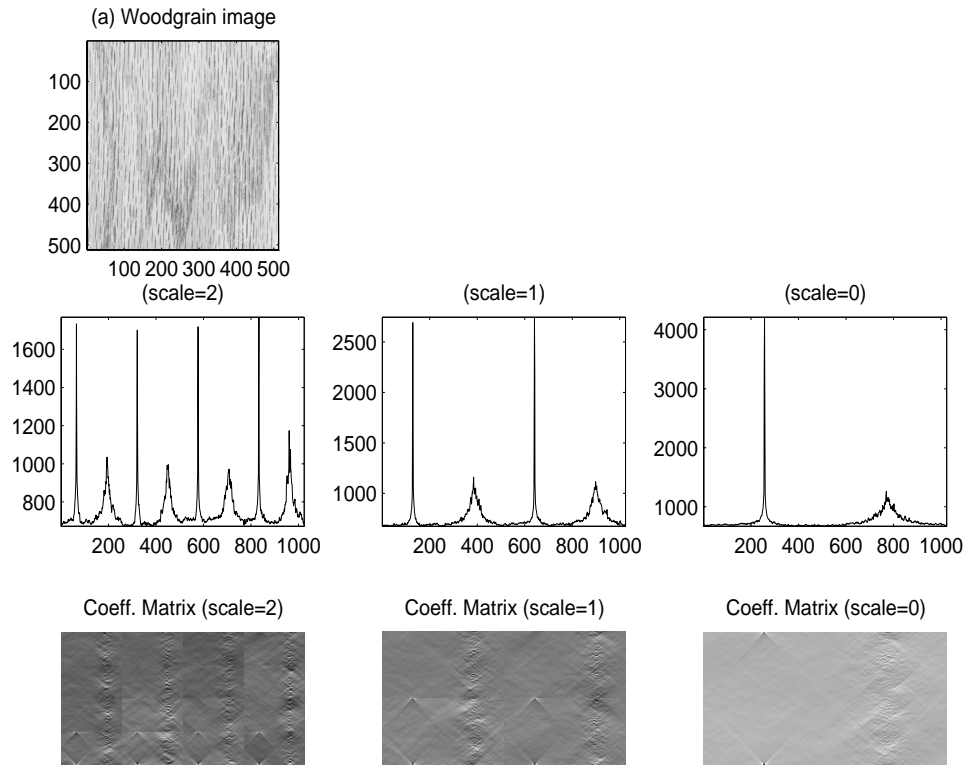


Figure B.5: Fast edgelet-like transform of wood grain image.

Here *real* means that they are from some other independent sources and are not made intentionally for our method. In these images, the linear features are embedded, and we see that our transform still captures them.

Figure B.5 is a wood grain image. There are many needle-like objects in the image, and they are pretty much at the the same scale—having the same length—and along the same direction. We do our transform at three scales (0, 1, 2 corresponding to no division of the

image, divided by two, and divided by four). The upper-left image is the original. The first row gives the columnwise maximum of the absolute values of the coefficients. We see that there are strong patterns in the columnwise maximums; in particular, it becomes large along the direction that is the direction of most of the needle-like features in the image. The second row are the coefficient matrices at different scales.

It takes about 40 seconds to carry out this transform on an SGI Onyx workstation.

