

中图分类号： TP3

论文 编 号 : 10006GS1221E82

北京航空航天大學  
专业硕士学位论文

二维稀疏频谱表达算法及其应  
用验证

作者姓名 阎晨强

学科专业 软件工程

指导教师 王华锋

培养院系 软件学院

# **The Technology of Sparse Frequency Representation In Two Dimensions And Application Verification**

A Dissertation Submitted for the Degree of Master

**Candidate: Min Chenqiang**

**Supervisor: Wang Huafeng**

School of SoftWare

Beihang University, Beijing, China

中图分类号： TP3

论文编号： 10006GS1221E82

## 硕士 学位 论文

# 二维稀疏频谱表达算法及其应用验证

作者姓名 闵晨强 申请学位级别 软件工程硕士

指导教师姓名 王华锋 职 称 副教授

学科专业 软件工程 研究方向 移动云计算

学习时间自 2012 年 9 月 20 日 起至 2014 年 12 月 19 日止

论文提交日期 2014 年 11 月 25 日 论文答辩日期 2014 年 12 月 19 日

学位授予单位 北京航空航天大学 学位授予日期 年 月 日

## 关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：\_\_\_\_\_

日期： 年 月 日

## 学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：\_\_\_\_\_

日期： 年 月 日

指导教师签名：\_\_\_\_\_

日期： 年 月 日

## 摘要

信号处理往往结合信号在多个空间表达中的特点来寻找最有利的处理方法，时域信号在多个空间中得表达通过变换来完成。正交基变换就是一直常用且重要的变换方法，例如傅里叶变换不管在信号处理的早期，还是在新技术新方法层出不穷的现今。能够在现代计算机上计算离散信号傅里叶变换的快速傅里叶变换算法也被称为改变世界的经典算法之一。但随着信息时代信息的膨胀，更大规模的信号数据催生着更简洁的信号表达方法，更快的空间变换算法。对基于过完备字典集的稀疏信号表达方法的研究诞生了稀疏信号表示理论。其虽然能够简洁的表达信号，但这种信号表示无法提高类似的傅里叶分析方法，很多经典的信号处理方法无法继续使用。相比之下，傅里叶变换因其具有一些特殊性而无法被替代。为寻求更快的傅里叶变换方法，同时也利用稀疏信号表示理论，对稀疏傅里叶变换，稀疏傅里叶频谱表达方法的探索也由此展开。

本文基于信号处理基本的理论，以及离散傅里叶变换基础，结合最近几年稀疏傅里叶变换算法取得的重大突破，寻找一种适用于二维离散信号的稀疏傅里叶变换方法，希望通过该方法，能够简洁高效的得到二维信号的稀疏频谱，而不需要计算二维离散傅里叶变换。并且，在某些信号处理应用中，能够使用信号的稀疏频谱表达作为信号处理的输入，利用其稀疏的性质来削减输入规模，达到降维的目的。

本文首先研究了现有的较为完善和实用的一维稀疏傅里叶算法和一种稀疏信号表示求解方法。在此基础上，改进了一种现有的理想二维严格稀疏傅里叶变换方法，突破了其对一些理想情形的假设约束。在探索对二维图像信号的稀疏频谱表达方法中，借鉴了该改进理想方法的一些稀疏频谱计算策略，并结合图像频谱的分布特性，设计了一种针对图像噪声频谱的稀疏表达计算方法，该方法能够使用少数采样信号来快速高效的计算图像的稀疏频谱表达，突破了快速傅里叶变换的最低频率采样率。在对多种类型图像的实验中，所有稀疏频谱表达图像和源图像的结构相似度都在 80% 以上，大部分稀疏频谱表达图像的结构相似度高于 95%。对于较大的图像，计算稀疏频谱的性能也优于快速傅里叶变换。

最后，本文列举了本文研究的参与的两个系统应用，并表现出了较好的成果。期望对于图像稀疏频谱的研究会给数字图像处理带来新的生命。

**关键词：**信号处理，数字图像，稀疏，频谱，傅里叶变换

## Abstract

In signal processing, usually use many signal representation methods in different space to find a most effective representation . By this representation, we can process signal easily. We define a transform to get different space representations of signal. Orthogonal basis transform is always regard as a important transform, for example the fourier transform, no matter in the early days of signal processing, or by now, can always find its widely use in diffident signal processing applications. The FFT algorithm, which can compute fourier transform of discrete signals in log running time, is regard as the top ten most classic algorithms who had changed the world. But by the expansion of information, the larger signals urge more simple representations, and faster transform algorithms. Over complete directory provides a sparse representation theory. Even though this method can get signal's simple representation, but many classic signal processing method can not be use by this representation. As an important frequency analyze technology, fourier transform can't be replaced by any other simple representation of signal. Based on the sparse representation theory, the research to find more effective fourier transform algorithms which only output the sparse frequencies of signal become popular.

In this paper, we studied the most effective sparse fourier transform algorithms proposed by MIT recently, combined with the signal processing and discrete fourier transform theory, we try to come up with a different fourier transform method, which can handle two dimension signals, the method improve the origin two dimension fourier transform method proposed by MIT with less constraint conditions. Based on this, we develop another technology to handle the sparse frequency representation of digital image, although the images have more noise frequencies than the optimal signals. Our method can compute the sparse frequencies with less constraint. We proved our method in many experiments, all of them show that this method can representation image with little frequencies but with high accurate. In all experiments, the structural similarity is above 80%, and most of them are above 95%.

At last, we applied our method in two image processing applications and got good result. We also hope the sparse representation technology can bring new energy to image processing.

**Keywords:** signal process, digital image, sparse, frequency, fourier transform

# 目 录

<b>第一章 绪论 .....</b>	1
1.1 研究背景.....	1
1.2 国内外研究现状分析.....	2
1.2.1 稀疏傅里叶算法理论的发展 .....	3
1.2.2 稀疏傅里叶算法应用的发展 .....	5
1.3 论文的主要研究目标及内容.....	5
1.3.1 研究目标 .....	5
1.3.2 研究内容 .....	6
1.4 论文的组织结构.....	6
<b>第二章 信号稀疏表示理论及关键技术分析 .....</b>	8
2.1 信号稀疏表示理论概述.....	8
2.2 基于线性编码理论的信号稀疏表示方法分析.....	11
2.3 基于频域哈希的一维稀疏傅里叶算法分析.....	14
2.4 本章小结.....	17
<b>第三章 二维严格稀疏频谱表达算法及其改进 .....</b>	18
3.1 二维稀疏表达算法研究问题概述.....	18
3.2 二维严格稀疏频谱表达算法关键技术分析.....	19
3.2.1 二维严格稀疏频谱计算主要策略 .....	20
3.2.2 理想一维 <i>one-sparse</i> 频谱计算方法 .....	21
3.2.3 算法原理及问题分析.....	22
3.3 改进二维严格稀疏频谱表达算法的设计与实现.....	25
3.4 算法分析及实验结果.....	28
3.5 本章小结.....	31
<b>第四章 二维图像稀疏频谱表达算法的设计与实现 .....</b>	32
4.1 二维图像频谱的稀疏特性.....	32
4.2 二维图像稀疏频谱表达关键问题.....	35
4.3 关键问题解决方案.....	36
4.3.1 近似高斯分布稀疏频谱的计算策略.....	36
4.3.2 噪声对稀疏频谱的影响和处理方法.....	39

4.4 算法设计与实现.....	42
4.5 算法分析及优化.....	46
4.6 本章小结.....	49
<b>第五章 二维图像稀疏频谱表达实验及应用验证 .....</b>	<b>50</b>
5.1 二维图像稀疏频谱表达算法实验分析.....	50
5.1.1 图像质量评价方法 .....	50
5.1.2 实验及结果分析.....	51
5.2 基于稀疏频谱表达的地质图像频域处理.....	58
5.3 本章小结.....	60
<b>第六章 人脸识别流水线系统的设计与实现 .....</b>	<b>62</b>
6.1 系统需求分析概述.....	62
6.2 系统人脸识别总体结构.....	63
6.3 系统各模块的实现.....	64
6.4 实验分类识别结果.....	70
6.5 本章小结.....	71
<b>总结与展望 .....</b>	<b>72</b>
结论.....	72
展望.....	73
<b>参考文献 .....</b>	<b>72</b>
<b>附录 .....</b>	<b>78</b>
<b>致谢 .....</b>	<b>84</b>

# 图目录

图 1 实数 1 的 8 次复数根.....	9
图 2 卷积散列函数图像.....	15
图 3 算法散列过程.....	16
图 4 组合行列稀疏频谱恢复策略.....	21
图 5 二维 one-sparse 稀疏频谱恢复算法 .....	24
图 6 算法不收敛情形稀疏频谱分布.....	24
图 7 加入行列标识的迭代.....	26
图 8 改进二维严格稀疏频谱表达算法流程.....	28
图 9 不同稀疏度不同采样下各算法对比.....	29
图 10 稀疏度 5000 二维严格稀疏频谱表达算法对比 FFTW 频谱 $mse=0.0011387$	30
图 11 稀疏度 10000 二维严格稀疏频谱表达算法对比 FFTW 频谱 $mse=0.0018217$	
	30
图 12 二维空间图像及其频谱对数图.....	32
图 13 对数频谱幅值统计分布.....	33
图 14 高斯平滑后的图像及其对数频谱.....	34
图 15 高斯平滑后的对数频谱统计分布.....	35
图 16 频域稀疏频率选择函数.....	40
图 17 空间窗口滑动卷积方法.....	41
图 18 程序执行先后顺序.....	43
图 19 算法主要实现类.....	46
图 20 二维稀疏频谱表达算法流程.....	47
图 21 输入图像 lena 及其 FFTW 频谱对数灰度图 .....	51
图 22 稀疏度为 18883 的频谱表达图像 $PSNR=26.8615dB, SSIM=0.9314$ .....	52
图 23 不同稀疏度下稀疏频谱表达效果.....	53
图 24 图像 school 及其 FFTW 频谱对数图 .....	54
图 25 场景图像在不同稀疏度下得表达效果.....	55
图 26 结构相似与差异部分显示.....	56
图 27 停车场场景大图及其 FFTW 频谱对数图 .....	56
图 28 停车场图像稀疏频谱表达对比.....	57
图 29 车牌大图稀疏表达对比.....	58
图 30 地质断层图像及其 FFTW 频谱对数图 .....	59
图 31 稀疏频谱表达地质断层图像.....	59
图 32 地质断层图像稀疏频谱表达断层对比.....	60
图 33 人脸识别系统总体需求.....	62
图 34 人脸识别系统处理流程.....	64

图 35 人脸检测与提取设计实现.....	66
图 36 人脸图像初始化主要设计与实现.....	67
图 37 单层卷积分类.....	68
图 38 多层卷积神经网络.....	68
图 39 卷积神经网络分类器调用类图.....	69
图 40 人脸库 10 类人脸系统分类差误差变化曲线.....	71
图 41 火车站 29 类人脸图像系统分类识别误差.....	71

# 第一章 绪论

## 1.1 研究背景

自从人们步入信息时代以来，随着获取信息的能力的不断加强，新设备新技术的发明创造，信息技术的飞跃发展，对获取信号的存储，表达，分析处理等方面都不断接受新的考验。现今更是一个信息膨胀的时代，DNA 基因组序列，GPS 同步卫星信号，扫描地质图像信号等，这些信号仅规模就比以前的信号庞大很多，而新时代创造了更多的新领域新应用也使得信号处理分析变得更为复杂，这些都呼唤着数字信号处理领域不断创新与进步。

在多种类型的信号中，图像和视频信号更容易被人们所采纳和理解。在计算机上，数字图像是最能够直接表达视觉信息的常用图像形式。其信号表示结构通常是一个按行列排列的二维离散信号矩阵，如果描述的是一个三维物体，其信号结构是一个三维矩阵，但在很多应用场景中，对三维物体图像的分析会截取其中的一些切面来完成。如今，随着信号采集能力的提升，图像信号的大小也不断膨胀，市面上普通智能手机的就可以随时采集 800 万像素的图像，而在一些工业和科研应用中，需要分析和处理的数字图像的规模通常更为庞大。

在信号处理领域中，直接由采集设备采样量化而来的离散信号是信号在时域空间中的表达形式，其遵循奈斯奎特采样定理。相比时域信号表达形式，信号还有多种表达方式。傅里叶变换就是获取信号在频域空间表达的一种方式，其他还有例如拉普拉斯变换，Z 变换，小波变换，多尺度变换等，分别使用不同的空间基来表达时域信号。这些不同的信号表达形式都是等价的，可以通过变换算法在不同的表达形式之间切换。合理的信号表达方式有助于信号的分析和处理，基于傅里叶变换的频谱信号表达就提供了信号在频域的傅里叶分析方法，其使用一组三角函数正交基来逼近表达时域离散信号。由于三角正交基函数连续光滑且周期固定等特性，在时域局部分析和逼近时域间断信号存在缺陷，短时傅里叶变换和小波变换理论很好的弥补了这一缺陷。但在更广阔的应用中，傅里叶频谱分析仍然具有非常重要地位，例如在生理医学中，依赖傅里叶变换来分析 MRI，NMR，超声波图像，其他如地质探索，雷达系统等。快速傅里叶变换算法(FFT)以最低的频域采样定理实现了时域信号和傅里叶频域信号之间的快速变换，但寻求更快的变换算法的研究也从未停止。尤其是在信息膨胀的今天，希望更快的获取信号的频谱，

以及对更简洁的频谱表达方法的需求尤为迫切。

神经生理学的研究<sup>[1]</sup>表明人类视觉对外部刺激具有很强的方向性，而Olshausen和Field的研究<sup>[2]</sup>表明对自然图像的稀疏编码正好具备这样的特点。在此基础上提出了信号基于超完备字典集的稀疏表示，使得对信号稀疏表示的问题成为了研究的热点。在图像领域，基于稀疏表示理论的图像压缩，分类，聚类，人类识别等应用的取得了巨大成功。

稀疏表示理论以远低于奈奎斯特采样频率的方式对信号进行表达，并以很高的概率重建信号。但是基于超完备字典集的信号表示有时并无法满足一些需要频谱分析的特定场合。如果傅里叶频谱也是信号的一种稀疏表示，那么原来很多信号处理中傅里叶频谱分析就可以继续发挥其更大的应用。对大多数自然信号的研究表明，大部分信号傅里叶频谱确实具有稀疏的性质，虽然其稀疏度无法与基于超完备字典的稀疏表示的稀疏度，但相比快速傅里叶变换的频域采样率，其稀疏度已经可以大大简化普通快速傅里叶变换频谱表达信号的复杂度。并且，由于其并没有改变傅里叶变换的性质，基于傅里叶频谱的信号稀疏表达仍然可以无缝的运用到现有的傅里叶分析理论和应用中去。

近年来，MIT对计算傅里叶稀疏频谱的稀疏傅里叶变换算法的研究取得了很多进展，其对一维时域离散信号的稀疏傅里叶变换能够在远低于最优化的快速傅里叶变换算法实现FFTW的时间性能下计算信号稀疏频谱，并在很大领域取得了成功应用。但是，对多维信号是否也可以有多维形式的稀疏傅里叶算法来得到稀疏频谱的研究略显不足，在2013年，MIT的研究提出了一种针对满足特定约束条件的二维离散信号的稀疏傅里叶算法<sup>[3]</sup>。在2014年6月份发表了能够应用于图像的二维稀疏傅里叶算法<sup>[4]</sup>。

## 1.2 国内外研究现状分析

随着计算机性能的发展，处理和分析的信号的规模也变得非常庞大。当计算这些数据的傅里叶频谱时，快速傅里叶变换算法的缺陷也越为明显。也正因为数据规模的增长，基于信号稀疏理论各类研究也变的积极活跃。

近二十年来，稀疏傅里叶算法也取得了很多显著的突破，自1991年Kushilevitz第一次提出了哈码达变换(Hadamard Transform)的亚线性算法<sup>[5]</sup>以来，多个研究复杂输入的亚线性稀疏傅里叶算法也相继发表，其中具有代表性的如：Mansour关于随机插值和近似求解稀疏多项式的算法<sup>[6]</sup>，在理论上给出信号稀疏表示的可行策略，Gilbert等提出的基于信号采样的近理想稀疏傅里叶变换算法<sup>[7]</sup>，能够在满足一定的稀疏度约束条件时，

展示出比 *FFTW* 显著的性能提升，但对于信号必须具有较低稀疏度的要求使之不能广泛被运用。之后 *Gilbert* 等又在此基础上提出了改进运行时间上限的近理想稀疏傅里叶表示算法<sup>[8]</sup>，其运行时间复杂度为  $O(k \log^c(n) \log(n/k))$ ，其中  $c < 2$ ，并且要求稀疏度  $k$  必须是能够小于  $\Theta(n/\log^a n), a > 1$  的整数。对于  $n=2^{22}$  的信号向量，当稀疏度  $k \leq 135$ ，算法的性能优于 *FFTW*。*MIT* 的研究在 2012 年提出的一维稀疏傅里叶算法<sup>[9]</sup>能够在稀疏度  $k < \Theta(n/\log n)$  时保证在  $O((nk)^{1/2} \log^{3/2} n)$  的运行时间内计算得到所有  $k$  个稀疏频谱系数，对于  $n=2^{22}$  的信号向量，该算法在稀疏度  $k \leq 2200$  的区间内性能都优于 *FFTW*。其之后提出的近理想稀疏傅里叶算法<sup>[10]</sup>进一步放宽对稀疏度  $k$  的限制，对严格稀疏的信号，可以保证  $O(k \log n)$  运行时间复杂度，对于普通逼近稀疏的信号，可以保证  $O(k \log n \log(n/k))$  的运行时间复杂度。对于  $n=2^{22}$  的信号向量，在  $k \leq 2^{17}$  区间内算法性能都优于 *FFTW*。

### 1.2.1 稀疏傅里叶算法理论的发展

快速傅里叶算法的缺陷在于对于任何输入信号，因无法判断其频谱的稀疏性，所以认为每个频率系数都是非零的，而对  $N$  长的信号计算其  $N$  个频率，不管计算结果中是否会出现大量幅值为零或近似为零的频率。因为幅值为零表明时域离散信号并不具备该频率分量，而幅值近似为零的频率在很大概率上是噪声信号的频率。在对大多数自然信号的快速傅里叶变换中，这样多余的计算是普遍存在且浪费的。

稀疏傅里叶算法认为，理想无噪声时域离散信号的频谱中只有幅值为零的频率或幅值显著不为零的频率，并称后者为稀疏频率，正是由这些稀疏频率构成了时域离散信号的完整频谱，并能够无损的表达时域离散信号，不会产生时域混叠现象。对于有噪声的时域离散信号，其频谱存在幅值为零或近似为零的频率和幅值显著不为零的频率两种，并认为后者为稀疏频率，这样的稀释频谱可以近似表达时域离散信号。对于噪声较低的音频信号，图像和视频信号，稀疏频谱能够以非常高的概率还原时域信号，并且具备一定的降噪能力。