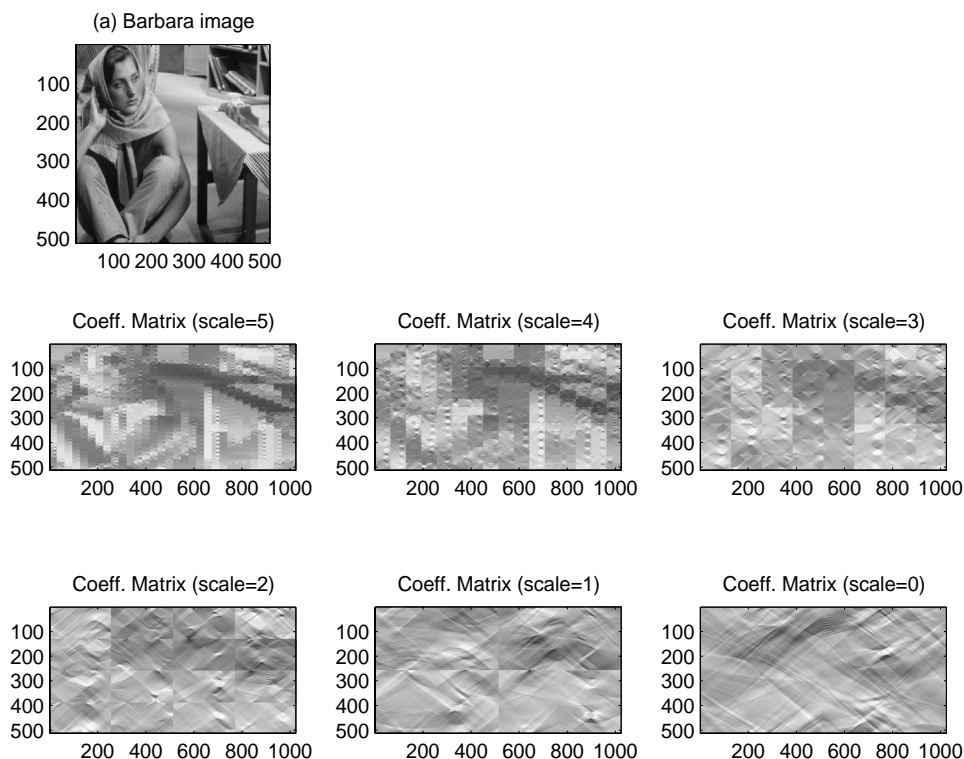


Figure B.6: Fast edgelet-like transform of wood grain image.

Another example is the Barbara image. Again, the upper-left image in Figure B.6 is the original image. The remaining images are the coefficient matrices corresponding to scale 0 through 5. The dark area corresponds to the significant coefficients. We observe that when the scale is increased, we have more significant coefficients. See the coefficient matrix at scale equal to 5. This implies that when we divide the image into small squares, the linear features become more dramatic, hence it becomes easier for a monoscale fast edgelet-like transform to capture them. Further discussion is beyond the scope of this chapter; we will leave it as future research.

It takes about 60 seconds to carry out this transform on an SGI workstation.

## B.6 Miscellaneous

### B.6.1 Examples of Interpolation Functions

We present three examples of the interpolation functions  $\rho$  and  $\hat{\rho}$ .

1. We can choose  $\hat{\rho}(\cdot)$  to be a rectangular function

$$\hat{\rho}(\xi) = \begin{cases} 1, & -\frac{1}{2} < \xi \leq \frac{1}{2}; \\ 0, & \text{otherwise.} \end{cases}$$

Then  $\rho$  is the sinc function

$$\rho(x) = \int \hat{\rho}(\xi) e^{2\pi\sqrt{-1}\xi x} d\xi = \frac{\sin(\pi x)}{\pi x}.$$

Figure B.7 illustrates the sinc function and its Fourier transform.

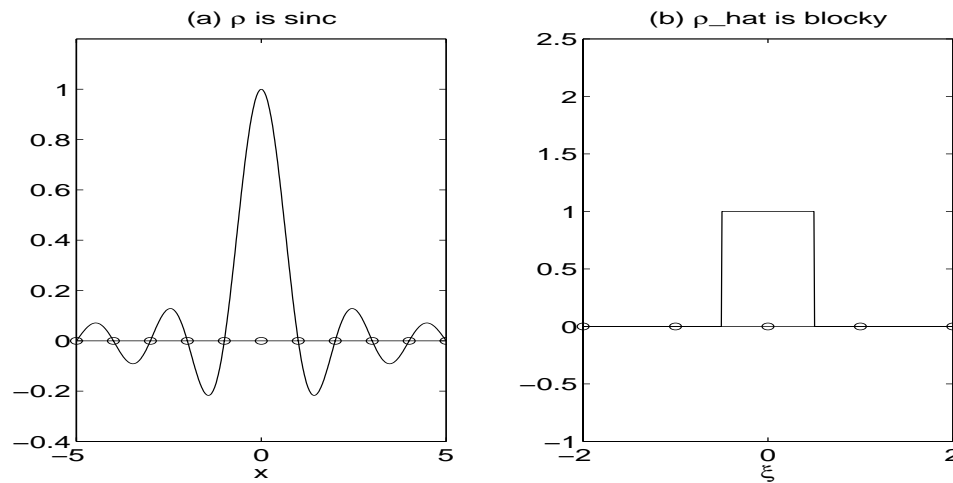


Figure B.7: Sinc function in (a) and its Fourier transform—blocky function in (b).

2. We can also choose  $\hat{\rho}(\cdot)$  to be the raised cosine function,

$$\hat{\rho}(\xi) = \begin{cases} \frac{1}{2} + \frac{1}{2} \cos(2\pi\xi), & -\frac{1}{2} < \xi \leq \frac{1}{2}; \\ 0, & \text{otherwise.} \end{cases}$$

So  $\rho$  is

$$\rho(x) = \int \hat{\rho}(\xi) e^{2\pi\sqrt{-1}\xi x} d\xi = -\frac{\sin(\pi x)}{2\pi x(x^2 - 1)}.$$

Note in this case,

$$\rho(0) = 1, \rho(1) = \rho(-1) = \frac{1}{2}, \text{ and } \rho(i) = 0, i = \pm 2, \pm 3, \dots$$

Figure B.8 shows the raised cosine function and the corresponding  $\rho$  in the time domain.

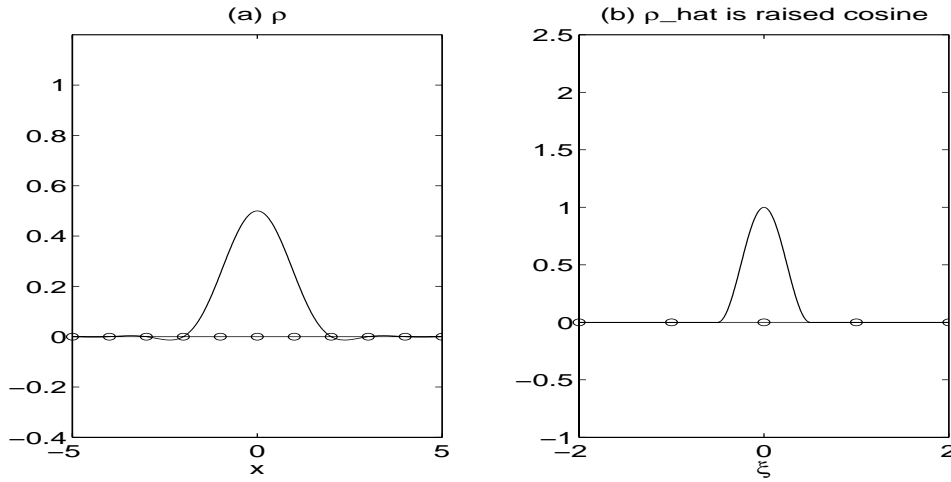


Figure B.8: Raised cosine function in (b) and its counterpart in time domain in (a).

3. In the first case, there is no tapering. In the second case, the tapering rate is 100%. It is interesting to look at something in between. Suppose  $p$  ( $0 < p < \frac{1}{2}$ ) is the tapering point, which means that  $\hat{\rho}$  is chosen to be

$$\hat{\rho}(\xi) = \begin{cases} 1, & |\xi| < p; \\ \frac{1}{2} + \frac{1}{2} \cos \frac{\pi(|\xi| - p)}{\frac{1}{2} - p}, & p < |\xi| < \frac{1}{2}; \\ 0, & \text{otherwise.} \end{cases}$$

In this case,

$$\text{tapering rate} = \frac{\frac{1}{2} - p}{\frac{1}{2}} = 100(1 - 2p)\%.$$

The function  $\rho$  is

$$\begin{aligned} \rho(x) &= \int \hat{\rho}(\xi) e^{2\pi\sqrt{-1}\xi x} d\xi \\ &= \int_{-p}^p \cos(2\pi\xi x) d\xi + \int_p^{1/2} \cos(2\pi\xi x) \left[ 1 + \cos \frac{2\pi(\xi - p)}{1 - 2p} \right] d\xi \\ &= \frac{\sin(\pi x) + \sin(2\pi p x)}{2\pi x} \frac{1}{1 - (1 - 2p)^2 x^2}. \end{aligned}$$

When  $p$  is  $1/2$ , we get the sinc function in case one. When  $p$  is  $0$ , we get the  $\rho$  function in case two. When  $p = \frac{1}{4}$ , the Figure B.9 shows how the function  $\rho$  and function  $\hat{\rho}$  look.