稀疏傅里叶算法的初衷是希望能够在计算时域离散信号的频谱时，能够甄别稀疏频率与非稀疏频率，从而选择性的只计算稀疏频谱而不去计算这些系数为零或近似为零的频率。因此其计算逻辑也并不会像快速傅里叶算法那样严格遵循离散傅里叶变换公式。但仍然遵循傅里叶变换使用三角函数基表达时域信号的原理。如果频谱只有  $K$  个频率稀疏，那么从数学知识可以得出结论：只需要  $N$  长信号向量的任意  $K$  个维度的值就可以计算出着  $K$  个频率。其原理如同  $K$  个方程可解出  $K$  个未知数一般。正因为如此，目前

主流的稀疏傅里叶算法都是采样时域离散信号来还原稀疏频谱，只有采样的信号参与迭代运算，因此稀疏算法不需要读取时域信号的所有维度值，这样稀疏算法也称为亚线性算法。

目前主流的稀疏傅里叶算法，都采用了相同的思想，可以概括为：首先，使用一个哈希函数对频谱散列，将幅值显著不为零的频率散列到不同的分区中。分区的个数和信号的稀疏度近似，以此来保证均匀散列，以一个较大概率来保证每个分区中只有一个稀疏频率，哈希函数的优劣也直接影响散列的效果，导致散列冲突，多个稀疏频率散列到同一个分区中。散列的过程打乱了原来频谱的顺序，因此需要一个定位函数重新计算每个分区中的频率在频谱中的位置，最后由一个幅值计算函数计算出该频率的幅值。为了保证算法的亚线性，散列函数一般选择为一个在时域和频域中值都比较集中的滤波函数，例如高斯函数，其傅里叶变换仍然是一个高斯函数，其非零值集中在均值附近。根据不同的哈希函数，可以将现有的稀疏傅里叶算法大致分为两类：基于迭代的算法和基于插值的算法。基于迭代的算法选择频域下类似矩形函数作为其散列函数，其在时域中表现为一个狄利克雷核，其缺点在于该频域函数尾部收敛截断，会导致散列时某些系数散列到其他已有系数的分区中，为避免此情形而迭代多次；基于插值的算法并不常用，该方法是将离散信号作为一个连续函数，因此需要在没有值的地方进行插值。

MIT 的两个算法使用高斯函数与矩形函数的卷积作为散列函数，弥补基于迭代算法的缺陷，使整个算法的结构大为精简。

另外，由于稀疏理论的发展，另一种基于线性编码理论求解信号稀疏表示的方法也被提出，在该方法中，如果用傅里叶基构成一个基向量字典集，最后求解出的信号稀疏表示即为稀疏傅里叶频谱。

以上研究都是针对一维稀疏傅里叶变换展开的研究，对于二维或更高维的稀疏傅里叶变换算法的研究还比较少，二维信号序列如图像，由于其在空间上规则排列，其频谱稀疏的特性也非常明显。MIT 研究人员 *Badih Ghazi* 等人在 2013 年提出了一种针对稀疏频谱服从伯努利分布的二维稀疏傅里叶算法<sup>[3]</sup>，对满足条件二维信号序列的能够较快的求得其稀疏频谱，但在实际应用中，很少二维信号能满足此条件，但该算法中很多思想和构成了本文的理论基础，本文对二维图像信号序列，探索了一种简单高效的傅里叶稀疏频谱表达方法。

### 1.2.2 稀疏傅里叶算法应用的发展

稀疏表示信号理论的发展和稀疏傅里叶算法的突破，也促进了信号处理中很多相关的其他领域的进步。在认知无线电领域，*Haitham Hassanieh* 等发明了利用稀疏傅里叶算法实现在较低的奈斯奎特采样率下对  $GHz$  宽频信号实时频谱感知与解码的 *BigBand* 技术，该技术极大提高动态频谱接入能力，使感知用户能够探测到极短的亚微秒频谱缺失并弥补，从而提高频谱的整体利用率。另外，较低的采样率使得政府或工业界可以在城市中部署更多的感知器用于大规模实时频谱监控，这将有助于更好的了解频谱的利用，以及更灵活的频谱分配策略<sup>[11]</sup>。

全球定位系统(*GPS*)是一种广泛使用的无线技术，在世界范围内用于大部分智能手机和便携导航设备中。定位的关键技术在于锁定 *GPS* 卫星信号，并解码卫星轨迹和时间数据。解码的任务通常由 *GPS* 基站来完成，所以现在便携设备主要完成对卫星信号的锁定，通常的做法是对每个卫星的 *CDMA* 编码和设备接收的卫星信号做卷积来计算移位相关，在时域中卷积的时间复杂度是  $O(n^2)$ ，所以现在工业界的做法是通过计算频域的乘积达到计算卷积目的，时间复杂度将为  $O(n \log n)$ ，但即使如此，对于计算能力有限的终端便携设备也是巨大的计算成本。最新的研究采用了稀疏频谱恢复和稀疏傅里叶算法，利用最终信号卷积稀疏的特性，对频谱乘积的结果采用稀疏傅里叶逆变换算法得到时域卷积，同时，由于稀疏傅里叶变换算法为亚线性算法，所以只需对采样引号计算频谱乘积即可，这一过程大大降低了计算复杂<sup>[12]</sup>

*Lixin Shi* 在处理人脑的核磁共振医学图像时，使用稀疏傅里叶表示算法降低了频谱获取时间，同时减少了二维相关谱的截断光谱<sup>[13]</sup>；

随着信息的不断膨胀，稀疏傅里叶算法理论的完善，以及信号稀疏表示理论的深入研究，信号处理领域依赖傅里叶频谱分析的应用中将越来越多的出现稀疏傅里叶算法的身影，发挥其对信号处理领域的巨大贡献。

## 1.3 论文的主要研究目标及内容

### 1.3.1 研究目标

本文基于信号处理基本的理论，以及离散傅里叶变换基础，结合最近几年稀疏傅里叶变换算法取得的重大突破，寻找一种适用于二维图像信号的稀疏傅里叶变换方法，希望通过该方法，能够简洁高效的计算得到二维图像的稀疏频谱表达，而不需要计算复杂的二维离散傅里叶变换。并且，通过实验验证稀疏频谱表达的正确性，并尝试在一些数

字图像处理应用中使用这种简单的信号表达方法。

### 1.3.2 研究内容

为了能够实现正确的图像稀疏频谱表达，本文研究了最近几年稀疏傅里叶算法上最新的研究，并结合信号表示理论的一些基础知识，改进了现有 MIT 发表的二维稀疏傅里叶算法收敛条件，并基于此算法策略，针对图像频谱的性质，寻找了一种适合于二维图像稀疏频谱表达的计算方法，并通过大量使用验证表达的正确性，具体内容简要概况如下：

(1). 总结了信号稀疏表示理论基础知识，并研究了一种基于线性编码理论的信号稀疏表示方法，和一种优化的一维稀疏傅里叶算法。总结了其中一些信号表示原理和算法思想。

(2). 重点研究了现有二维严格稀疏傅里叶变换算法，分析了其对适用信号的约束条件，算法失败情况，在此基础上提出一种改进后的二维严格稀疏傅里叶变换算法，其收敛情形和适用条件都得到很大改善。并对频谱严格稀疏随机信号验证了算法稀疏频谱表达的正确性和算法的性能。

(3). 研究了二维图形信号的傅里叶频谱分布特点，在二维严格稀疏傅里叶算法的基础上，针对图形稀疏频谱恢复中存在的图像信号包含噪声以及频谱分布接近高斯分布两个二维严格稀疏算法无法解决的难点，提出了两个可行的解决方案，并重新改进了算法实现，提出了一种用于二维图像稀疏频谱表达的方法，并通过实验验证其对二维空间图像的表达能力。

(4). 研究了二维图像稀疏频谱表达方法在三维地质图像断层切面频域分析中的运用。最后，在人脸识别系统中，稀疏频谱表达方法作为一种基于频域的主成分分析方法，降维频域的同时，保留的空间图像的主要频域信息。本文研究了其作为频域特征用于图像的分类和识别，并结合空间图像来提高和改善识别的效率。

## 1.4 论文的组织结构

本文主要由以下部分构成：

第一章：绪论部分主要对本文的研究背景意义，目前国内外的研究成果，发展趋势以及对工业界的巨大影响做了概述，并介绍了本文的主要研究内容。

第二章：主要介绍了信号稀疏表示理论基础，以及本文相关技术分析。主要分析了

一种基于编码理论的稀疏信号表达方法，以及一种简单高效的稀疏傅里叶算法。其作为本文的关键理论和技术基础，为本文研究提高了坚实的支撑和技术指导。

第三章：介绍最近由 MIT 发表的一种二维严格稀疏傅里叶变换算法，重点研究了该算法的主要算法策略，分析该算法的收敛条件与适用范围。在此基础上提出一种改进的二维严格稀疏傅里叶算法，使其适用于更高的稀疏度。对符合条件的二维信号试验其计算稀疏频谱表达的性能和正确性。

第四章：介绍二维图像频谱的一些特点，与计算稀疏频谱表达索要解决的关键问题。结合第三章二维稀疏傅里叶算法的稀疏频谱计算策略，提出一种针对二维图像计算其稀疏频谱表达方法，给出具体的设计与实现，并分析可能的时间复杂度。

第五章：对不同类型图像的尝试使用第四章提出的计算稀疏频谱表达方法，验证图像表达的准确性。并简要介绍了其目前在一些图像处理中得应用。

第六章：简要介绍了本文应用项目人脸识别流水线系统核心识别模块的设计和实现，本文研究内容主要服务于该系统的一个主要模块。

## 第二章 信号稀疏表示理论及关键技术分析

信号稀疏表示是一种新的信号表达方法，目前已在很多信号处理应用中发挥了很大的作用。对信号稀疏表示的研究也是最近的热门。傅立叶变换作为另一种信号表示方法，并且其频谱也具有一定的稀疏性。因此两者之间在思想和计算稀疏表达方法都一些近似之处。因此求解稀疏傅立叶变换也可以借鉴一些信号稀疏表示的理论和方法。

### 2.1 信号稀疏表示理论概述

近年来，稀疏已经成为信号处理及其应用领域处于第一位的概念之一。在香农的压缩传感理论中<sup>[14]</sup>，首次正式提出了信号本身具有稀疏的特性。在信号处理的很多领域，都希望得到一种简洁而不失真的信号表达方法，而在过去的发展中，信号都是在基于正交基的表达方法上建立的。而实际应用中，大多数信号是很多种类型的自然现象的叠加，不同类型的自然现象并不局限于能够使用一组正交基合理而正确的表达。因此叠加后的混合信号也很难在单一的正交基上寻找到简洁合理的表达，唯一能做的就是用更高频的向量无限逼近。举例来说，对于一个由脉冲信号和正弦信号叠加而来的信号，其基于正弦函数基或基于脉冲函数基的表达方法都无法很好的体现信号的全部特征，其中，基于三角函数正交基的傅里叶变换的结果是无限维的，离散傅立叶变换按最低的频域采样率来得到与输入信号相同维度的频谱表达。而信号稀疏表示并不局限于使用这样一组基，事实上其弱化了空间和基的概念，致力于更简洁的信号表示，因此其只需要用一个脉冲信号和一个正弦信号无损表达了输入信号。

在图像处理领域，对于二维离散信号的处理，基于稀疏表示的图像识别，去噪，分类算法研究也都取得了很大的突破。

信号稀疏表示理论的基本思想是：所有的自然信号都是稀疏的，即使在当前空间表示中不稀疏，也可以通过变换到其他空间中体现其稀疏性，即：

$$r = \sum_i c_i \phi_i \quad \phi_i \in F \quad (2.1)$$

其中， $r$  是  $n$  维欧几里得空间中得一个  $n$  维信号向量， $r=(r_0, r_1, \dots, r_{n-1})$  就是其在当前空间中得表达形式，如果当前表示并非稀疏，那么也可以找到另一个空间  $F$ ， $\phi_i$  是该空间的一组基，当前空间信号  $r$  可以由  $F$  中得基向量线性表达， $r=(c_0, c_1, \dots, c_{n-1})$  就是  $r$  在空间  $F$  中得表达形式，公式(2.1)即定义了  $r$  在两个空间中得变换过程。在空间  $F$  中， $r$  或许是稀疏的，即  $(c_0, c_1, \dots, c_{n-1})$  中只有少数几个维度的系数不为零，也即  $r$  可以由空间  $F$

的少数几个基向量的线性组合来逼近表达;

傅里叶变换也是这样一种空间变换, 其空间基定义为一组三角函数正交基, 并具有周期性质, 一维离散傅里叶变换公式可以写为:

$$\hat{x}_k = \sum_{j=0}^{n-1} x_j \omega_n^{jk} \quad (2.2)$$

式中  $x$  是时域信号,  $\hat{x}$  是傅里叶空间变换后的频域信号表示,  $\omega_n^k$  就是傅里叶频谱空间的基向量, 是一个周期函数, 其值为一个以  $e$  为底的复指数周期信号, 可以通过欧拉公式转换三角函数的组合, 如下式(2.3)所示:

$$\omega_n^k = e^{2\pi k i / n} = \cos\left(\frac{2\pi k}{n}\right) + i \sin\left(\frac{2\pi k}{n}\right) \quad (2.3)$$

从上面公式也可以发现  $\omega_n^k$  是 1 得一个复数  $n$  次方根, 并且根据其三角周期函数组合, 可以计算其周期  $T=n$ , 以  $n=8$  为例, 其在一个周期内的值可以表示为:

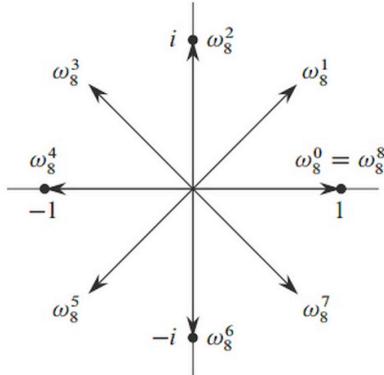


图 1 实数 1 的 8 次复数根

因此  $\omega_n^{kj}$  可以看出是周期函数  $\omega_n^{kx}$  的离散序列, 其频率为  $k/n$ 。傅里叶变换也就是使用这样一组周期函数信号来逼近空间信号, 来达到一种等价表达。

式(2.2)定义了一组由  $n$  个方程组成的方程组, 如果将其变换为矩阵形式, 并以列向量为基, 就可以得到如(2.1)式所示的形式。以下将对其逆变换展开为矩阵变换形式, 并解释其空间变换过程:

$$x_j = \sum_{k=1}^{n-1} \hat{x}_k \bar{\omega}_n^{kj} \quad (2.4)$$

式(2.4)是傅里叶逆变换公式, 将其写成矩阵形式为:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} \bar{\omega}_n^{0*0} & \bar{\omega}_n^{0*1} & \cdots & \bar{\omega}_n^{0*(n-1)} \\ \bar{\omega}_n^{1*0} & \bar{\omega}_n^{1*1} & \cdots & \bar{\omega}_n^{1*(n-1)} \\ \vdots & \vdots & \cdots & \vdots \\ \bar{\omega}_n^{(n-1)*0} & \bar{\omega}_n^{(n-1)*1} & \cdots & \bar{\omega}_n^{(n-1)(n-1)} \end{bmatrix} \bullet \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \vdots \\ \hat{x}_{n-1} \end{bmatrix} \quad (2.5)$$

如果写成矩阵变换形式：

$$x = A\hat{x} \quad (2.6)$$

若记  $\phi_i$  为矩阵  $A$  的第  $i$  列，式(2.6)即可写成式(2.1)的形式，即：

$$x = \sum_i \phi_i \hat{x}_i \quad (2.7)$$

至此，式(2.7)表明时域信号  $x$  可以由傅里叶空间中的一组基向量线性组合表达。

傅里叶变换后的空间通常称为频域空间，相应的原来信号的空间称为时域。上式表明了时域信号可以有频域的一组信号线性组合表达。通过傅里叶变换，不仅使信号的某些处理分析在频域变得简单可行，同时，频域信号相比时域信号通常具有更为稀疏的特性，这将极大的减少信号处理的输入规模，使原本计算成本极高的任务变得可行。

信号稀疏表示理论研究的重点问题，在于如何最快速的计算得到一个原本非稀疏信号在某个空间的最稀疏表示，而稀疏傅里叶算法主要研究的是如何最快速的计算得到非稀疏时域信号在频域中的稀疏频谱。

仍设  $r$  是  $n$  维欧几里得空间  $R$  中得一个向量， $F=\{\phi_0, \phi_1, \dots, \phi_{m-1}\}$  是  $n$  维空间  $W$  中一组向量的集合，并且  $m>n$ ,  $F$  的秩为  $n$ ，即  $F$  中任意  $i< n$  个向量都是线性无关的，因此称  $F$  为  $n$  维空间  $R$  的一个过完备字典集， $R$  中任意一个向量都可由  $F$  中一组元素线性组合表示，如公式(2.1)所示，并且，由于字典集是过完备的，这样的表示并不唯一，也许会存在多种表示方式，但唯一存在一个最稀疏表示，即： $\|r\|_{0,F}$ ，表征  $r$  基于过完备字典集  $F$  的线性组合表示中稀疏非零项的个数最少。设  $c=(c_0, c_1, \dots, c_{m-1})$  是  $m$  维向量空间中的任意向量，并且  $\|c\|_0$  表示向量  $c$  中非零项的个数。信号稀疏表示理论研究的关键问题，就是能够最快的寻找到其中的一种表示方式，使得向量  $c$  能够满足：

$$\min_{c \in C^m} \|c\|_0 \quad s.t. \quad r = cF \quad (2.8)$$

上式即希望找到向量  $r$  在过完备字典  $F$  中一个线性表示或逼近，使得该表示中系数的非零项最小。但仅需这几个非零大系数及其对应的字典集元素，即可很好的展现出信号的本质特征；这就是信号稀疏表示理论中得最紧凑表达问题(*MCR*)<sup>[15]</sup>，如果强制要求过完备字典是由傅里叶频谱空间的向量构成的，那么问题就转换为傅里叶稀疏表达问题。虽

然这样做构造较好的过完备字典集来使信号的表达更为稀疏，但基于傅里叶空间基的表达保持了傅里叶频谱的良好性质，很多傅里叶变换，小波变换的频谱分析方法仍然可以运用。

## 2.2 基于线性编码理论的信号稀疏表示方法分析

上述 *MCR* 问题中引入了一个  $l_0$  范数，定义为向量  $\mathbf{c}$  中非零项的个数，在最小化非零项个数的同时必须考虑到信号表达的误差，是一个很难解决的优化问题，而事实上这也是一个 *NP-Hard* 问题，并没有很好的解决办法；很多研究者转而研究其相似问题如对  $l_1$  范数：

$$\min \|\mathbf{c}\|_1 = \sum_{i=0}^{m-1} |c_i| \quad (2.9)$$

的求解，并通过一个判断规则来求解 *MCR* 问题，这正是 *BP(Basis Pursuit)*<sup>[16]</sup> 的做法。目前 *MCR* 问题比较常用的算法有 *BP(Basic Pursuit)* 和 *OMP(orthogonal matching pursuit)*<sup>[17]</sup>。其中，正交基匹配(*OMP*)是一个贪心算法，其在一开始初始化一个残差向量  $\mathbf{r}$  为需要求解的原始信号，然后每一次在过完备字典中寻找到一个元素，该与残差向量有最大得相关性，并用残差向量在该字典元素上得投影来更新当前残差信号，直到算法收敛，所以也称作投影追踪算法。基追踪(*BP*)算法运用凸优化的方法来寻找信号向量在过完备字典中得稀疏表示，求解最小化  $l_1$  范数，其先初始化一个由  $n$  个线性无关向量按列组成的基矩阵，并在之后的每次替换中，使用字典中得一个向量来替换基矩阵中得一列，使得目标解得到改善来不断优化表达效果。