When  $p$  is  $1/4$  and function  $\rho(\cdot)$  only takes integer values, unlike the two previous cases, the function  $\rho$  does not have finite support.

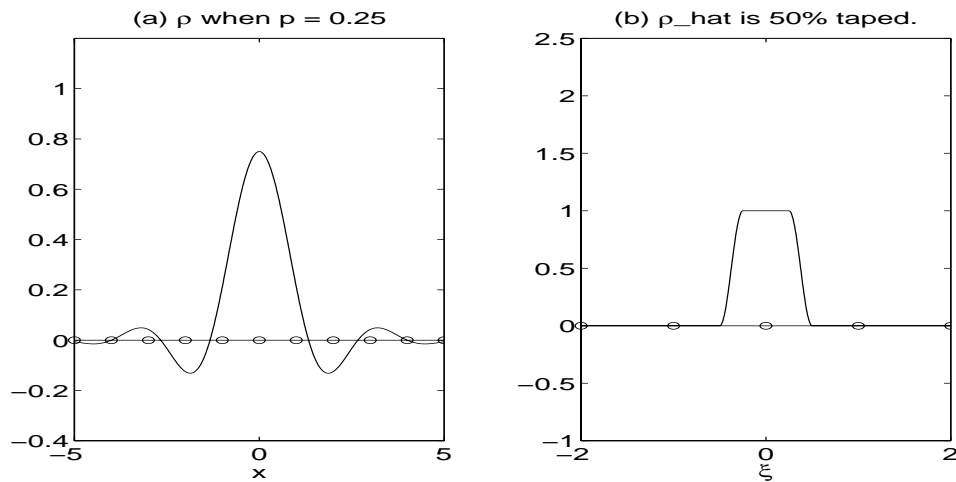


Figure B.9: Fifty percent tapered window function in (b) and its counterpart in time domain in (a).



# Bibliography

- [1] Sensor Data Management System web site. <http://www.mbvlab.wpafb.af.mil/public/sdms/>.
- [2] USC-SIPI Image Database. <http://sipi.usc.edu/services/database/Database.html>.
- [3] H. K. Aghajan. *Subspace Techniques for Image Understanding and Computer Vision*. PhD thesis, Stanford University, March 1995.
- [4] O. Axelsson. Incomplete block matrix factorization preconditioning methods: the ultimate answer? *J. Comput. Appl. Math.*, 12:3–18, 1985.
- [5] O. Axelsson. A general incomplete block-matrix factorization method. *Linear Algebra Appl.*, 74:179–90, 1986.
- [6] David H. Bailey and Paul N. Swarztrauber. The fractional Fourier transform and applications. *SIAM Rev.*, 33(3):389–404, 1991.
- [7] Richard G. Baraniuk and Douglas L. Jones. Shear madness: new orthonormal bases and frames using chirp functions. *IEEE Transactions on Signal Processing*, 41(12):3543–9, December 1993.
- [8] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.
- [9] Michele Benzi, Carl D. Meyer, and Miroslav Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.

- [10] T. Berger. *Rate Distortion Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [11] J. Bergh and J. Löfström. *Interpolation Spaces: An Introduction*. Springer-Verlag, 1976.
- [12] Julian Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Royal Statistical Society, Series B, Methodological*, 36:192–236, 1974.
- [13] G. Beylkin. Discrete radon transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(2):162–72, Feb. 1987.
- [14] G. Beylkin. On the fast Fourier transform of functions with singularities. *Applied and Computational Harmonic Analysis*, 2(4):363–81, 1995.
- [15] Peter Bloomfield. *Fourier Analysis of Time Series: An Introduction*. John Wiley & Sons, 1976.
- [16] Stephen Boyd. Ee364: Convex optimization, class notes. <http://www.stanford.edu/class/ee364/>, Winter 1997.
- [17] Ronald N. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill, 1986.
- [18] M. L. Brady. A fast discrete approximation algorithm for the radon transform. *SIAM J. Computing*, 27(1):107–19, February 1998.
- [19] E. Oran Brigham. *Fast Fourier Transform*. Prentice-Hall, 1974.
- [20] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, 1998.
- [21] Emmanuel J. Candès. *Ridgelets: Theory and Applications*. PhD thesis, Stanford University, 1998.
- [22] Emmanuel J. Candès. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis*, 6(2):197–218, 1999.
- [23] J. Capon. A probabilistic model for run-length coding of pictures. *IRE Transactions on Information Theory*, pages 157–63, 1959.

- [24] C. Victor Chen and Hao Ling. Joint time-frequency analysis for radar signal and image processing. *IEEE Signal Processing Magazine*, 16(2):81–93, March 1999.
- [25] Scott Shaobing Chen. *Basis Pursuit*. PhD thesis, Stanford University, November 1995.
- [26] Scott Shaobing Chen, David Donoho, and Michael A. Saunders. *About Atomizer*. Stanford University, April 1996.
- [27] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Scientific Computing*, 20(1):33–61, 1999. electronic.
- [28] Wen-Hsiung Chen, C. H. Smith, and S. C. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, Com-25(9):1004–9, Sept. 1977.
- [29] Xiru Chen, Lincheng Zhao, and Yuehua Wu. On conditions of consistency of  $ml_1$ n estimates. *Statistical Sinica*, 3:9–18, 1993.
- [30] Charles Chui. class notes for stat323: Wavelets and beyond, with applications. Stanford University, Spring 1998.
- [31] P. Concus, G. H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6(1):220–52, 1985.
- [32] J. W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
- [33] Thomas M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.
- [34] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.
- [35] Geoffrey Davis. *Adaptive Nonlinear Approximations*. PhD thesis, Courant Institute of Mathematical Sciences, September 1994.
- [36] A.H. Delaney and Y. Bresler. Globally convergent edge-preserving regularized reconstruction: an application to limited-angle tomography. *IEEE Transactions on Image Processing*, 7(2):204–21, 1998.

- [37] E. M. Deloraine and A. H. Reeves. The 25th anniversary of pulse code modulation. *IEEE Spectrum*, pages 56–64, May 1965.
- [38] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–8, 1982.
- [39] R. S. Dembo and T. Steihaug. Truncated Newton algorithms for large-scale unconstrained optimization. *Math. Programming*, 26(2):190–212, 1983.
- [40] R.A. DeVore and V.N. Temlyakov. Some remarks on greedy algorithms. *Advances in Computational Mathematics*, 5(2-3):173–87, 1996.
- [41] Y. Dodge. *Statistical Data Analysis: Based on the  $L_1$ -norm and Related Methods*. North-Holland, 1987.
- [42] D. L Donoho and etc. *WaveLab*. Stanford University, Stanford, CA, .701 edition. <http://www-stat.stanford.edu/~wavelab>.
- [43] David Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. Working paper, June 1999.
- [44] David L. Donoho. Interpolating wavelet transforms. Technical report, Stanford University, 1992.
- [45] David L. Donoho. Smooth wavelet decompositions with blocky coefficient kernels. Technical report, Stanford University, 1993.
- [46] David L. Donoho. Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and Computational Harmonic Analysis*, 1(1):100–15, 1993.
- [47] David L. Donoho. Unconditional bases and bit-level compression. *Applied and Computational Harmonic Analysis*, 3(4):388–92, 1996.
- [48] David L. Donoho. Fast ridgelet transforms in dimension 2. Personal copy, April 30 1997.
- [49] David L. Donoho. *Digital ridgelet transform via digital polar coordinate transform*. Stanford University, 1998.