*BP* 和 *OMP* 算法都可以运用到任意形式的过完备字典上，求解稀疏表示。但是这两类算法都没有仔细考虑过是否可以构造一个有效的过完备字典，使其能够支持简单高效的稀疏分解。

与 *BP* 和 *OMP* 求解 *MCR* 问题的方法不同，这里考虑将 *MCR* 问题的求解与错误纠正编解码理论联系起来寻求一种新的算法。在这以前，已有一些研究人员实现了过完备字典集与基于复数域的线性编码理论的联系。正是被这样的联系与代数编解码理论启发，新算法使用范德蒙矩阵来构造过完备字典集，该字典集可以看做是里德所罗门编码在复数域的泛化。如果设  $\|\mathbf{r}\|_{0,F} \leq n/2$ ，即  $n$  维向量  $\mathbf{r}$  基于过完备字典集  $F$  的最稀疏表示中系数非零项的个数小于等于  $n/2$ ，那么 *MCR* 问题可以表示为一个复数域的线性解码问

题，并通过泛化的复数里德所罗门解码<sup>[18,19]</sup>算法求解，该解码算法在线性编解码领域是一个广为人知的算法理论。

线性编码也称线性错误控制编码<sup>[20]</sup>，以较简单的汉明编码为例，其码表  $C$  中每个编码为一个长为  $n$  的二进制数，也称为一个密语(*codeword*)。每个密语中非零位的个数称为该密语的权重。而之所以称为线性编码，是因为码表中任意两个密语的按位相加得到二进制数也必定是码表中的一个密语。两个密语中不相同位的个数称为这两个密语的距离，也可以看做是两个密语按位相减后的差中非零位的个数，也即差的权重。整个码表中距离最小的两个密语的距离称为码表的最小间距。对于线性编码，码表的最小间距  $d(C)$  为其中权重最小的密语的权重。

网络中得信号在传递前先被编码，接收方根据相同的码表解码。信号不能被直接传递的原因在于传递过程中可能会受一些原因的影响而发生传输错误，原始信号如果没有做任何编码处理，抗干扰能力非常差，接收方无法知道错误发生在哪，往往轻微的错误就导致整个信号无法使用。编码的初衷就是为了提高信号在传输过程中得抗干扰能力，编码后的信号可以根据码表本身的一些性质和附加的一些校验位等，在信号传输发生错误时，能够定位错误发生的位置，并无损的解码恢复正确信号。

研究接收信号编码中可能的错误个数等价与研究接收信号编码和发送信号编码中间的距离。例如，假设  $c=c_0c_1\dots c_{n-1}$  为发送信号编码， $r=r_0r_1\dots r_{n-1}$  为接收信号编码，错误信号就可以表示为  $e=r-c=e_0e_1\dots e_{n-1}$ ，解码算法的目标，就是找到最可能的发送信号编码，或者最可能得错误信号。很多解码算法采用最邻近解码策略，从码表中选择一个密语作为可能的发送信号编码，并使错误信号  $e$  的权重最小，这个策略假设具有较小权重的错误信号比具有较高权重的错误信号更可能出现。

如果编码信号的密语发生传输错误，从一个密语变为码表中的另一个密语，那么普通的解码策略是无法发现错误的，此时  $e$  的权重至少为  $d(C)$ ，对于权重大于等于  $d(C)$  的错误信号，除非依靠校验，普通解码策略无能为力。因此，具有较大的  $d(C)$  的码表具有更好的抗干扰能力，也称最大间距编码，遵循以下两个定理<sup>[20,21]</sup>：

1. 一种编码能够检测任何密语中最多  $s$  个错误，前提是  $d(C) \geq s+1$ 。
2. 一种编码能够纠正任何密语中最多  $t$  个错误，前提是  $d(C) \geq 2t+1$ 。

里德所罗门编码就是这样一种最大间距编码方法，与汉明码不同的是，它是一种非二进制循环线性编码，但同样遵循上面两个定理。汉明码中，密语的每一位取值为 0 或 1，而在里德所罗门编码中每一位的值取自  $GF(2^m)$  伽瓦罗域，是一个有  $m$  个二进制位构成

的值。其解码为伯利坎普-梅西算法(*Berlekamp-Messey, BM*)算法<sup>[18,20,21]</sup>, 通过构造一个错误定位多项式<sup>[18,20,21]</sup>, 并求解多项式的根得到错误信号的位置。

设  $F=\{\phi_0, \phi_1, \dots, \phi_{m-1}\}$  是覆盖  $n$  维空间  $W$  的一个过完备字典, 设:

$$V = \left\{ d = (d_0, d_1, \dots, d_{m-1}) \in C^m : \sum_{i=0}^{m-1} d_i \phi_i = 0 \right\} \quad (2.10)$$

$V$  是一个向量空间, 其中每一个向量长度为  $m$ , 并且与过完备字典的乘积为 0; 由于过完备字典  $F$  的秩为  $n$ , 因此  $F$  中任意小于等于  $n$  个向量都是线性无关的, 由此可以计算向量空间  $V$  的维度为  $m-n$ ; 由上所述,  $n$  维空间  $W$  中得任意向量都可以过完备字典中得元素线性组合表示,  $c=(c_0, c_1, \dots, c_{m-1})$  即为该表示的系数向量, 因此, 向量  $c$  也可以认为是如:

$$c - V = \{c - d \mid d \in V\} \quad (2.11)$$

即向量  $c$  可以表示成自己与空间  $V$  中任意一个向量的差, 其证明如下:

$$\begin{aligned} r &= \sum_{i=0}^{m-1} c_i \phi_i \\ &= \sum_{i=0}^{m-1} c_i \phi_i - \sum_{i=0}^{m-1} d_i \phi_i = \sum_{i=0}^{m-1} (c_i - d_i) \phi_i \end{aligned} \quad (2.12)$$

因此, *MCR* 问题就从求解向量  $c$  的最小  $l_0$  范数变为求解向量  $e=c-d$  的最小  $l_0$  范数  $\|c-d\|_0$ ;

如果把向量空间  $V$  中得向量看做是由  $m$  长复数构成的一组线性编码, 系数向量  $c$  看做接收到的信号编码  $r$ , 并且  $r$  中存在错误, 正确的编码密语是向量空间中的向量  $d=d_0d_1\dots d_{n-1}$ , 那么上述求解向量  $e$  的最小  $l_0$  范数就变为了求解权重最小的错误信号, 并使  $d=c-e$  为正确的编码密语。这就转换为了一个解码过程, 可以用相应的解码算法来完成。在 *Mehmet Akcakaya* 的文章<sup>[21]</sup>中详细描述了如何使用范德蒙矩阵构造字典集  $F$ , 使向量空间  $V$  称为一个基于复数域的最大间距里德所罗门编码, 通过 *BM* 算法等解码算法解得错误信号  $e$ , 也就是向量  $r$  在字典集  $F$  中得稀疏表达。

如果使用傅里叶频域向量构造字典集  $F$ , 那么最后得到的向量表达就是稀疏傅里叶频谱表达, 在 *MIT* 对稀疏傅里叶算法的研究中, 为了计算稀疏傅里叶频谱, 借鉴了该方法, 再使用 *BM* 算法解得稀疏频率的位置后, 采用一个矩阵乘法公式计算最后的傅里叶稀疏频谱, 该公式是式(2.5)的变化:

$$\hat{x} = (0, 0, \dots, \hat{x}_{k_1}, \hat{x}_{k_2}, \dots, \hat{x}_{k_t}, \dots, 0)$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{t-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \bar{\omega}_n^{k_1} & \bar{\omega}_n^{k_2} & \cdots & \bar{\omega}_n^{k_t} \\ \bar{\omega}_n^{2k_1} & \bar{\omega}_n^{2k_2} & \cdots & \bar{\omega}_n^{2k_t} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\omega}_n^{(t-1)k_1} & \bar{\omega}_n^{(t-1)k_2} & \cdots & \bar{\omega}_n^{(t-1)k_t} \end{bmatrix} \begin{bmatrix} \hat{x}_{k_1} \\ \hat{x}_{k_2} \\ \vdots \\ \hat{x}_{k_{t-1}} \\ \hat{x}_{k_t} \end{bmatrix} \quad (2.13)$$

## 2.3 基于频域哈希的一维稀疏傅里叶算法分析

正如绪论中对稀疏傅里叶算法发展的介绍中提到的，除了上述基于编码理论的稀疏频谱恢复算法外，更为实用且高效的是基于时域采样并散列频谱的算法，该算法针对一维信号序列计算稀疏频谱，主要思想是通过一种哈希方法将频谱散列到不同的分区中，并在每个分区中单独计算散列到该分区中得频谱，本节主要分析一种最近由 MIT 提出的优化后算法，该算法颠覆了之前算法同类算法不是基于迭代或就是基于插值的分类<sup>[9]</sup>，并且在性能上有了很大提升。本文在研究二维频谱表达算法时借鉴了其中很多思想和方法。

算法最大的创新点在于采用了一种新的哈希策略，与以往基于迭代的同类算法不同的是，每次求得频谱不需要再反馈输入信号来抵消已恢复频谱对输入信号的影响，并更新下一次迭代的输入。哈希方法基于两个方法，随机重排频谱与特殊卷积散列函数。

首先，为了避免散列冲突，不同稀疏频谱散列到同一个分区中，设定散列分区的个数与稀疏频率的个数相同，以此来降低每次散列时发生冲突的概率。但这样做并不能保证冲突不发生，因此，第二重保障对频谱进行多次哈希，每次哈希前随机重排频谱，使每次散列的结果不同，重排结果有一个输入的随机因子决定，可以认为每次重排的结果都是不同的。事实上，算法的目的在于恢复稀疏频谱，所以在算法过程是无法知道频谱详细信息的，因此虽说是重排频谱，也必须想办法通过别的手段使之在不操作频域的前提下完成对频域的操作，这个想法是可行的，由于时域和频域是等价的，频域中得很多计算可以在时域中找到等价的计算方法，例如频域乘积等价与时域卷积。这一点也是本文中经常会使用到的方法。为了达到重排频谱的目的，需要定义如下变换  $P_{\sigma,\tau}$ <sup>[9]</sup>，使得：

$$(P_{\sigma,\tau}x)_i = x_{\sigma i + \tau} \quad (2.14)$$

其中， $x$  是  $n$  维向量，并且  $x$  的下标对应其对  $n$  取余后才是其真正的位置， $\tau$  是小于  $n$

的整数，因此就有：

$$\begin{aligned}
 \widehat{(P_{\sigma,\tau}x)}_{\sigma_i} &= \widehat{(P_{\sigma,\tau}x)}_a = \sum_{j=0}^{n-1} x_{\sigma j + \tau} \omega^{aj} \\
 &= \sum_{j=0}^{n-1} x_j \omega^{a(j-\tau)\sigma^{-1}} \\
 &= \hat{x}_{a\sigma^{-1}} \omega^{-\tau a \sigma^{-1}} = \hat{x}_i \omega^{-\tau i}
 \end{aligned} \tag{2.15}$$

上式定义了重排时域信号和重排频域的映射，通过这个变换，可以保证每次重排后散列到同一分区的频谱不同，因此，本次发生散列冲突的频率在下一次散列时必定在两个不同分区中。

第二个关键技术在于特殊的卷积散列函数<sup>[9,10]</sup>，优化算法中使用的散列函数相比其他算法中得散列函数具有更好的表现。

卷积散列函数的目的是能够对频域进行自动的筛选，将稀疏频率归类到不同的分区中，不同于传统的哈希方法将待哈希对象映射到一个索引的做法，首先一点是对于时域信号，其频谱是未知的，哈希对象对哈希方法是透明的。另外，并不存在像传统哈希方法可以有公式定义的映射关系。卷积散列函数是对时域信号操作来间接散列频谱的一个自适应的哈希方法，其表现更像一种过滤函数，只让满足条件的频率进入分区。

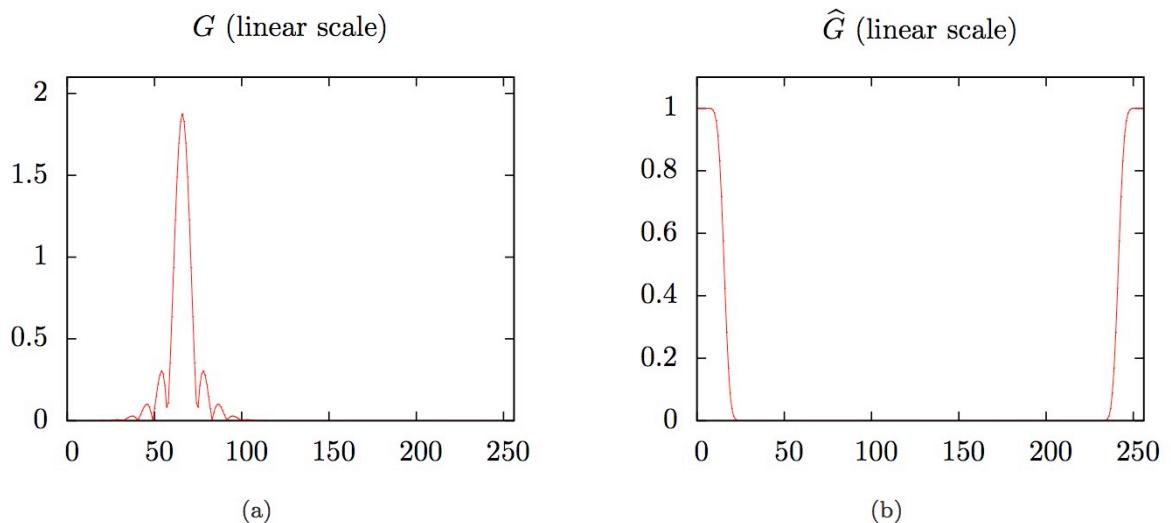


图 2 卷积散列函数图像

本节阐述的算法中卷积散列函数的独特之处在于，函数具有较为平缓的通过域，并且，通过域尾部以一个指数级的光滑曲线下降为零，因此能很好的避免稀疏频率散列是被遗漏。很多其他算法使用类似矩形函数作为散列函数，其结果就是处于通过域边缘的频率被遗漏，而不得不通过多次迭代来补救。本节卷积散列函数的函数图像如图 2 所示，

左边为时域内的函数  $G$ , 右边为频域内的函数  $G'$ 。

使用函数  $G'$  作为哈希函数的原理是: 根据卷积的性质, 稀疏频率与卷积选择函数  $G'$  的卷积将在该频率附近形成一个平坦区域, 认为该频率被散列到了这个平坦区域构成的分区。详细的算法过程如下图所示:

首先, 图 3a 中所示表示最终的频谱会有 4 个稀疏值, 并且经过时域重排后的频谱分布, 在图 3b 中, 将频谱和过滤函数  $G'$  进行卷积, 而时域中相应的操作只需计算重排的时域信号与过滤函数的乘积, 乘积后相应的频谱分布由图 3b 中得虚线所示, 可以看到, 右边两个稀疏频谱很好的散列到了两个不同的分区中。

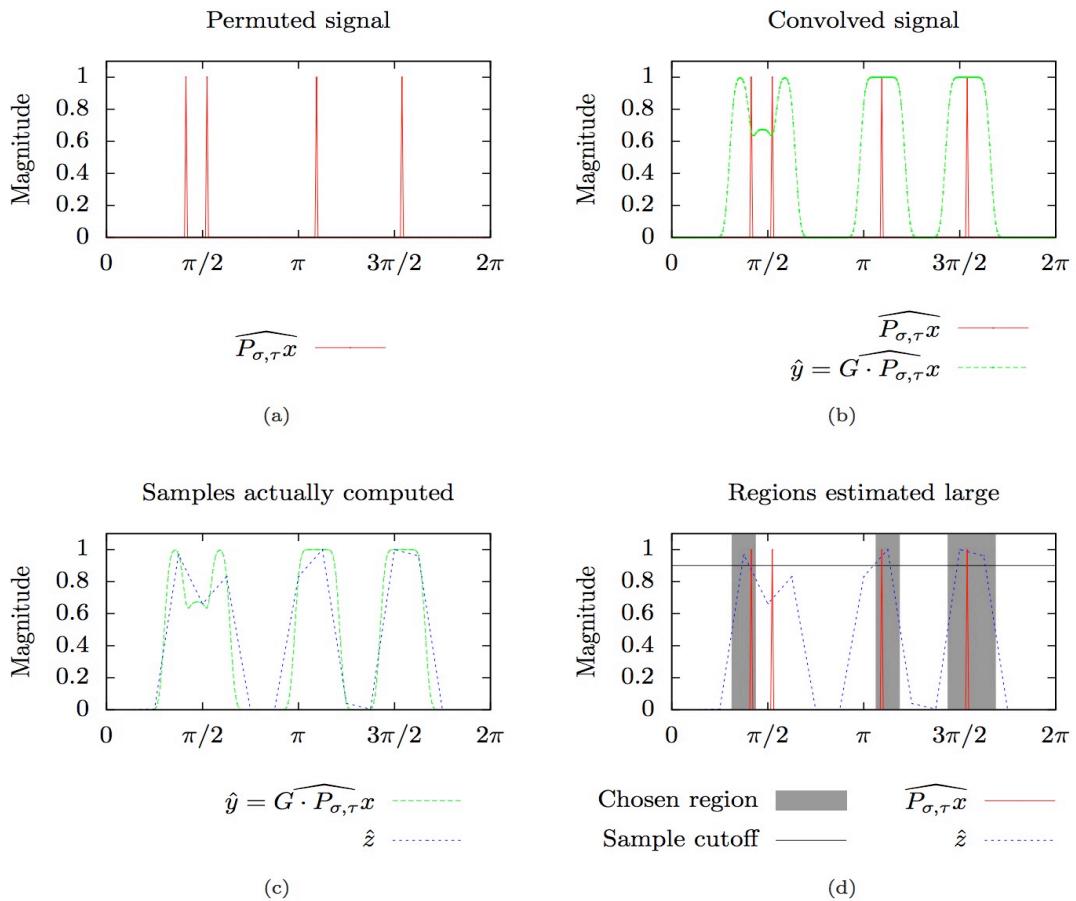


图 3 算法散列过程

图 3c 中, 对卷积后的频谱等间距采样, 如上文所述, 这里必须找到一个采样频谱等价的时域计算方法, 通过操作时域来达到采样频谱的目的, 因此, 有以下推导, 仍然设  $x$  使一个  $n$  维向量, 下标对  $n$  取余, 设  $B$  为可被  $n$  整除的一个整数,  $i \in [0, B]$  并且有:

$$y_i = \sum_{j=0}^{n/B-1} x_{i+Bj} \quad (2.16)$$

上式中,  $y$  是一个  $B$  维向量,  $y_i$  是  $x$  向量中所有下标对  $B$  取余等于  $i$  的分量的累加, 对  $B$