- [50] David L. Donoho. Fast edgelet transforms and applications. Manuscript, September 1998.
- [51] David L. Donoho. *Orthonormal Ridgelets and Linear Singularities*. Stanford University, 1998.
- [52] David L. Donoho. *Ridge functions and orthonormal ridgelets*. Stanford University, 1998. <http://www-stat.stanford.edu/~donoho/Reports/index.html>.
- [53] David L. Donoho. Sparse components of images and optimal atomic decompositions. Technical report, Stanford University, December 1998. <http://www-stat/~donoho/Reports/1998/SCA.ps>.
- [54] David L. Donoho. Curvelets. Personal copy, April 1999.
- [55] David L. Donoho and Stark P. B. Uncertainty principles and signal recovery. *SIAM J. Applied Mathematics*, 49(3):906–31, 1989.
- [56] David L. Donoho and Peter J. Huber. The notion of breakdown point. In *A Festschrift for Erich L. Lehmann*, pages 157–84. Wadsworth Advanced Books and Software, 1983.
- [57] David L. Donoho and B. F. Logan. Signal recovery and the large sieve. *SIAM J. Applied Mathematics*, 52(2):577–91, April 1992.
- [58] David L. Donoho, Stéphane Mallat, and R. von Sachs. Estimating covariance of locally stationary processes: Consistency of best basis methods. In *Proceedings of IEEE Time-Frequency and Time-Scale Symposium*, Paris, July 1996.
- [59] David L. Donoho, Stéphane Mallat, and R. von Sachs. *Estimating Covariance of Locally Stationary Processes: Rates of Convergence of Best Basis Methods*, February 1998.
- [60] Serge Dubuc and Gilles Deslauriers, editors. *Spline Functions and the Theory of Wavelets*. American Mathematical Society, 1999.
- [61] P. Duhamel and C. Guillemot. Polynomial transform computation of the 2-d dct. In *ICASSP 1990, International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1515–18, 1990.

- [62] P. Duhamel and H. H'Mida. New  $2^n$  DCT algorithms suitable for VLSI implementation. In *Proceedings: ICASSP 87. 1987 International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1805–8, 1987.
- [63] P. Duhamel and M. Vetterli. Improved Fourier and Hartley transform algorithms: Application to cyclic convolution of real data. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(6):818–24, 1987.
- [64] Douglas F. Elliott and K. Ramamohan Rao. *Fast transforms: algorithms, analyses, applications*. Academic Press, 1982.
- [65] S. P. Ellis and S. Morgenthaler. Leverage and breakdown in  $l_1$  regression. *J. the American Statistical Association*, 87:143–48, 1992.
- [66] E. Feig and S. Winograd. Fast algorithms for the discrete cosine transform. *IEEE Transactions on Signal Processing*, 40(9):2174–93, September 1992.
- [67] E. Feig and S. Winograd. On the multiplicative complexity of discrete cosine transforms. *IEEE Transactions on Information Theory*, 38(4):1387–91, July 1992.
- [68] Patrick Flandrin. *Time-frequency Time-scale Analysis*. Academic Press, 1999.
- [69] D. Gabor. Theory of communication. *J. IEE*, 93:429–57, 1946.
- [70] A. Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.
- [71] P. E. Gill, W. Murray, and M. A. Saunders. *User's Guide for SNOPT 5.3: A Fortran Package for Large-Scale Nonlinear Programming*, May 20 1998. Draft.
- [72] Gene Golub, Anne Greenbaum, and Mitchell Luskin, editors. *Recent Advances in Iterative Methods*, volume 60 of *IMA volumes in mathematics and its applications*. Springer-Verlag, 1994. Papers from the IMA Workshop on Iterative Methods for Sparse and Structured Problems, held in Minneapolis, Minn., Feb. 24-Mar. 1, 1992.
- [73] Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31(1):50–102, 1989.
- [74] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

- [75] Vivek K. Goyal. *Beyond Traditional Transform Coding*. PhD thesis, University of California, Berkeley, December 1998. <http://cm.bell-labs.com/cm/ms/who/vivek/Thesis/>.
- [76] Robert M. Gray. Toeplitz and circulant matrices: A review. <http://www-isl.stanford.edu/~gray/toeplitz.html>, 1998.
- [77] Robert M. Gray and D. L. Neuhoff. Quantization. *IEEE Transaction on Information Theory*, 44(6):2325–83, October 1998.
- [78] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*, volume 17 of *Frontiers in Applied Mathematics*. SIAM, 1997.
- [79] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley Series in Pro. and Math. Sci., 1986.
- [80] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12:69–82, 1970.
- [81] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.
- [82] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educational Psychology*, 24:417–441, 498–520, 1933.
- [83] TaiChiu Hsung, Daniel P.K. Lun, and Wan-Chi Siu. The discrete periodic radon transform. *IEEE Transactions on Signal Processing*, 44(10):2651–7, October 1996.
- [84] J. Huang. *Quantization of Correlated Random Variables*. PhD thesis, Yale University, New Haven, CT, 1962.
- [85] J.-Y. Huang and P. M. Schultheiss. Block quantization of correlated gaussian random variables. *IEEE Trans. Comm.*, 11:289–96, September 1963.
- [86] Peter J. Huber. Fisher information and spline interpolation. *Annals of Statistics*, 2:1029–33, 1974.
- [87] Peter J. Huber. *Robust Statistical Procedures*, volume 27. CBMS-NSF, 1977.

- [88] Peter J. Huber. *Robust Statistics*. Wiley Series in Pro. and Math. Sci., 1981.
- [89] S. Jaggi, W.C. Karl, S. Mallat, and A.S. Willsky. High-resolution pursuit for feature extraction. *Applied and Computational Harmonic Analysis*, 5(4):428–49, October 1998. <http://www.cmap.polytechnique.fr/~mallat/biblio.html>.
- [90] R. A. Kennedy and Z. Ding. Blind adaptive equalizer for quadrature amplitude modulation communication systems based on convex cost functions. *Optical Engineering*, 31(6):1189–99, June 1992.
- [91] R. Koenker.  $l_1$  tortoise gains on  $l_2$  hare. *Statistical Computing and Graphics*, 8(2/3):17–24, 1997. <http://cm.bell-labs.com/cm/ms/who/cocteau/newsletter/index.html>.
- [92] B.G. Lee. A new algorithm to compute the discrete cosine transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-32(6):1243–5, 1984.
- [93] W. Li and J.J. Swetits. The linear  $l_1$  estimator and the Huber m-estimator. *SIAM J. Optim.*, 8(2):457–75, May 1998.
- [94] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Th.*, IT-28(2):129–37, March 1982. Originally an unpublished Bell Telephone Laboratories tech. memo., 1957.
- [95] M. Lobo and M. Fazel. Group presentation. ISL Stanford University, 1999.
- [96] B.F. Logan. *Properties of high-pass signals*. PhD thesis, Columbia University, New York, 1965.
- [97] H. Lu. A generalized Hilbert matrix problem and confluent Chebyshev-Vandermonde systems. *Siam Journal on Matrix Analysis and Applications*, 19(1):253–76, January 1998.
- [98] David G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, 1973.
- [99] Stéphane Mallat. Applied mathematics meets signal processing. In *Proceedings of the International Congress of Mathematicians*, Berlin, 1998.

- [100] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.
- [101] Stéphane Mallat, George Papanicolaou, and Z. Zhang. Adaptive covariance estimation of locally stationary processes. *Annals of Statistics*, 26(1):1–47, February 1998.
- [102] Stéphane Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–415, December 1993.
- [103] S. Mann and S. Haykin. The chirplet transform: physical considerations. *IEEE Transactions on Signal Processing*, 43(11):2745–61, November 1995.
- [104] J. Max. Quantizing for minimum distortion. *IRE Trans. Inform. Th.*, IT-6(1):7–12, March 1960.
- [105] Francois G. Meyer and Ronald R. Coifman. Brushlets: a tool for directional image analysis and image compression. *Applied and Computational Harmonic Analysis*, 4:147–187, 1997.
- [106] R. R. Meyer. Sufficient conditions for the convergence of monotonic mathematical programming algorithms. *J. Comput. System. Sci.*, 12(1):108–21, 1976.
- [107] Yves Meyer. *Wavelets, Algorithms & Applications*. SIAM, 1993. Translated and revised by Robert Ryan.
- [108] C. Michelot and M.L. Bougeard. Duality results and proximal solutions of the Huber M-estimator problem. *Applied Mathematics & Optimization*, 30:203–21, 1994.
- [109] D. Mihovilovic and R. N. Bracewell. Adaptive chirplet representation of signals on time-frequency plane. *Electronics Letters*, 27(13):1159–61, June 1991.
- [110] José M.F. Moura and Nikhil Balram. Recursive structure of noncausal Gauss-Markov random fields. *IEEE Transactions on Information Theory*, 38(2):334–54, March 1992.
- [111] José. M.F. Moura and Marcelo G.S. Bruno. DCT/DST and Gauss-Markov fields: conditions for equivalence. *IEEE Transactions on Signal Processing*, 46(9):2571–4, September 1998.
- [112] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995.