维向量  $y$  的傅里叶变换如下所示：

$$\begin{aligned}
 \hat{y}_i &= \sum_{a=0}^{B-1} y_a \omega^{ian/B} \\
 &= \sum_{a=0}^{B-1} \sum_{j=0}^{n/B-1} x_{Bj+a} \cdot \omega^{ian/B} \\
 &= \sum_{a=0}^{B-1} \sum_{j=0}^{n/B-1} x_{Bj+a} \omega^{i(Bj+a)n/B} \\
 &= \sum_{j=0}^{n-1} x_j \omega^{ij(n/B)} = \hat{x}_{i(n/B)}
 \end{aligned} \tag{2.17}$$

上述推导中涉及周期信号  $\omega$  的一些性质，可自行查阅验证推导过程。式(2.17)证明了对频谱的等间距采样，等价于在时域中按相同的间距折叠时域信号。因此只需计算时域中同间距折叠信号的傅里叶变换即可达到频域采样的目的，折叠后信号的长度远远小于原来信号，所以计算其傅里叶变换消耗的计算资源也较少。图3c中，采样了16个频域值，在图d中，选择其中最大的四个采样值，记录其幅值与其临近频谱，图中阴影部分频率的坐标。但此时还不能确定是哪一个坐标相应的频谱稀疏，通过多次这样的过程，可以得到和幅值相应的一个可能的坐标集合，在这个坐标集合中出现次数越多，越有可能是真正的稀疏频率下标，最后，对出现次数满足一定条件的坐标，通过反向卷积重排过程，求出他们各自真正的幅值，虽然求解结果中只会有一个是稀疏频谱，其他非稀疏频率的求解出得幅值都将是一个为零或近似为零的值，但是由于满足条件的坐标数较少，这样的计算并不会消耗很多计算资源，但提高了算法的准确性。这一点也是本文在尝试对二维频谱恢复时借鉴的思想。

## 2.4 本章小结

本章介绍了信号稀疏表示理论的简要内容，引出了稀疏傅里叶变换的本质及其与稀疏表示理论之间的联系。然后介绍了两种可行的稀疏算法，首先算法是一个求解信号稀疏表示的算法，但由于稀疏傅里叶变换和稀疏表示理论的联系，其也可被用于稀疏傅里叶频谱的求解。另一个直接从傅里叶变换频谱的稀疏性质出发，通过巧妙的散列方法恢复稀疏频谱。本章内容主要介绍理论内容和算法原理，其构成了本文的理论基础，也是本文对二维信号稀疏频谱研究的基础。

## 第三章 二维严格稀疏频谱表达算法及其改进

不同于一维离散时序信号，二维空间信号的频谱稀疏性折叠到了更高的维度中，因此直接分离维度套用一维时信号稀疏频谱表达的方法会显得粗糙低效。多维信号的频谱往往可以从多个维度方向来看待其稀疏性，这也许提供了相比一维时序信号更多的稀疏频谱计算方法。例如，对于二维离散信号频谱中的每一个稀疏频率，都可以看作是一行或一列的稀疏频率，因此，至少有两种方法可以来计算给频率，这也使得问题变得简单。

### 3.1 二维稀疏表达算法研究问题概述

上一章中介绍了两种信号稀疏表达算法，其中一种基于过完备字典集的稀疏表示，另一个是基于傅里叶频谱的稀疏表示。这两种算法都能可以用于准确的获取一维信号序列的稀疏频谱，尤其是第二种算法的实现 *SFFT* 已成功应用到工业界的很多信号处理领域，取得了重大成果，如第一章绪论中介绍了 *GPS* 定位技术等。但是，像数字图像这样的二维甚至三维信号序列，上面两种算法并不适用。事实上多维信号因其在更高的维度上的有序性，其频谱相比一维信号频谱更为稀疏。以二维离散信号为例，以下二维离散傅里叶变换的公式：

$$\hat{x}_{u,v} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x_{i,j} \omega_n^{ui} \omega_m^{vj} \quad (3.1)$$

二维离散傅里叶变换可以看做是按行方向对信号做一维傅里叶变换后，在变换后序列上再按列方向的又一次一维傅里叶变换，如果第一次一维傅里叶变换后，其得到的频谱暂时称为中间频谱，和输入信号相比，已经具有了一定的稀疏性，这是有傅里叶变换的性质决定的，如果是非常有规律的行信号，其中间频谱可能只有常数个频率。在这个中间频谱上的再一次傅里叶变换，其结果频谱相比中间频谱又会变得更稀疏，原因在于输入信号在列方向也是有序的。因此，对于长度为  $n$  的一维离散信号和长度为  $\sqrt{n} \times \sqrt{n}$  的二维离散信号，后者将具有更高的稀疏度。

虽然多维傅里叶变换可以降解为多个一维傅里叶变换的叠加，但是这样做直接的问题就是性能问题。二维离散傅里叶变换把原来对  $n$  长离散信号的一个变换，化为  $2\sqrt{n}$  个对长度为  $\sqrt{n}$  的离散信号的变换，在更高的维度，这样的切分越细，转换出来的变换个数出越多，虽然每个变换的输入规模变小了，但是如此多的短信号傅里叶变换反而增加了算法的复杂结构以及程序在运行时读取内存的次数。如果目标是为了计算稀疏频谱而在

计算每个维度时使用上一章中提到的任何一个算法，其性能都是糟糕的，例如对于表现较好的第二个算法，其只有在输入信号大小达到  $2^{13}$  时，且稀疏度满足一定条件时，其性能才能相对 *FFTW* 才会有提升，而且这也是理论上得，事实上程序运行时频繁的读取不同的短信号和跨内存访问都将很大的影响性能。

综上所述，对于多维信号，计算其傅里叶频谱至少存在两个问题：

1. 多维信号的傅里叶频谱更为稀疏，快速傅里叶算法浪费大量计算资源。
2. 逐维降解求解傅里叶频谱的算法性能较差，不适合做为频谱计算的方法。

因此，希望能找到一种针对多维信号整体的算法，不需要逐维拆解计算傅里叶变换，而是能够有效利用信号在多个维度上的结构关系，从整体上计算整个多维稀疏频谱。而本文的研究，正是希望能够对二维离散信号，找到这样的一种解决方案。并且，希望对更高维的情形，提供一些有用的参考。

在本章重点研究的，是一个针对二维稀疏频谱表达的理想算法，该算法无需逐维降解输入二维离散信号，而是考虑二维信号行与列之间的关系，从整体上对二维稀疏频谱进行恢复。并且，由于最后频谱的稀疏性，按照亚线性算法原理，无需遍历所有信号，具有迭代次数少，采样复杂度低，占用内存少等特点。但该算法假设受很多约束条件限制，并不能运用到实际的数字图像处理中，因此本文更偏向于将其看做为一种二维稀疏频谱恢复算法理论，并构成第四章应用算法的理论基础之一。

### 3.2 二维严格稀疏频谱表达算法关键技术分析

二维严格稀疏频谱表达理想算法来源于 *MIT* 发表的文章<sup>[3]</sup>，其中介绍了该算法，并分析了算法不收敛的情形，并通过一个强约束条件控制不收敛概率。对于不满足强约束条件的情形，文中提出了基于线性编码解码的稀疏频谱计算方案，其原理为第一章中介绍的基于线性编码理论的信号稀疏表示方法。相比之下，针对满足强约束条件的信号的算法具有非常好的亚线性，其通过巧妙的行列交替迭代策略，使得算法的复杂性，迭代次数，采样复杂度，运行时间都非常理想，但最大的不足之处就强约束条件比较苛刻，使得能够使用的信号非常少。本章首先分析了该理想算法的原理，并对其不收敛情形提出了改进的方案，突破强约束条件，能够使用于更多的严格稀疏频谱信号。虽然改进后的算法仍然不能直接用于普通噪声稀疏频谱信号，但其算法思想通过一些特殊的改善手段，可以用于各类信号，甚至更高维的信号。

### 3.2.1 二维严格稀疏频谱计算主要策略

第一章中介绍了两个稀疏表达算法，都是通过采样输入信号，通过时域和频域的关系，在时域中找到和频域中等价的计算方法，来定位频率的相位并估算幅值。而对于二维离散傅里叶变换，最终的频谱是行列两次傅里叶变换的结果，此时的时域是原始的二维离散输入信号，频域是两次变换最后得到的稀疏频谱，而第一章中算法用到的策略只能在一次傅里叶变换的时域和频域内寻找等价的计算方法，因此，对二维和多维情形，无法直接在输入信号上进行恢复。对于二维离散信号，可以在一次傅里叶变换后的中间频谱上实施恢复策略，因为中间频谱只需再经过一次傅里叶变换即可得到最终频谱，如(3.1)式。也即对于二维离散信号的频谱恢复策略，输入信号是中间频谱，要恢复的是最后的二维稀疏频谱。

首先要获取中间频谱，由于最后频谱的稀疏性，并不需要计算出完整的中间频谱，正如第一章算法中，算法用来计算稀疏频谱的信号只是整个输入信号的采样。但这里并不需要采样中间频谱，而是采样输入信号，对采样信号计算一维傅里叶变换得到的就是采样中间频谱。

二维离散傅里叶变换也是一种线性变换，由公式(3.1)所示，任意一个维度上得变换顺序并不是严格要求的，可以先是行一维傅里叶变换，也可以先进行列一维傅里叶变换，因此，中间频谱也就具有两种形式，即由行变换获取的频谱或由列变换获取的频谱。任意一种形式的中间频谱都可以用于最后的稀疏频谱恢复和表达，并且，这两种形式其实是互补的，表现在任何一行都和所有列相交，所以任何一行中间频谱和任一列中间频谱都会有一个公共频率，如果这正是一个稀疏频率，可能对于行中间频谱，并不能恢复该频率，而在列中间频谱中，却能很简单恢复。因此，综合利用行列中间频谱作为稀疏频谱恢复的输入，会比单纯使用一个方向上的中间频谱更好。

综上所述，总的恢复策略是按行和列采样二维输入信号，计算其一维傅里叶变换，得到行和列的中间频谱，并结合行列中间频谱，恢复二维稀疏频谱。如图 4 所示，红色的行或列是采样序列，并计算了其一维傅里叶变换，在得到的中间频谱上，计算二维稀疏频谱。如左图所示，其采样了行信号并得到中间频谱，同一列行采样中间频谱可以用于计算列方向傅里叶变换的稀疏频谱，如图中箭头所示。右图同理。

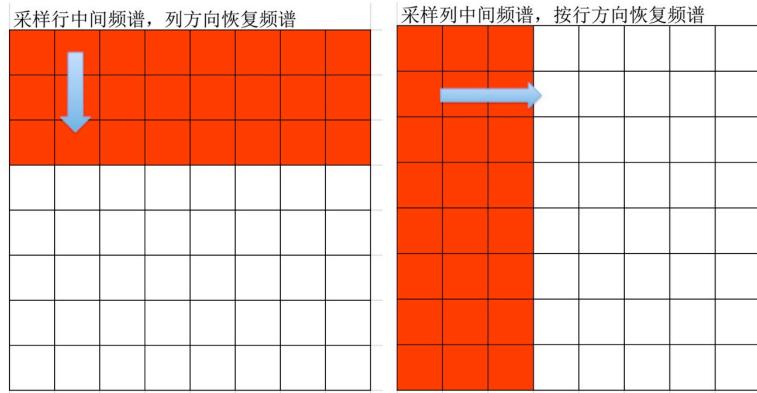


图 4 组合行列稀疏频谱恢复策略

### 3.2.2 理想一维 *one-sparse* 频谱计算方法

$n$  长信号频谱中只有一个幅值非零的频率，这样的频谱称为 *one-sparse* 频谱<sup>[3,9]</sup>，如果待处理的一维离散信号中有这样的信号，那么是否有办法检测出这样的信号，并通过一个简洁高效的算法计算得到这个唯一稀疏频率的幅值和相位。

不妨先假设  $u$  就是这样一个  $n$  长离散信号，其频谱可以表示为：

$$\begin{cases} \hat{u}_k = a & k = s \\ \hat{u}_k = 0 & k \neq s \end{cases} \quad (3.2)$$

$s$  是唯一稀疏频率的相位，也即在频谱序列中的下标， $a >> 0$  为该频率的幅值。在计算前，并不知晓这两个值，唯一知晓的只是输入信号向量在每个维度的值即  $\{u_0, u_1, u_2, \dots, u_{n-1}\}$ ，算法需要做的，就是对这样一个输入信号，能够判断其频谱是否为 *one-sparse*，如果是，能否快速正确的计算该频率的信息。

根据一维离散傅里叶逆变换公式(2.4)，可以在(3.2)式假设的频谱下，得到时域信号在该频域的表达形式，也即(2.5)式的矩阵，这里只需将矩阵元素替换为(3.2)式假设频谱的值，并只列举前几个维度上得值在频域的表达：

$$\begin{cases} u_0 = \hat{u}_s \\ u_1 = \hat{u}_s \omega_n^s \\ u_2 = \hat{u}_s \omega_n^{2s} \end{cases} \quad (3.3)$$

式(3.3)列出了一组由三个方程组成的方程组，并且发现其中只有两个未知数，因为方程组代表的是输入信号时域和频域变换关系，所以方程组一定有唯一解，并且，由上式，只需两个方程就可以计算其解：

$$\begin{cases} a = \hat{u}_s = u_0 \\ s \leftarrow \omega_n^s = u_1/u_0 \end{cases} \quad (3.4)$$

通过(3.4)式，实际上已经解出了(3.2)式假设的 *one-sparse* 频谱的稀疏频率的幅值  $a$  和相位  $s$ ，其中  $s$  只需一步对数计算。也就是说存在一个方法，只需采样  $n$  长输入信号最开始的两个维度的值，就可以在常数时间复杂度内恢复计算出该信号频谱中唯一稀疏的频率，但前提是该信号的频谱确实是 *one-sparse* 的，验证输入信号是否服从这个假设，只需把恢复出得频率幅值和相位代入(3.3)方程组中的第三个方程，检验其是否是方程的解即可。

以上算法是针对 *one-sparse* 频谱最为简单高效的稀疏频谱恢复方法，又称作 *OFDM* 方法<sup>[3]</sup>。算法假设的条件非常理想，以致对于一维离散信号，很难有适用该算法的情形。算法的意义在于揭示了稀疏频谱恢复表达算法的本质只需计算采样信号，定位相位和计算幅值。并且，对理想算法的不断改进使之适用更宽松的假设，最终会得到一个切合实际应用的算法。本章行列恢复策略正是基于这一理想算法，并且本文关于二维图像稀疏频谱表达算法的研究也是借鉴了该理想算法思想。

### 3.2.3 算法原理及问题分析

本节算法基础理论来自 *MIT* 对二维严格稀疏傅里叶变换的尝试，其建立对二维稀疏信号的两个前提假设之上，分别为：

1. 假设二维信号频谱是严格稀疏的，即没有噪声影响，每一个频率或稀疏，幅值显著大于零，或严格等于零。
2. 其次整个二维稀疏频谱服从伯努利分布，设二维信号总长度为  $N=n \times n$ ，稀疏度为  $k$ ，每个频谱坐标上的频率都具有相同的概率稀疏，其概率为  $k/N$ 。

本章并没有对有噪声的稀疏的情况下稀疏频谱表达进行研究，原因在于本章算法主要是作为第四章算法的理论之一，对噪声的解决方法将在第四章中算法中详细介绍。

假设当稀疏度  $k=an$ ，并且  $a < 1$ ，可以认为每一行或列中非零稀疏频率个数期望为小于等于 1，由此可以认为每一行或列构成了一个 *one-sparse*，因此通过检测并运用 *one-sparse* 算法来恢复这些行或列的稀疏频谱。

以图 5(a)中  $8 \times 8$  的二维信号为例，分析算法原理：

第一步：首先，算法组合利用行列稀疏频谱来计算二维稀疏频谱。首先算法采样了图 5 (a)所示信号的最开始三行和最左边三列。并计算了其一维傅里叶变换得到行采样稀

疏频谱序列和列采样中间频谱序列。作为计算二维稀疏频谱的直接输入。

第二步：接下来就是交替的行列中间频谱的迭代，图 5 (b)中逐行读取了列采样中间频谱序列，判断该行是否为 *one-sparse* 频谱。判断的方法就是在上一节 *one-sparse* 原理中介绍的那样，即先假设其位 *one-sparse* 并计算出该稀疏频率，通过该稀疏频率验证是否该行的所有中间频谱序列都可以由该频率表达。图 5(b)中判断第一行为非 *one-sparse* 行，继续判断第二行。第二行判断为 *one-sparse* 行，因此经 *one-sparse* 计算得到的频率就是该行的唯一稀疏频率，位于该行频谱序列的第六个位置。计算得到该 *one-sparse* 稀疏频率后，需要更新行采样中间频谱。此时就是更新行采样中间频谱的第六列值，更新的方法是第六列行采样中间频谱序列减去该频率乘以相应的傅里叶权重。例如，对于第  $i$  行行采样中间频谱，其更新方法为：

$$S_{r,j} = S_{r,j} - y_{i,j} \omega_m^{jr} \quad (3.5)$$

式中  $S_{r,j}$  为行采样中间频谱第  $r$  行第  $j$  列中间频谱信号，此处  $j=6$ ， $y_{i,j}$  为刚恢复二维频谱的第二行第六列稀疏频率，此处  $i=2$ ， $m$  为二维信号的列高。这一列所有行采样中间频谱更新后，继续下一行迭代。图 5 (b)中，重复以上过程，依次判断剩下所有行，例如，判断第三第四行，操作和第一行操作相同。第五行操作过程如第二行。

第三步：按行方向迭代列采样中间频谱后，接下来按列方向迭代行采样中间频谱。如图 5 (c)所示，逐列读取行采样中间频谱，判断该列是否为 *one-sparse*，如果是，则以 *one-sparse* 恢复的频率为该行的稀疏频率，并更新列采样中间频谱。否则，继续下一行判断，方法和第二步中行方向迭代类似。主要注意的是，对于第 5, 6 列的 *one-sparse* 判断，从图 5(a)中看，行方向第 5, 6 列的并非 *one-sparse*，这两列都具有两个稀疏频率，但是在图 5(b)的行迭代中，分别恢复了这两列的一个稀疏频率，并更新了行采样中间频谱，使得图 5 (c)对第 5, 6 列的判断就是 *one-sparse*。就是这种行列的迭代更新，使得迭代能够继续进行，不断的解耦并恢复稀疏频率，直到所有的稀疏频率被计算。

第四步：重复第二，三步的迭代过程，直到所有的稀疏频率被计算完成。

算法简洁高效，充分利用了二维矩阵信号的特点，通过  $\log N$  次行列交替迭代和反馈更新。使用 *OFDM* 方法快速定位恢复稀疏频谱，整个算法运行时间复杂度为  $O(n \log N)$ ；

以上算法思想必须满足对二维频谱稀疏度  $k=an, a < 1$  的理想设想，并且，其存在的一个最大的问题是：算法本身不收敛。另外例如迭代次数无法确定，伪代码中取  $C \log N$  其实是迭代次数的一个上界，以保证足够的迭代次数。

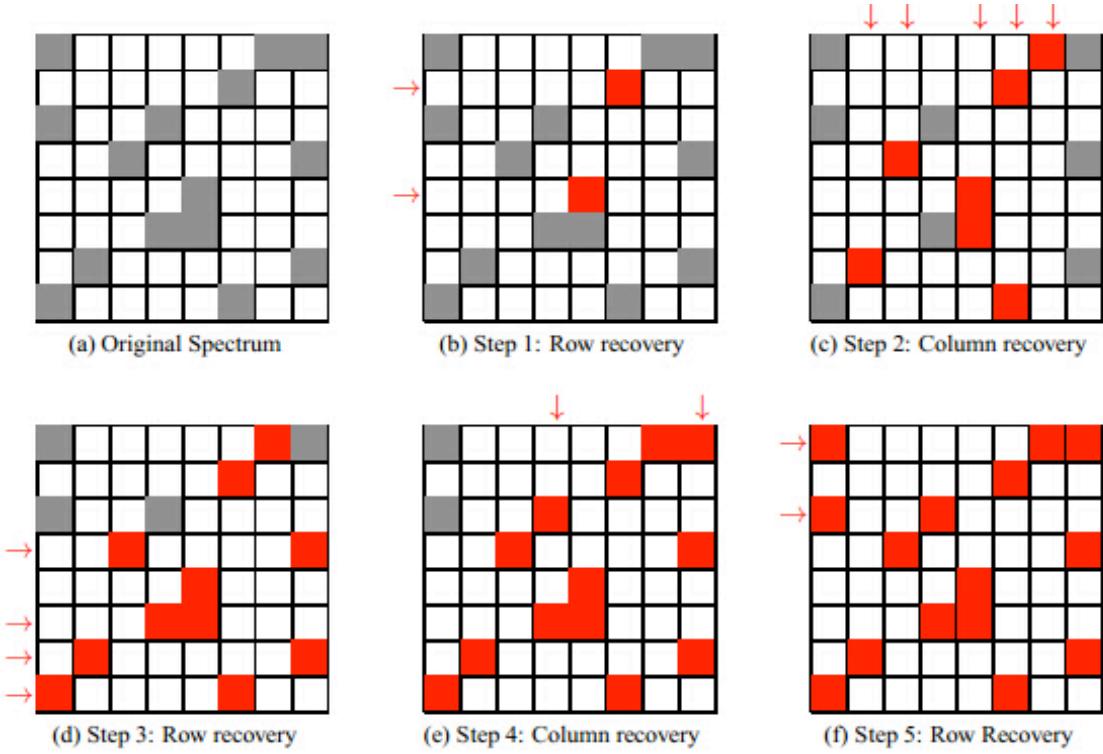


图 5 二维 one-sparse 稀疏频谱恢复算法

存在一种情形使以上的算法不收敛，其主要表现为，如果当多次迭代恢复后剩下的稀疏频率分布呈现如图 6 所示的状态，算法将进入死锁，每一次行列迭代都不会恢复出新的稀疏频率。直到设定的迭代次数迭代完成，算法失败退出：

如图 6 (a),(b) 的两种情形，无论从行的方向还是列的方向，都无法通过 one-sparse 验证，也无法通过其他方向的更新来降解该方向的稀疏度，所以每次迭代都无法恢复任何稀疏频率，之后每次迭代的情况相同，直到迭代完成。

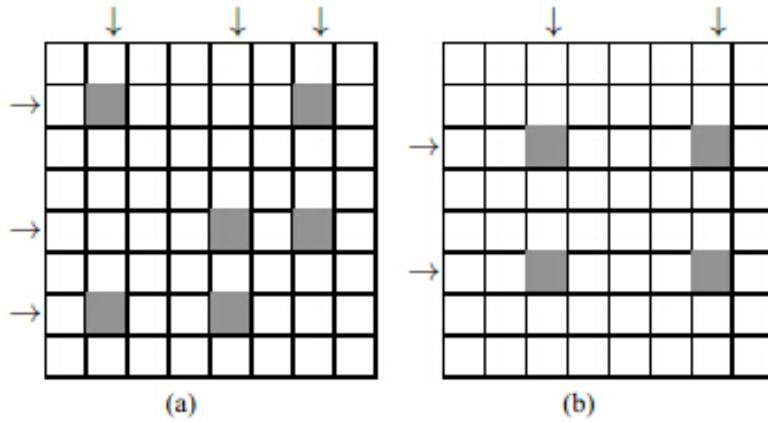


图 6 算法不收敛情形稀疏频谱分布

上述不收敛情形稀疏分布的特点是，稀疏频率按行或列的方向构成一个闭环，任何稀疏的行或列都有超过一个幅值不为零的频率，并且无法通过其他方向的迭代来降解。

MIT 的文章中分析了这个不收敛情形，并通过本节一开始的两个假设来控制这种情形发生的概率是一个小概率事件<sup>[3]</sup>。这样做带来的问题是，虽然假设保证了算法的成功概率，但对两个假设的强烈约束限制了其可以使用的范围。稀疏度限制为只能是小于等于行宽或列高，因此这只能看做是一个理想算法。

随着稀疏度  $k$  越大，算法失败的概率越大。因此，对所有不满足强约束条件的情形，MIT 的文章中提出了另一种算法来弥补当稀疏度  $k$  较大时的情形。当  $k > O(n)$  时，为了规避以上导致以上算法失败的情形，采用另一种稀疏频谱恢复策略，其原理是采用第一章中介绍的基于线性编码理论信号稀疏表达方法思想，采样列，计算中间频谱，逐行诊断行稀疏频谱，其原理见第一章算法分析，然后通过  $BM$  算法构造错误定位多项式，并采用一个高效的求解多项式的根的算法  $PAN$  算法<sup>[3]</sup> 来得到稀疏频谱的位置，得到位置下标后通过一个矩阵向量乘法得到稀疏频谱的幅值。相比理想的行列组合恢复策略，这种编码诊断策略的效率是很低的，每一行诊断的时间复杂度为  $O(n^2)$ 。

如果事先知道稀疏频谱的位置，那就不需要通过  $BM$  和  $PAN$  算法而可以直接由矩阵向量乘法求解稀疏频谱；算法逻辑及结构将得到极大改善，运行时间复杂度也将有很大提升。

### 3.3 改进二维严格稀疏频谱表达算法的设计与实现

本节主要是对上一节理想算法的改进，希望改进后的算法克服原算法中得不收敛情形，而不需要对稀疏度  $k$  施加严格的约束，算法在稀疏度  $k = O(n)$  时可以保证收敛，在  $k > O(n)$  时算法能否成功恢复频谱取决于采样行或列的设定，当采样足够的行或列时，算法仍然能够收敛。

总体策略仍然采用理想算法的组合行列频谱恢复策略，算法的迭代逻辑也和理想算法相同，都是使用 *one-sparse* 算法判断和恢复稀疏频率。本节的创新之处在于，加入了两个稀疏频谱恢复标识，来记录行列方向的迭代情况。首先，是一个记录按行方向迭代的稀疏频谱恢复情况的行标识，该标识用一个二进制位来记录每一行稀疏频率是否已经被计算完成，如果已经被计算，则标识为 0，否则保持标识为 1，因此，整个行标识就是一个比特数组，数组的长度等于行的数量。其次是一个记录按列方向迭代的稀疏频谱恢复情况的列标识，使用一个二进制位来记录每一列的稀疏频率是否已经被计算完成。列标识也是一个比特数组，数组长度等于列的数量。这样的行列标识可以用 C++ 的 *bitset* 实现，因此这里称为行列 *BitSet* 标识。

在行列迭代计算稀疏频谱是更新标识状态，例如，以图 6(b)为例，按行方向迭代列采样中间频谱，第一行判断为非 *one-sparse* 频谱，因此其稀疏频谱等待以后计算，行标识的第 1 位为 1 保持不变。第二行是 *one-sparse* 频谱，可以计算其稀疏频率值，因此完成计算后行标识第 2 位更新为 0。图 5(c)迭代完成后行列标识的状态如图所示。

行列标识使得稀疏频谱恢复过程变得可视，在按列方向计算稀疏频谱时，行标识中非零比特位的位置提供了该列稀疏频率可能的位置信息。这里是一个包含关系，该列在行标识中某个非零比特位的位置未必稀疏，但在该列的稀疏频率的位置行标识的比特位一定非零。

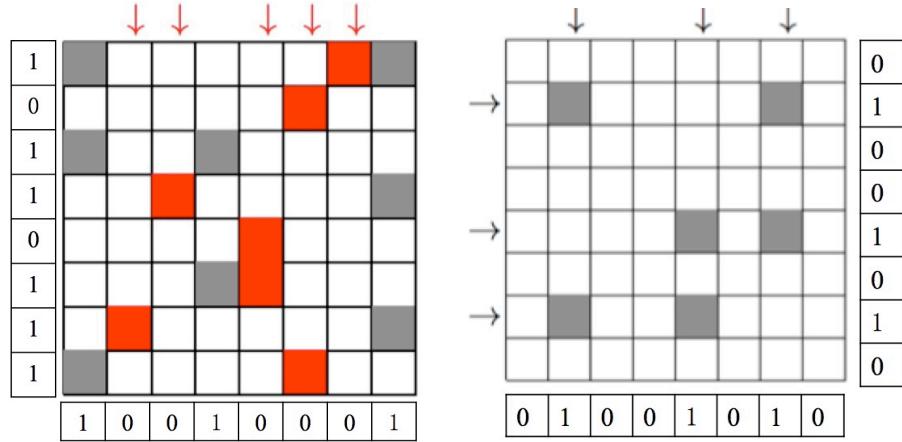


图 7 加入行列标识的迭代

使用 *BitSet* 记录行列稀疏恢复情况的最大好处是，可以有效解决上一节算法中得不收敛问题，仍然以上一节中不收敛稀疏分布情形的图 6(a)为例，当算法迭代中遇到此稀疏分布时，上一节算法将进入死锁最后失败退出，但假如使用 *BitSet* 后，列方向的迭代在计算恢复第二行频谱时发现其并非 *one-sparse* 的，但是通过检查行标识非零比特位后发现稀疏频率只可能出现第 2, 5, 7 列，因此可以大胆的认为第二行频谱中最多只有第 2, 5, 7 列的频率稀疏，知道了位置，就可以通过一个矩阵向量乘法，计算这几个位置的频率系数。详细推导和计算过程如下：

$$\left\{ \begin{array}{l} x_0 = \hat{x}_1 + \hat{x}_4 + \hat{x}_6 \\ x_1 = \hat{x}_1 \omega_n^1 + \hat{x}_4 \omega_n^4 + \hat{x}_6 \omega_n^6 \\ x_2 = \hat{x}_1 \omega_n^2 + \hat{x}_2 \omega_n^8 + \hat{x}_6 \omega_n^{12} \end{array} \right. \quad (3.6)$$

写成矩阵运算形式：

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \omega_n^1 & \omega_n^4 & \omega_n^6 \\ \omega_n^2 & \omega_n^8 & \omega_n^{12} \end{bmatrix} \bullet \begin{bmatrix} \hat{x}_1 \\ \hat{x}_4 \\ \hat{x}_6 \end{bmatrix} \quad (3.7)$$

上述公式中,  $x_0$ ,  $x_1$ ,  $x_2$  和  $\omega_n^i$  都是已知的, 未知变量只有三个稀疏频率幅值。右边矩阵式一个范德蒙矩阵, 是一个非奇异矩阵, 所以方程组有唯一解, 只需求得右边矩阵的逆矩阵, 左乘等号左边的采样信号即可。逆矩阵可以提前计算好放入内存, 不需要在需要时再计算; 本论文中所有矩阵运算都采用开源 C++ 矩阵运算库 *Eigen*。

回到图 6(a) 中, 上述计算方法假设了第二行频谱有三个稀疏值并通过矩阵运算求出了频率的幅值, 事实上如图中所示该行只有两个稀疏频率。但上述计算过程并不会出错, 由于无法准确的估计稀疏频率的数量和位置, 所以计算了它们的所有可能位置上的幅值。对于真正的稀疏频谱, 计算结果是它的一个超集, 其包括非零稀疏频率, 也包括一些错误估计为稀疏而其实非稀疏的频率, 这些非稀疏频率经上述计算方法得到值幅值为 0 或近似为 0。因此在计算中把它当做稀疏值对待, 但并不影响结果的准确性, 只是因多计算了一个非稀疏值浪费了一些计算资源, 但这样的浪费是非常微小的, 相比使用复杂的方法编码诊断方法构造错误多项式并解出其精确的稀疏位置, 这些计算资源开销是非常值得的。

当第二行稀疏频谱得到恢复后, 图 6(a) 中稀疏频率的闭环形分布被解锁, 在后续的迭代中, 可以继续使用 *one-sparse* 算法判断和计算频率。算法可成功收敛。这是使用 *BitSet* 后最大的受益; 另外, 可以通过检查 *BitSet* 所有位是否都为 0 来判断是否频谱恢复完成, 而不用设定一个固定的迭代次数, 迭代次数过多只会造成不必要的计算浪费。最后, 确认被标识为 0 的行或列不用再参与迭代, 仅需测试比特位即可判断。

上述算法依赖采样行或列数的正确设置, 因此最好可以对频谱稀疏度有一个大概的先验估计。如果没有这样先验的经验, 当采样行或列不够时, 程序实现会返回一个提醒, 并告知建议的采样数量后退出, 这个建议的采样数量可以作为下次执行的先验估计值。

另外, 当按行方向恢复频谱时, 在恢复某一行频谱检查行标识时可能发现有较多列稀疏, 但可能真实情况是该行其实只有少量的稀疏频率, 造成行标识显示存在大量稀疏列的原因可能是因为不同行的稀疏频谱在行标识上的共同投影造成的, 此时, 如果仍然采用行标识提供的稀疏分布信息可能需要大量多余的计算, 甚至会因为采样列不足和程序退出。本文实现在此的做法是并非直接退出并提醒用户设置更大的采样列, 而是回先

求助第一章中提到的编码诊断方法，构造错误定位多项式，求解其根来找出真实的稀疏分布，如果确实具有较大的稀疏度并大于现有采样列数，则提醒并退出，否则，可直接采用上述方法恢复频谱，解锁当前频谱并更新采样中间频谱，进入下一次迭代。

以下是算法频谱恢复流程图

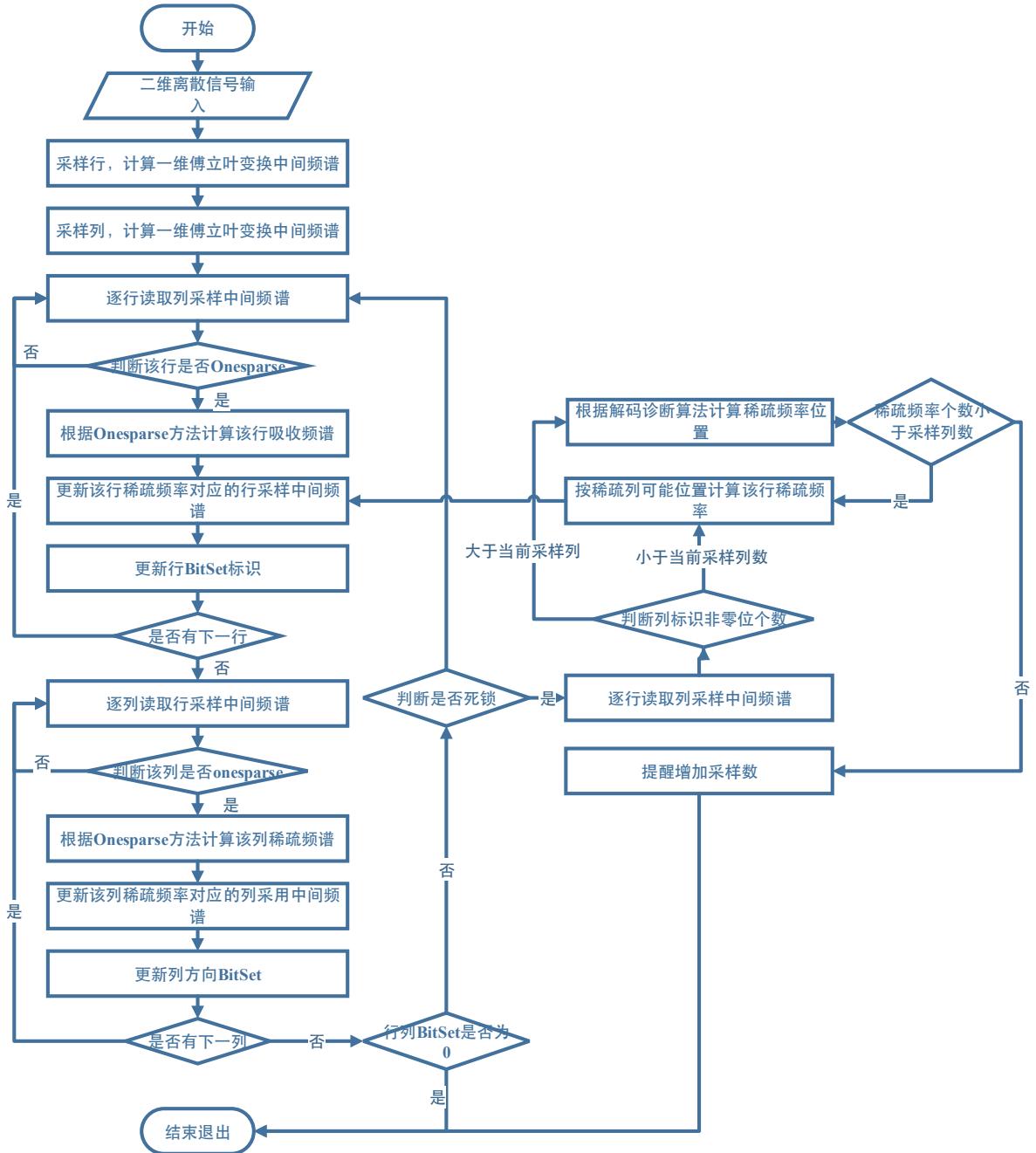


图 8 改进二维严格稀疏频谱表达算法流程

### 3.4 算法分析及实验结果

设二维信号的规模为  $N=n \times n$ ，稀疏度为  $k$ ，采样行和列的数量是为  $S=C_1k/n$ ，对采样

信号的傅里叶变换时间复杂度为  $O(k \log n)$ , 在最坏情况下, 不妨设每次迭代只能恢复计算得到一个稀疏频谱, 因此需要  $k$  次迭代, 每次迭代的算法复杂度为  $O(n)$ , 最坏情况下算法的时间复杂度为  $O(kn)$ 。而对于不同的稀疏度, 最坏情况发生的概率是不同的。当  $k=O(n)$  时, 改进后算法的性能和原算法在  $k=an, a < 1$  情形下的性能相似, 时间复杂度更贴近  $O(n \log N)$ 。当  $k > O(n)$  甚至更大时, 稀疏频率分布变的密集, 发生最坏情况的概率变高, 甚至可能导致改进后算法因为采样不足而失败退出。但在采样行列满足稀疏频谱计算条件的情况下, 改进算法在更大的稀疏度  $k$  时会转换为基于线性编解码的稀疏傅里叶算法, 其时间复杂度为  $O(k \log N \log(N/k))$ , 这可以看做改进算法在不失败情况下更逼近的最坏情形时间复杂度。以下是改进二维严格稀疏频谱表达算法的实验结果及分析。