- [113] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.
- [114] Henry J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer-Verlag, 1982.
- [115] Kramer H. P. and Mathews M. V. A linear coding from transmitting a set of correlated signals. *IRE Trans. Inform. Theory*, 2:41–46, September 1956.
- [116] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–29, 1975.
- [117] Christopher C. Paige and Michael A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.
- [118] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, second edition, 1995.
- [119] A. G. Ramm and A. I. Katsevich. *Radon Transform and Local Tomography*. CRC Press, 1996.
- [120] B. D. Rao. Signal processing with the sparseness constraint. In *Proceedings of ICASSP*, pages III–1861–4, 1998.
- [121] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.
- [122] A. H. Reeves. French Patent No. 852,183, October 3 1938.
- [123] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
- [124] S. Sardy, A. Bruce, and P. Tseng. Block coordinate relaxation methods for non-parametric signal denoising with wavelet dictionaries. Web page, October 1998. <http://www.math.washington.edu/~tseng/papers.html>.
- [125] S. Sardy, A. G. Bruce, and P. Tseng. Block coordinate relaxation methods for non-parametric signal de-noising. Received from E. Candès, October 1998.
- [126] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, 1996.

- [127] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–56, October 1948.
- [128] J. E. Spingarn. Partial inverse of a monotone operator. *Applied Mathematics and Optimization*, 10:247–65, 1983.
- [129] Frank Spitzer. Markov random fields and Gibbs ensembles. *American Mathematical Monthly*, 78:142–54, February 1971.
- [130] A.S. Stern, D.L Donoho, and Hoch J.C. Iterative thresholding and minimum  $\ell^1$ -norm reconstruction. based on personal communication, 1996 or later.
- [131] Mann Steve and Simon Haykin. Adaptive “chirplet” transform: an adaptive generalization of the wavelet transform. *Optical Engineering*, 31(6):1243–56, June 1992.
- [132] Gilbert Strang. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.
- [133] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *J. the Royal Statistical Society, Series B*, 58:267–288, 1996.
- [134] Richard Tolimieri, Myoung An, and Chao Lu. *Algorithms for Discrete Fourier Transform and Convolution*. Springer, 2nd edition, 1997.
- [135] Paul Tseng. Dual coordinate ascent methods for non-strictly convex minimization. *Mathematical Programming*, 59:231–47, 1993.
- [136] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM, 1992.
- [137] R. J. Vanderbei. *Linear Programming*. Kluwer Academic Publishers, 1996.
- [138] S. Vembu, S. Verdú, and Y. Steinberg. The source-channel separation theorem revisited. *IEEE Trans. on Inform. Theory*, 41(1):44–54, Jan. 1995.
- [139] Z. Wang and B.R. Hunt. Comparative performance of two different versions of the discrete cosine transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-32(2):450–3, 1984.
- [140] Zhongde Wang. Reconsideration of “a fast computational algorithm for the discrete cosine transform”. *IEEE Transactions on Communications*, Com-31(1):121–3, Jan. 1983.

- [141] Mladen Victor Wickerhauser. Smooth localized orthonormal bases. *Comptes Rendus de l'Académie des Sciences de Paris*, 316:423–7, 1993.
- [142] Mladen Victor Wickerhauser. *Adapted wavelet analysis from theory to software*. Wellesley, 1994.
- [143] S. Winograd. On computing the discrete fourier transform. *Mathematics of Computation*, 32(141):175–99, 1978.
- [144] Margaret H. Wright. The interior-point revolution in constrained optimization. Technical Report 98-4-09, Bell Laboratories, Murray Hill, New Jersey 07974, June 1998. “<http://cm.bell-labs.com/cm/cs/doc/98/4-09.ps.gz>”.
- [145] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): towards a unified theory for texture modeling. *International J. Computer Vision*, 27(2):107–26, 1998.