**实验 1:** 对  $4096 \times 4096$  的二维离散输入信号不同相同稀疏度, 不同采样数下改进二维严格稀疏频谱表达算法, 理想二维严格稀疏频谱傅里叶变换算法, 及 FFTW 的比较。

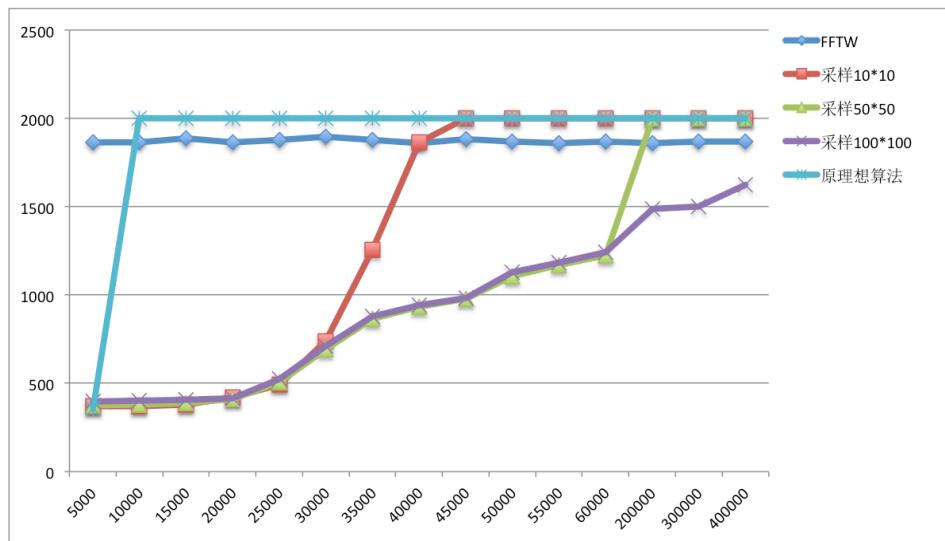


图 9 不同稀疏度不同采样下各算法对比

上图中, 横坐标是稀疏度, 纵坐标是程序运行时间, 以毫秒计。运行环境是 Linux 操作系统, CPU 双核心四线程, 主频 2.6GHz, 内存 8G。FFTW 运行模式选择为 FFTW\_ESTIMATE。需要说明的是, 图中运行时间大于 FFTW 的均属于算法失败情形。

从图中可知, 原算法对  $k=an, a < 1$  的假设非常依赖, 当稀疏度为 10000 时, 算法失败。改进后算法在稀疏度小于 30000 时, 不同采样数对算法的影响不是很大。当稀疏度继续增大时, 不同采样值表现差异较大。分析其原因, 在于当稀疏值小于 30000 时, 没行或两列的稀疏频率个数期望低于 10, 产生死锁的概率较低, 算法性能较高, 即使较小的采样数  $10 \times 10$  也完全能够求解稀疏频谱。当稀疏度进一步增大时, 死锁的概率变大, 而导致程序有时需要借助 BitSet 解锁, 当采样值较小是, BitSet 提高的稀疏信息仍

然不能满足计算需要，因此而求助基于线性编码理论的解码算法，如果解码得到的稀疏信息仍不满足当前计算需求，算法失败退出，如图中  $10 \times 10$  的采样在稀疏度大于 4 万以及  $50 \times 50$  的采样在稀疏度大于 20 万时的表现。采样  $100 \times 100$  在图中所有稀疏度时都未失败，但本文中并未比较更高的稀疏度和添加更大得采样值情形，因为当前采样在稀疏度为 40 万时运行时间已接近 FFTW，对于更高的稀疏度，本文中实现的程序运行时间劣于 FFTW。可以知道的一个原因在于 FFTW 是一个高度优化的程序，其性能相比普通快速傅里叶算法要高出许多。而本章所改进二维严格稀疏频谱表达算法亦是一种理想状态的算法，其意义在于为第四章实际应用中的稀疏频谱表达算法提供指导思想和策略。因此实验程序并未做相应优化，只是简单算法实现。

**实验 2：**对  $4096 \times 4096$  的离散二维输入信号在不同稀疏度和相同采样值下，改进二维严格稀疏频谱表达算法结果正确性对比。对比方法为将其计算得到的稀疏频谱与经 FFTW 得到的频谱计算最小均方差 MSE，通过 MSE 值的大小来判断算法输出的准确性。

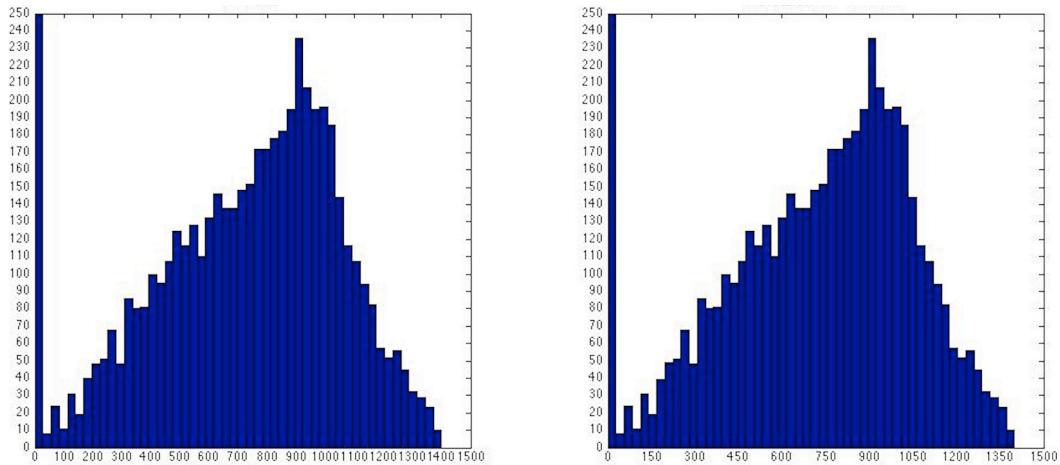


图 10 稀疏度 5000 二维严格稀疏频谱表达算法对比 FFTW 频谱  $mse=0.0011387$

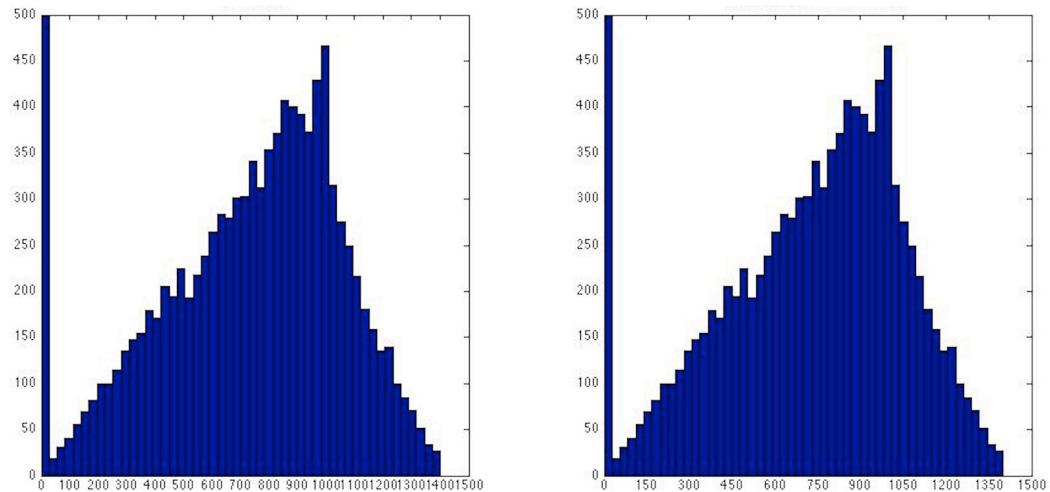


图 11 稀疏度 10000 二维严格稀疏频谱表达算法对比 FFTW 频谱  $mse=0.0018217$

上面两幅图是稀疏度分别为 5000 和 10000 时的稀疏频谱幅值分布直方图, 以  $FFT W$  变换得到的频谱为源频谱幅值信号, 经改进后二维严格稀疏频谱表达算法在采样  $50 \times 50$  的时得到的稀疏频谱为目标频谱幅值信号, 计算源频谱幅值和目标频谱幅值的最小均方差来对比两个频谱信号的差异。由于在两个变换中得幅值都是双精度数, 因此精度上得差异也会导致  $MSE$  值的轻微差异。下表是在其他不同稀疏度下的  $MSE$  统计值, 及程序多次运行时间统计:

**表 1 二维严格稀疏频谱表达算法在不同稀疏度下  $50*50$  采样的实验结果**

稀疏度	$FFT W(ms)$	二维严格稀疏频谱表达算法(ms)	$MSE$
5000	1866	376	0.0011387
10000	1865	381	0.0018217
20000	1889	412	0.0019374
40000	1866	932	0.0022573
60000	1871	1224	0.0023892

### 3.5 本章小结

本章从最简单的二维稀疏频谱恢复  $OFDM$  算法开始, 阐述了理想二维严格稀疏算法原理, 约束条件, 不收敛条件, 以及在不满足约束条件时现有的做法。针对这些算法, 本章一一介绍了其中的不足和缺陷, 并在改进算法中, 对原来强约束的理想算法提出了去约束的解决方案, 不仅实现了算法在更大的稀疏度下可收敛, 并优化了迭代次数参数由运行数程序判断, 避免了过多的迭代。最后, 对  $FFT W$  和理想算法, 及改进后算法在不同稀疏度下对满足假设条件的二维信号做了实验对比, 证明了本章算法在稀疏度合理的范围内的最高效的。但本章改进算法仍然有很大局限性, 在图像和很大数字信号中无法直接使用。在第四章中将讨论在本章算法思想上针对图像应用的改进方法。

## 第四章 二维图像稀疏频谱表达算法的设计与实现

数字图像是一种非常直观的信息表达和传递的方式，对数字图像信号处理的研究也备受重视。二维或三维图像的傅立叶频谱通常也具有稀疏的性质，只需要少量的频谱信号就可以表示大部分图像信息，而在图像分类，人脸识别，车辆检测等领域，也更强调对图像整体语义的识别。用稀疏的傅立叶频谱作为图像表达能够很好的给图像信号降维，并不影响图像的分类识别等操作，而且傅立叶分析的理论也可以适用。

### 4.1 二维图像频谱的稀疏特性

二维数字图像在信号处理领域广泛使用的信号对象，在空间表示上，二维图像按像素点排成一个二维矩阵。图像信号是非常有规律信号，表现在每个像素点的值在不同方向上都具有一定的连续性，相邻两个像素点的值一般不会发生值大小的突变，正是因为遵循了不同的排列规则，这些二维像素点构成了不同类型的图像。越是规律的信号，其频谱表现越是简洁。因此，其数字图像傅里叶频谱也具有一定的规律性，图 12 是一个灰度图像和其频域幅值对数图：

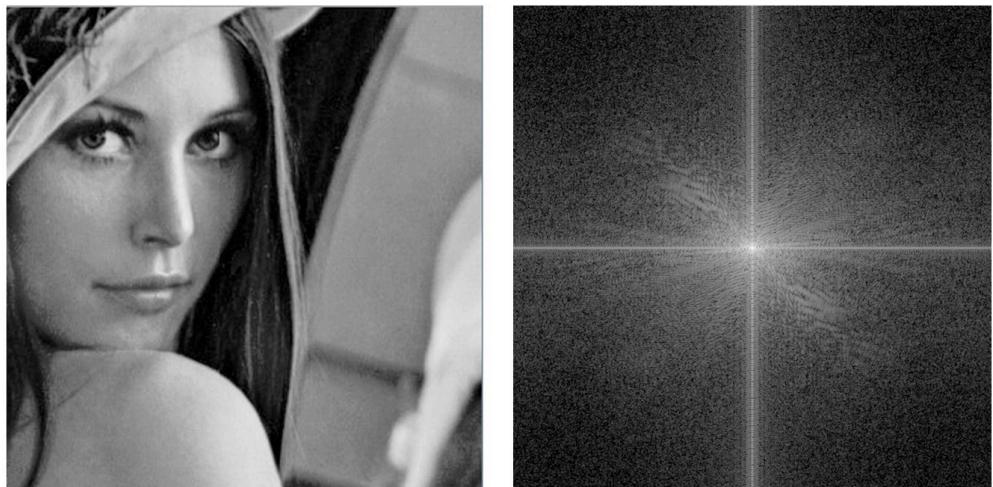


图 12 二维空间图像及其频谱对数图

上图中，左图为空间图像，右图为其对应的频谱图，为了方便处理和显示，对计算频谱时对频谱做了居中归一化处理，并且频谱幅值最大值最小值相差较大，为方便显示，使用对数归一化，右图就是归一化后的灰度图显示，图中，位于中心的是零频分量，对应空间图像中所有像素值的线性叠加，因此也称直流分量。频谱中低频的部分一般对应了空间图像中总体的效果趋势，而高频成分对应图像的边缘和细节，也包括一部分的噪声。

图 12 中右图频谱分布已经能够看出一定的稀疏性，其中较暗的像素点对应的频率幅值为零或近似为零，由此看来，该频谱的稀疏并不如理想中得明显，甚至在较高频的位置，任然有很多亮点，图 13 是对频谱对数分布图的幅值统计：

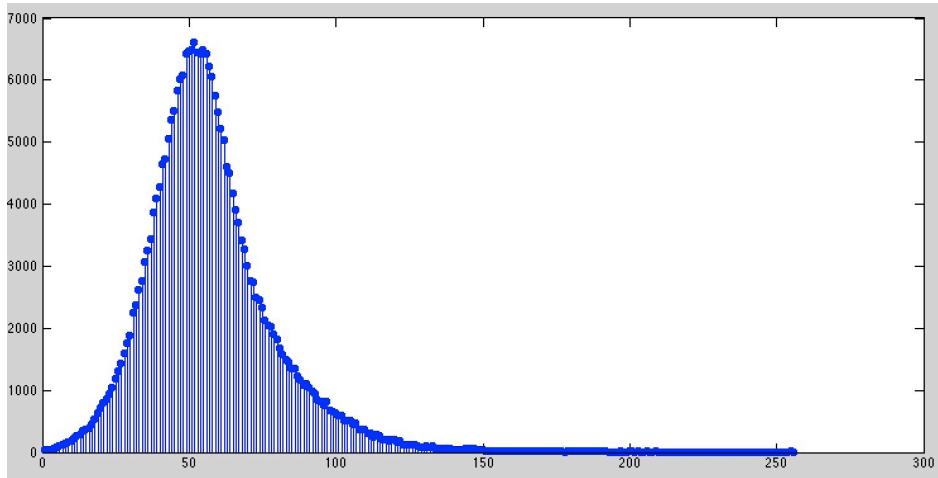


图 13 对数频谱幅值统计分布

对频谱幅值统计直方图的观察发现，幅值严格为零的频率并不多，即使包括一些幅值近似为零的频率，整个频谱的稀疏度也并不理想。但事实上，对于数字图像这类二维有序信号，其频谱应该是非常稀疏的。产生这种误判的关键原因之一在于数字图像中得噪声，噪声是一种随机信号，其分布并不遵循任何规律，对于这类快速变化的信号，其频谱表现一般为高频分量，即需要更高频率，周期短，变化快得周期信号的线性叠加来拟合逼近。另外，对图 12 中频谱分布图的观察发现，整个频谱的分布也呈现一定的特点，大量幅值较大的频率集中在中心点附近，并由中心向外，幅值大小逐渐递减，非常类似高斯分布。

影响图像频谱稀疏度判断的主要原因之一，在于随机噪声在频域中产生的高频成分，如果能去除其影响后再评价图像的频谱稀疏性会更准确。去除噪声的方法有很多，由于大部分噪声信号在图像频谱中体现为具有较高的频率的部分，如果考虑截去高频分量这种方法来去除频域中的噪声，可以直接对频谱信号乘以一个二维矩形函数来达到目的，但这种简单粗暴的高频截断方法往往会对原空间图像造成破坏性的影响。原因在于空间图像和频域图像是一种等价表示，二者都是一种有序信号，任何一方的有序性被强行破坏都会影响另一方的表达效果。高频截断相当于是频域中被截断部分信号的突变，其在空间图像中得影响是引入新的噪声，在数字图像处理中称作环(ring)。推荐的简单降噪方法是使用一些平滑的滤波函数，例如巴特沃斯函数和高斯函数。尤其是高斯函数由于其在时域和频域的函数不变性，更适合作为平滑滤波函数。另外一点是二维图

像本身的频域分布类似高斯分布，因此可以假设图像分布就是一个高斯分布和一个随机噪声分布的叠加，对其施加二维高斯函数后，服从高斯分布的频率成分被保留，而不服从高斯分布的频率则被抑制，如果被抑制的都是噪声信号，那么就很好的分离了原图和噪声，达到了理想的去噪滤波效果，但实际的情况是即使是无噪声的二维图像的频谱也不能严格的服从高斯分布，所以，不可避免的会有一些非噪声的频率被抑制，而这些频率很可能正是空间图像中某些细节的体现。但幸运的是，在很多场合下需要关注的往往是图像整体的检测和识别，某些非常细节的部分并不需要考虑，如人脸识别，图像分类等图像处理中，一般使用高斯函数滤波做图像预处理，而在像在注重清晰边缘和图像细节等研究中，使用高斯函数滤波预处理往往适得其反。在本文研究对二维图像的稀疏频谱恢复表达中，也假设了图像的部分稀疏频谱服从是高斯分布，算法最后恢复的频谱，是忽略了图像的小细节和噪声部分，只保留图像总体和局部特征的图像频谱。最后，本文会根据出一个评价准则，来判断算法稀疏表达效果的优劣。

下图是经高斯函数平滑处理后的二维图像及其频谱分布图：

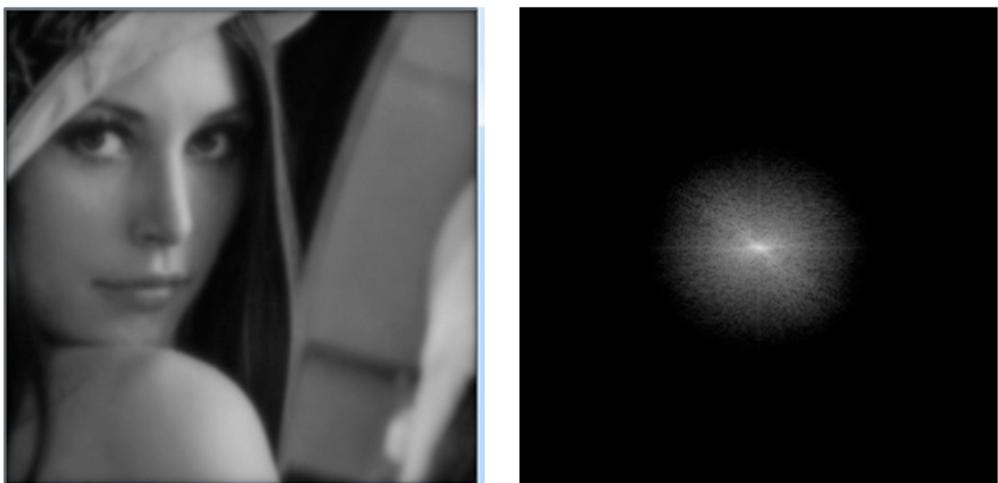


图 14 高斯平滑后的图像及其对数频谱

从图中可以看出，图像的傅里叶频谱乘以高斯函数后，其相应的空间图像相比原图变得柔和了，颗粒和纹理噪声在平滑后的图像中的表现也被弱化了。但是整个图像的轮廓变得不是很清晰。

下图是高斯函数过滤后频谱对数幅值统计图，如果把对数幅值在 50 以下的都认为是噪声或系数逼近为零的频谱信号，那么对于大小为 512\*512 的频谱，大概只有 2500 多个严格稀疏非零频率，这些频率大都对应频谱对数图中中心点零频分量附近较亮的部分，除了一些图像的细节，这些频率已经足以反应空间图像的特征，也能够满足图像识别和分类要求。

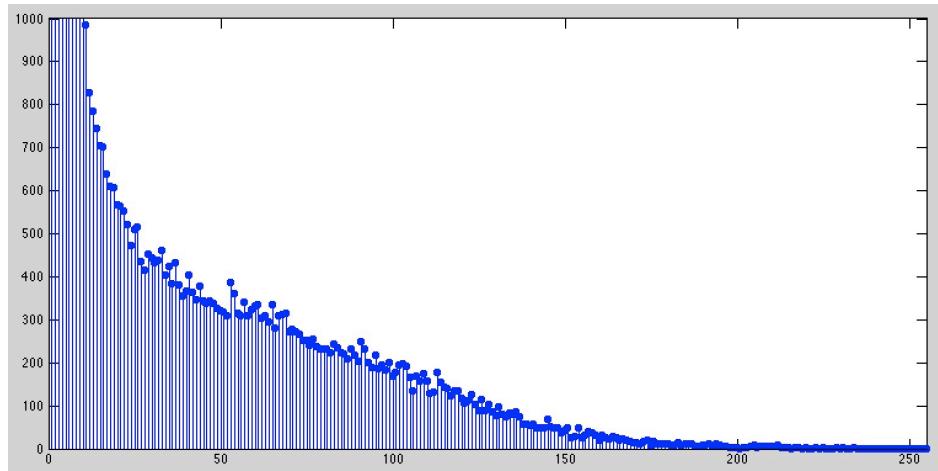


图 15 高斯平滑后的对数频谱统计分布

综上所述，二维图像的频谱分布特性大致呈现为类似高斯分布的集中分布，并且稀疏度满足能够  $k=O(n)$ ， $n$  假设是其中一个维度的长度。并且，二维图像的频谱稀疏特性体现在频谱并非严格稀疏，而是包含很多噪声的近似稀疏，并且稀疏频谱并不如上一章中假设的那样，服从伯努利随机分布，而是具有更强的有序性，近似服从高斯分布。这两个特性使得第三章算法并不适用与二维图像稀疏频谱的恢复和表达。

## 4.2 二维图像稀疏频谱表达关键问题

以上对二维图像频谱的分析表明，实际图像频谱通常并不具有明显的严格稀疏特性，因为噪声信号的存在，使得频谱高频成分幅值有很大概率不为零，另外，即使是去噪后的图像，其频谱仍然是一种噪声稀疏，并不能够像第三章中理想的信号那样具有严格的稀疏性质。图像频谱是表达更多的采用一种近似逼近的稀疏表示方法，存在一定的误差，这种稀疏表示方法将一定阈值范围内的小幅值频率视为非稀疏的频率，并主要致力于恢复表达那些幅值显著不为零的稀疏频谱。但是如此一来，第三章中很多方法变得不再可用，OFDM 方法<sup>[3]</sup>和基于线性编码理论的稀疏频谱恢复算法<sup>[21]</sup>都是基于频域严格稀疏的原理，对于近似稀疏的频谱计算则无能为力，另外，二维图像的频谱分布也不再服从第三章中假设的伯努利分布，这使得第三章的所有算法都不能直接在二维图像频谱恢复上取得成果。因此，要解决二维图像的稀疏频谱恢复表达，必须解决两个关键问题：

1. 如何在使用采样信号恢复频谱过程中尽可能多得排除掉噪声信号的影响，使能够正确的恢复主要稀疏频谱。由于在空间图像中，噪声信号混迹于真实图像信号中，不像频谱中那样按频率高低分层而来查找过滤，因此，从采样空间信号恢复其稀疏频谱，噪声信号的存在将影响最后恢复频谱的准确性，甚至所恢复频谱的相位也会随

之发生偏移。

2. 找到一种计算信号频谱近似服从高斯分布的信号稀疏频谱表达的方法。二维图像的稀疏频谱分布较为集中，且呈现一定的特点。在低频区聚集大量描述图像主要特征的频率，这部分频率幅值较大，是图像稀疏频谱的主要组成部分。由于其分布特点与第三章改进二维严格稀疏频谱表达算法所假设的信号频谱分布差异较大，无法直接使用第三章算法。

### 4.3 关键问题解决方案

第三章提到的改进后收敛算法必须是建立在两个假设之上，即稀疏度  $k$  在合理的范围内最好是  $k=O(n)$ ，另外，稀疏频谱分布服从伯努利分布，每个频率是否稀疏的概率是相同的。在二维图像稀疏频谱恢复中，稀疏频谱分布更接近服从高斯分布。而对于第三章改进算法中，对信号频率分布服从伯努利分布是整个算法的核心，也是能够交替行列迭代使用 *one-sparse* 方法计算稀疏频率的前提条件，因此，*one-sparse* 方法不能适用当前情形。

#### 4.3.1 近似高斯分布稀疏频谱的计算策略

比较第二章中介绍的信号稀疏表示算法和散列频谱计算稀疏傅里叶变换的算法，前者通过 *BM* 算法来构造错误定位多项式<sup>[21]</sup>，并通过解多项式的根来找到稀疏频率的位置，然后在得到各个稀疏频率在频谱中的位置后，利用设计好的基向量矩阵与输入信号的乘积求得每个稀疏位置处的幅值。后者的方法也是类似，先通过哈希频谱的方法散列频谱，找到所有可能的稀疏位置，然后在稀疏位置计算幅值。*one-sparse* 是同时计算了稀疏位置和幅值。第三章改进后二维严格稀疏频谱计算方法在求解频率分布出现死锁情形时，也是通过询问行列 *BitSet* 标识来先判断可能的稀疏位置，再在其上计算幅值。因此，可以概括求解信号稀疏表达的一般两步走策略：先求解稀疏位置，在计算稀疏幅值。

正如上一节所述，二维图像的稀疏频谱分布并满足第三章算法所假设的情形，对于大多数有规律的自然信号，包括二维图像信号，其频谱分布都近似高斯分布。正因为如此，二维图像频谱分布也是有规律可循，通过对大量图像频谱分布的观察，可以大致定位出其中主要稀疏频谱分布。总的来说，以图 14 中得频谱图为例，设零频分量的位置为原点，水平方向和垂直方向为坐标轴切分整个频谱空间，那么幅值远大于零的显著稀疏频率分布在以原点为核心的附近一定范围，以及坐标轴两侧的有限区间内，具体范围

和区间的大小随图像类型的差异而不同。坐标轴两侧的有限区间的长度在不同图像中变化范围较小，在稀疏频谱表达算法中可以取一个常数上限，在最大的区间内恢复频谱，最后得到的稀疏频谱是由计算结果中幅值远大于零的频率组成。可以根据频谱恢复效果对区间大小进行微调。本文中，称这部分频谱可以称为零散稀疏频谱。原点附近频谱分布范围较大，这一部分的频谱分布集中，并且具体的分布范围难以确定，本节中暂时称这部分频谱为显著稀疏频谱，并称原点附近的显著稀疏频谱分布范围为高斯区，因为这部分频率都在高斯函数的截止频率内，而称在截止频率之外的区域为非高斯区。

首先需要一个策略能够区分这两类频谱，以在计算频率幅值时可以针对不同的分布使用不同的幅值计算方法。上一节中，为了调研二维图像的频谱分布，使用了高斯函数过滤频谱得到了其显著稀疏频谱的分布，这部分频谱正是我们希望恢复的，但是这样做的前提是先得到完整的二维频谱，再用高斯函数过滤。而在频谱恢复算法中，并不会计算完整的二维频谱。本节提出一种方法类似的替代方法来确定原点附近的显著稀疏频谱分布范围，即高斯区的大小。

总的计算策略仍然采用第三章的方法，通过行列采样信号的一维傅里叶变换序列来恢复二维频谱。不同的是，第三章算法假设每个频率具有相同的概率稀疏，稀疏频谱均匀分布在整个二维空间，每一行每一列具有相同的稀疏频率个数期望，所以不用判断稀疏频谱的分布范围。而本章中，图像显著稀疏频谱集中分布在原点附近的一个范围内，因此，必须能够在计算频谱幅值前知道哪里是显著稀疏频谱分布，哪里是零散稀疏频谱分布。本章假设二维图像显著稀疏频谱分布近似高斯分布，因此，可以像上一节中那样通过高斯区的大小来确定原点附近的分布，区别于上一节中使用二维高斯函数对完整二维频谱过滤，本节提出的方法实在不计算完整的二维频谱前提下，仅需操作行列采样信号的一维傅里叶变换序列来估算出高斯区，即显著稀疏频谱范围的大小。

利用高斯函数过滤二维频谱的过程可以用以下一个公式表示：

$$\hat{x}'_{u,v} = c \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x_{i,j} \omega_n^{iu} \omega_m^{jv} \cdot e^{-\frac{(u^2+v^2)}{2\sigma^2}} \quad (4.1)$$

公式右边的  $c$  是所有常数的成乘积，需要说明的是，在程序实现中，会对频谱做居中处理，所以，式中下标  $u, v$  对应这样表现形式为  $u=\{0, 1, 2, \dots, n/2, -n/2+1, -n/2+2, \dots, -1\}$ ,  $v=\{0, 1, 2, \dots, m/2, -m/2+1, -m/2+2, \dots, -1\}$ 。经过简单的变换，可以把高斯函数放入二维傅里叶变换的过程中，如下所示：

$$\begin{aligned}
\hat{x}'_{u,v} &= c \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} x_{i,j} \omega_n^{iu} \omega_m^{jv} \cdot e^{-\frac{(u^2+v^2)}{2\sigma^2}} \\
&= c \sum_{i=0}^{n-1} \sum_{j=1}^{m-1} x_{i,j} \omega_n^{iu} e^{-\frac{u^2}{2\sigma^2}} \cdot \omega_m^{jv} e^{-\frac{v^2}{2\sigma^2}} \\
&= c \sum_{i=0}^{n-1} \omega_n^{iu} e^{-\frac{v^2}{2\sigma^2}} \left( \sum_{j=0}^{m-1} x_{i,j} \omega_m^{jv} e^{-\frac{u^2}{2\sigma^2}} \right)
\end{aligned} \tag{4.2}$$

式(4.2)的意义在于，把对二维傅里叶变换后的完整频谱做高斯函数过滤转换成了在每个维度上傅里叶变换同时计算二维高斯函数对该维度的影响，如此一来的好处是，不需要等到得到完整的二维频谱时才能对其做高斯函数过滤才能知道显著稀疏频谱分布，每个维度上得傅里叶变换序列可以通过乘以一个对其没有影响的另一位维度的二维高斯函数分量来判断在该维度上二维高斯函数通过范围，式(4.2)中，如果括号内的值为零或近似零值，那么对于任何  $u$ ，其最后计算得到的二维频谱都将是零或近似零的值，即认为被高斯函数过滤。

以对列采样信号按行恢复频谱为例说明具体的判断过程：首先对列采样信号计算其一维傅里叶变换，二维频谱就是所有列傅里叶变换序同一行组成的序列的再次傅里叶变换。对所有列采样信号傅里叶序列按行分别乘以二维高斯函数按行方向的投影，乘积如果为零或近似为零，那么可以认为该行频谱在二维频谱中必定是被过滤的，也即非显著稀疏频谱。对行采样信号逐列判断是否稀疏的方法与此同理。

需要注意的是，上述步骤并不是在恢复频谱，仅仅是为针对不同分布的稀疏频谱使用不同的恢复方法的策略提供一个判断的依据，通过上述步骤判断出得显著稀疏频谱分布范围也并不是最后会被恢复的频谱，但是同样为最后恢复显著稀疏频谱提供了参考。

接下来要讨论的是如何确定恢复稀疏频谱，如上所述，采用两种策略，对零散分布的稀疏频谱，设一个默认值  $a$  表示每一行或每一列在坐标轴附近的稀疏频率的个数，并且可以认为这  $a$  个稀疏频率的位置为  $\{n/2-a/2-1, n/2-a/2+1, \dots, n/2, -n/2+1, \dots, -n/2+a/2\}$ ，一旦确定了位置，可以按式(3.6), (3.7)的方法恢复频谱，由于  $a$  是一个较小的值，因此只需要  $a$  个采样行或列参与计算。

上述恢复的频率将用来更新另一个方向的采样信号，例如如果恢复了某行第  $i$  列的频率，那么接下来所有行采样信号傅里叶序列第  $i$  列都必须减去该频率对该列的影响，以在下一次在该列方向上恢复频谱时不再恢复该频率。

对于显著稀疏频谱分布，同样接收一个较大的稀疏值  $K$ ，表示每行或列假设的稀疏

频率的数量，实际每行或列的稀疏频率数量  $d$  以按  $K$  值恢复出得频率中非零值得数量为准，但必定有  $d \leq K$ 。 $K$  值得选取参考两个标准，第一个是上文中提到的使用高斯函数过滤后每行或列剩下变换后采样信号个数  $T$ ，并且  $T \geq K$ ，原因在于经过高斯函数判断非稀疏的行列必定非稀疏。另一个标准是采样行或采样列的数量  $S \geq K$ ， $K$  必须小于  $S$  是由于对于  $S$  个采样行，最大只能恢复  $S$  列稀疏频率，当  $K$  大于  $S$  时，必须增加采样数。可以认为  $K$  是每一行或列中稀疏频率个数的一个上限，并且，由于显著稀疏频谱分布集中，每一行或列真实的稀疏频率个数与  $K$  的差值是一个较小的值  $e$ ，也即算法恢复的  $K$  个稀疏频率中有  $e$  个零值，虽然对资源稀疏频率个数估算并不完全正确，但是较小的误差值  $e$  仅能导致浪费很少的计算资源，相比使用复杂的定位算法求解精确的稀疏频率分布位置，性能上已经得到了很大提升。

总的来说，本节提出的算法是基于一种模糊定位方法，首先确定一个可以覆盖所有稀疏频率的范围，并且，这个范围尽可能小。假设这个范围内的所有频率都是稀疏的，按照公式(2.13)计算恢复所有假设范围内频率的幅值。虽然会有一些频率是非稀疏的却参与了稀疏频率恢复算法，但其经算法恢复的幅值为零，并不影响整个恢复结果的正确性。

#### 4.3.2 噪声对稀疏频谱的影响和处理方法

上一节中提出了使用模糊定位方法确定稀疏频率位置和恢复幅值的方法，实际上如果频谱是严格稀疏的，即只有幅值大于零稀疏频率和幅值等于零的稀疏频率，没有噪声频率，那么按照公式(2.13)可以顺利的恢复相应位置频率的幅值。但实际应用中，噪声总是存在的。在图像的二维频谱中，噪声的影响不可忽略，虽然可以假设大部分噪声频谱大都是高频分量，分布在离零频分量较远的空间，但是，实际应用中噪声频率也很有可能进入低频区，并且只要噪声信号存在，都会对公式(2.13)计算幅值的正确性产生或多或少的影响，如何排除噪声的影响，使得公式(2.13)式能够正确的计算得到相应位置频率的幅值是本节要讨论的话题。

排除噪声影响的难点在于，算法是要恢复表达稀疏频谱，因此，在恢复完成前，并不会知道频谱的形态，也就是说，在算法过程中，图像频谱对算法来说是透明的，因此，并不能知道哪些是噪声，也就很难排除他们的影响。

上一节提供了一种在得到频谱前估算显著稀疏频谱分布的方法，显著稀疏频谱分布之外的区域就是被高斯函数过滤掉得区域，也就是噪声分布的范围。但仅仅知道噪声分

布的范围是远远不够的，因为算法需要借助公式(2.13)类似的方法计算幅值，每一个噪声都会对公式(2.13)方程左边的输入信号值大小产生影响，如果不能去除噪声在这些输入信号上得影响，最后经过矩阵运算后的频率幅值也必然会有偏差。现在的问题就在于，如何在未得到频谱但知道频谱分布的前提下，去掉频域噪声对时域输入信号的影响。

在本文第二章介绍基于频域哈希的稀疏傅里叶算法<sup>[9]</sup>时，引入了一个卷积散列函数，通过该函数与频谱的卷积，可以将所有稀疏频率筛选出来，并在这些频率周围形成一个平坦的通过区。该函数与普通矩形选择函数相比的优势在于其尾部并不像矩形函数那样突变，因此在筛选时不会发生某些处于通过区边缘的函数被遗漏，同时，光滑的下降曲线不会给频谱带来额外的噪声。

本文中为了去除噪声对稀疏频谱恢复产生的影响，也考虑使用一个频域过滤函数，该过滤函数应该具有如下特点：首先，能够平滑的过滤高频噪声频率，其次，对通过的稀疏频率没有任何加权影响，尽量保持原来稀疏频谱的幅值不变。另外，虽然是频域过滤函数，到由于稀疏频谱恢复时并不会得到最终的频谱，所以要求对频域的过滤操作能够在空间域中找到对应等价的方法。因为该函数实际上确定了需要回复的稀疏频率范围，因此本文中称这个函数为稀疏频谱选择函数。其第一第二个特点可以用下图描述：

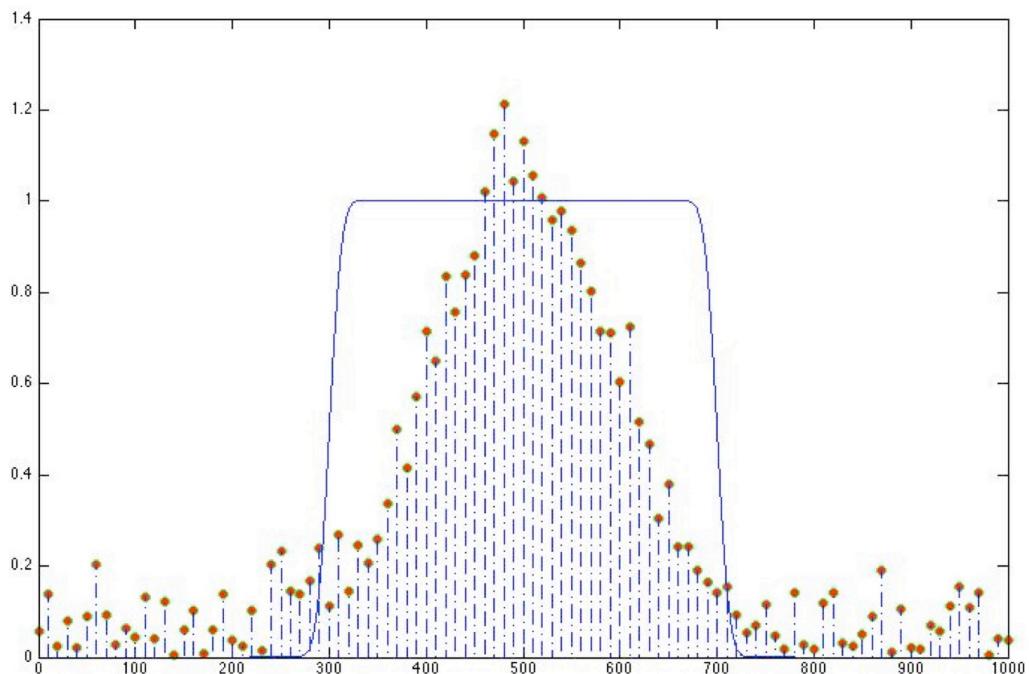


图 16 频域稀疏频率选择函数

上图函数图像是由方差 10 的高斯函数和非零区间大小为 400 的矩形函数卷积而来。对于第三点，如何在采样中间频谱构成的空间域寻找到可以替代频域过滤函数的等价操作。较容易想到的是使用空间域的卷积代替频域乘积，同时需要求得过滤函数的傅

里叶逆变换。事实上对图像的滤波操作确实分为空间域滤波和频域滤波，频域滤波就是对变换后的傅里叶频谱进行筛选，过滤掉不符合条件的频率。而空间域滤波则是频域滤波的等价操作，其最大的长处在于不用计算图像的傅里叶频谱，而直接对空间像素的计算。二维图像空间域滤波采用的方式是使用一个窗口函数，或称掩模，与空间像素矩阵进行卷积，这也验证了频域乘积等价与空域卷积的定理。掩模的性质决定了滤波的性质，较为常用的高斯滤波，就是使用了一个具有高斯核的掩模，其效果和在频域中使用高斯函数过滤频谱的效果相同。关于空间域滤波详细的知识可以参考数字图像处理，图简要说明其卷积过程。

空间域滤波可以作为频域滤波的一种等价方式，可以考虑作为上述对过滤函数第三点要求的解决办法。但是具体如何将频域的过滤函数映射到空间域，并完成相应的操作，与过滤函数的选取非常相关。例如，对于高斯函数，其通过傅里叶逆变换到空间域中仍然是一个高斯函数，区别仅在于标量系数。其简单易操作也是空间域滤波大量采用高斯函数来作低通，高通滤波的原因之一。因此，首先，过滤函数的选取要满足第一第二点要求，其次，在满足这两点的同时，应尽量使对过滤函数的第三点要求易于实现。

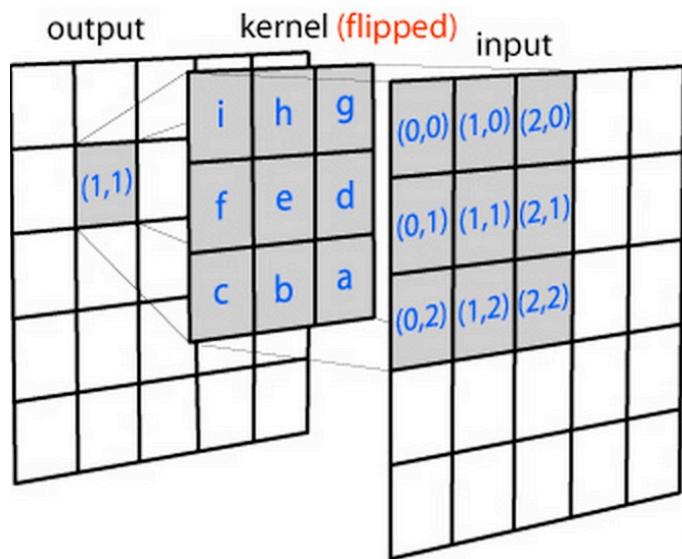


图 17 空间窗口滑动卷积方法

在本文的过滤函数的选取中，至少有两个函数能够很好的满足第一第二点要求，其一为巴特沃斯低通滤波传递函数，另一个是上文提到的，高斯函数与矩形函数的卷积，另外类似高斯径向基函数与矩形函数的卷积也能满足条件，比如切比雪夫函数与矩形函数的卷积，为了保证简单可行性，当前暂不考虑复杂类型。

虽然巴特沃斯函数能够非常好的切合第一第二点要求，并且其在频域滤波中使用广泛，但是，相比第二个函数，其傅里叶逆变换到空间域，并找到其相应的窗口掩模较为

复杂，而对于第二个函数，其傅里叶逆变换就是高斯函数和矩形函数傅里叶逆变换在空间域的乘积，也可以认为是核为高斯函数的掩模与核为矩形函数傅里叶逆变换函数的掩模的乘积，由于矩形函数也是一个简单函数，这样切换到空间域的方法较为简单可实现。

选择了过滤函数及其空间域形式，只需在对采样中间频谱进行行或列恢复前，将采样中间频谱与过滤函数空间域掩模进行卷积，其等价于在采样中间频谱对应的频域中乘积过滤函数来去除噪声信号。

## 4.4 算法设计与实现

### 并行初始化

算法首先初始化必要的算法运行信息，需要用户输入的采样行数和列数，图像的路径和规模，以及作为卷积选择函数和构成过滤函数的高斯函数的方差，这些内容构成一个算法执行计划。在获取这些参数信息后，程序在算法真正执行前开始初始化工作，包括初始化傅里叶变换系数矩阵，并计算得到其逆矩阵存储在内存中，系数矩阵的大小由采样行数和列数决定，并不受图像大小制约，例如如果设定采样行为 5，那么行傅里叶逆变换系数矩阵为：

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_m^1 & \omega_m^2 & \cdots & \omega_m^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_m^4 & \omega_m^8 & \cdots & \omega_m^{4m-4} \end{bmatrix}^{-1} \quad (4.3)$$

需要注意的  $m$  是列的长度，因为采样行用于列方向恢复稀疏频谱。在算法中当定位了稀疏频率的位置需要计算幅值时，只需从上述矩阵中选取相应列组成新的矩阵左乘采样中间频谱序列即可，不必每次需要时才去计算复杂的矩阵逆变换。另一个需要初始化的是过滤函数空间掩模，根据输入高斯函数方差，计算其与相应的矩形函数，及矩形函数的傅里叶逆变换序列，取两个序列的乘积的五个分位点的值并乘以一个标量因子作为掩模的值。最后需要初始化的是两个记录行列频谱恢复状态标识的 *BitSet*，使用一个比特位的零一来标识行或列是否还有未恢复的稀疏频率，初始化所有比特位为 1。

初始化的内容是为算法在迭代计算二维稀疏频谱时准备的，提前计算好一些必要的矩阵和变量，使得算法在迭代时可以直接使用，不需要到每次需要时重新计算。但算法采样和计算中间频谱时并不需要用到初始化的内容，因此，这两部分内容可以分开执行。本文在实现时将初始化内容在算法执行计划实例化后开始运行，甚至在读取信号之前。

因为初始化的内容并不受输入信号值得影响。但是必须在算法进入稀疏频谱恢复迭代前完成。其关系如图所示：

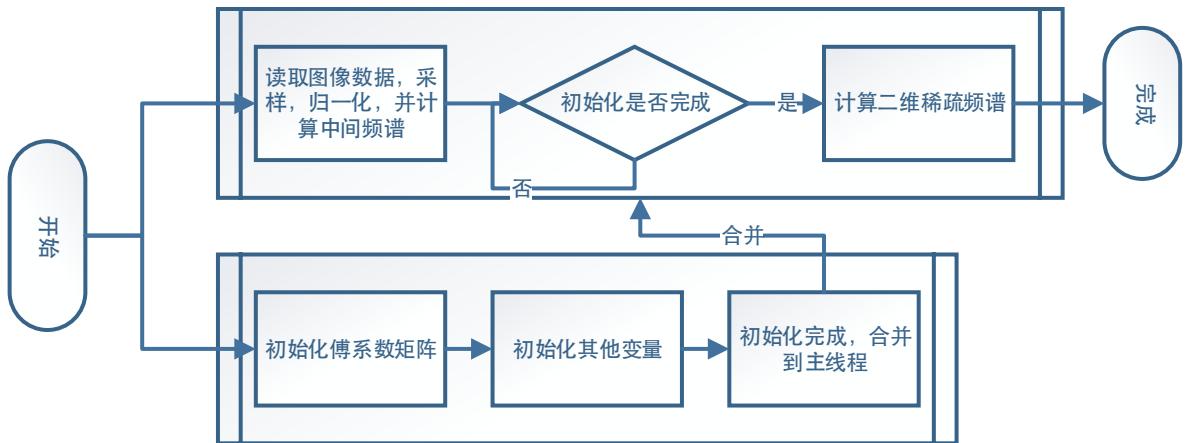


图 18 程序执行先后顺序

图 18 同时也是算法整体流程的简要描述，在具体的程序实现中，初始化过程主要由 `SparseExtract_2D::InitMatrix()` 函数完成，其详细实现参见附录伪代码描述。

### 行列采样归一化及中间频谱计算

算法仍然采用第三章中提出的行列组合采样恢复策略，第一步是采样输入信号，首先读取二维 256 灰度图像，按行在内存中存在一维序列。根据算法执行计划中设定的采样行数和采样列数，采样一维序列。对于采样行数  $r$ ，只需读取内存中一维序列的前  $rn$  个信号的值。如果采样列，则需要不断的跳跃相同数量的内存单元。另外，采样图像信号的值是一个字节 0-255 的像素值，为了方便傅里叶变换和后面算法的计算，将其归一化为 0-1 的双精度值。并且在每一个像素归一化后，对其值乘以 -1 的幂使最后的频谱居中，例如采样第  $i$  行或采样第  $j$  列时，可以用以下两式表示：

$$\begin{aligned} r_k^i &= x_{i \cdot \text{width} + k} \cdot (-1)^{i+k} / 255 \\ c_k^j &= x_{k \cdot \text{width} + j} \cdot (-1)^{j+k} / 255 \end{aligned} \quad (4.4)$$

每采样且值归一化完成一行或一列，就对其进行一维傅里叶变换以计算中间频谱，变换使用快速傅里叶变换算法，变换结果分别存储在行列两个容器中。正如上文所述，采样归一化及中间频谱计算等过程属于准备数据和预处理阶段，其并不需要等到初始化完成后执行。因此，在启动初始化线程后，主线程就进入采样和归一化程序。两个线程并行执行，并且因为不会同时操作共享存储区，不会有竞争问题。

主线程在采样并计算完所有采样中间频谱后，检查初始化进程是否已完成并返回。

只有在初始化线程执行完成后，主线程才能进入到下一步的计算迭代步骤。

行列采样图像数据，归一化及计算傅里叶变换中间频谱等过程主要由以下两个函数 *SparseExtract\_2D::SampleInRow* 和 *SparseExtract\_2D::SampleInCol*，其详细实现可参见附录伪代码描述。

## 二维零散分布稀疏频谱恢复

行采样信号中间频谱从列方向恢复稀疏频谱，列采样信号中间频谱从行方向恢复稀疏频谱。在得到两个方向的采样中间频谱后，算法正式进入迭代恢复过程。如 4.3.1 节所述，整个稀疏频谱的恢复分为两个策略，零散稀疏频谱恢复和显著稀疏频谱恢复。第一次迭代逐行读取列采样中间频谱，计算该行的二维稀疏频谱。首先，判断第  $i$  行位置是否在列高斯函数过滤的截止频率范围之内，判断方法可以简单计算高斯函数在该行位置的值即可，如果是一个近似为零的值，就认为在截止频率之外，采用零散稀疏频谱恢复策略。具体的算法逻辑是：

1. 按 4.3.1 节中零散分布稀疏频谱恢复策略，定位  $a$  个稀疏频率的位置；
2. 对该行列采样中间频谱卷积过滤函数的空间掩模；
3. 读取初始化傅里叶逆变换系数矩阵中相应的  $a$  列和从第一行开始的  $a$  行，左乘该行列采样中间频谱，得到  $a$  个稀疏频率；
4. 找到计算得到的稀疏频率的所属的列，更新该列行采样中间频谱。例如对于第  $i$  行第  $j$  列稀疏频率，更新所有行采样中间频谱多第  $j$  列的值，第  $r$  行的更新方法是：

$$S_{r,j} = S_{r,j} - y_{i,j} \omega_m^{ir} \quad (4.5)$$

$S_{r,j}$  表示行采样中间频谱的第  $r$  行第  $j$  列， $y_{i,j}$  表示刚计算得到的二维频谱的第  $i$  行第  $j$  列频率幅值， $m$  为二维输入信号的列长。

5. 更新行恢复标识，将记录行方向稀疏恢复状态的 *BitSet* 的第  $i$  位置为 0；

以上是第一次迭代读取列采样中间频谱从行方向恢复稀疏频谱。第二次迭代读取行采样中间频谱，从列方向恢复稀疏频谱，其主要算法流程第一次迭代相仿，注意的是算法过程中用到的过滤判别函数，傅里叶逆变换系数矩阵都是列方向的。

对于行列非高斯区零散稀疏频率的计算和迭代过程主要以下函数实现，各函数的具体功能描述如下，均省略参数，详细的实现过程可以参加附录中相应的函数伪代码，以下是伪代码中计算零散稀疏频谱部分主要实现函数的功能描述：

表 2 零散稀疏频谱计算主要函数

<i>RecoverByRowIterFst()</i>	按行迭代列采样中间频谱，并判断该行是否在行高斯区；
<i>RecoverByColIterFst()</i>	按列迭代行采样中间频谱，并判断该列是否在列高斯区；
<i>RecoverHighRowFre()</i>	计算一行非高斯区零散稀疏频率，并更新行采样中间频谱；
<i>RecoverHighColFre()</i>	计算一列非高斯区零散稀疏频率，并更新列采样中间频谱；

### 二维显著稀疏频谱恢复

经过了上述一次行方向和一次列方向的迭代，算法基本上恢复了位于坐标轴两侧的零散分布稀疏频谱，并更新了其对行列采样中间频谱的影响。接下来需要恢复的是二维稀疏频谱的主要部分：显著稀疏频谱。仍然从行方向将列采样中间频谱作为算法的输入，以下是其主要的算法逻辑：

1. 检查列 *BitSet* 中不为零比特位的个数 *K* 是否小于等于当前采样列的个数 *S*，如是，继续下面步骤。否则，提示更加列采样数至 *S=K*，算法退出；
2. 读取行 *BitSet* 的第 *i* 位，检查其值是否为 1，如是，读取列采样中间频谱的第 *i* 行，如否，则跳过该行，并读取下一位继续检查；
3. 更新所读取的第 *i* 行列采样中间频谱为其与过滤函数空间掩模的卷积；
4. 读取傅里叶逆变换系数矩阵，此时不需要删减行列，左乘该行列采样中间频谱；计算其结果作为二维系数频谱的第 *i* 行系数频率；
5. 将行 *BitSet* 第 *i* 位置为零，检查整个 *BitSet* 转换为整数的值是否为零，如否，*i=i+1*，重复以上步骤，否则退出循环；

上述循环成功退出后，显著稀疏频谱的恢复基本完成。并不需要在读取行采样中间频谱从列方向在进行一次迭代，因为这部分属于行列方向都能够恢复的频谱区域。不在进列方向迭代的好处是可以减少行采样数目，减少输入规模，同时少一次迭代，提升算法的性能。但缺点当然是算法结果相比多一次迭代稍逊一筹。如果对准确度要求较高，可以再进行列方向的迭代，同时行采样中间频谱必须有足够的采样数目。最后取行列两次迭代恢复结果的均值作为最终稀疏频率的幅值。列方向的迭代实际可以看做是对行方向迭代的一个验证，可以更好的对抗噪声产生的影响，本文提供一个模式参数选择是否需要加入这样的验证迭代。列方向迭代的算法逻辑与行反向类似，同样需要注意的是列方向迭代读取的是行采样中间频谱，以及其他参与计算的变量的方向。

对高斯区显著稀疏频率的计算主要由 *SparseExtract\_2D:: RecoverByRowMain()* 函数和 *SparseExtract\_2D:: RecoverByColMain()* 函数实现，具体可见附录伪代码描述。

## 程序详细实现流程

图 20 是算法详细流程图，采用 UML 活动图描述了二维图像稀疏频谱表达算法的整个过程。详细设计伪代码和部分程序见附录；程序主要实现类和其关键函数如图所示，其中每部分函数的主要功能见上文分模块描述，其详细实现过程可见附录伪代码。由于伪代码的局限性，函数的参数和返回值可能有所不同。

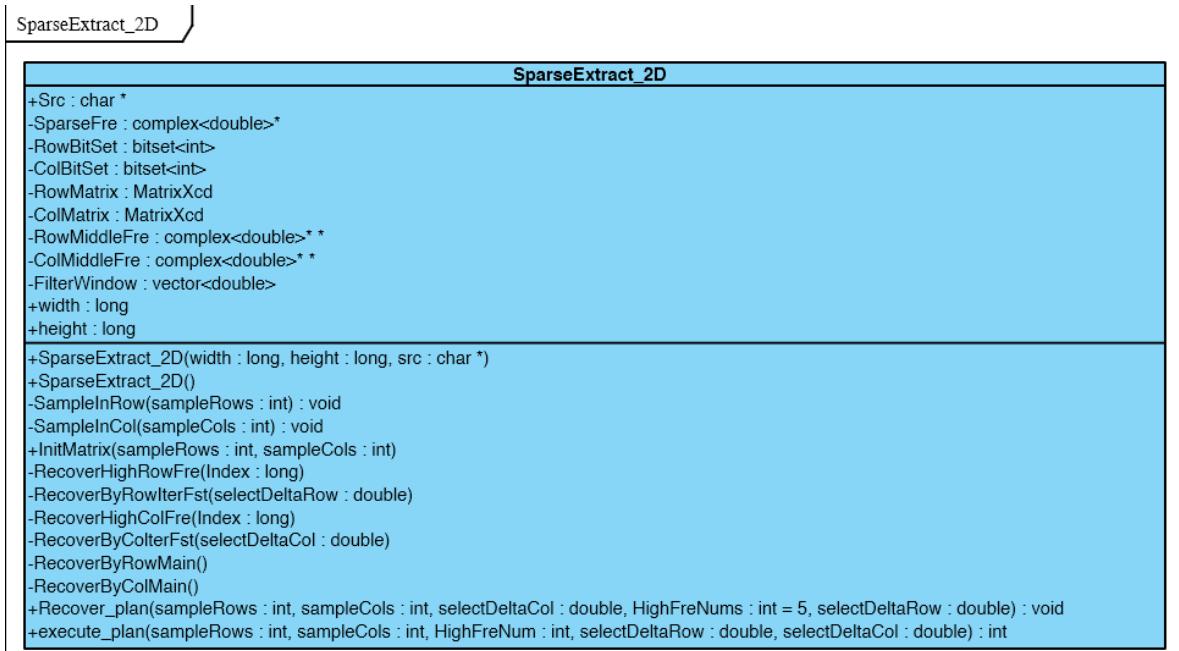


图 19 算法主要实现类

## 4.5 算法分析及优化

本章所提出的二维稀疏频谱表达算法，是针对实际应用的一类算法，区别于第三章中提到了一些对输入二维离散信号具有较强假设约束的算法，本章算法适合用于二维数字图像稀疏频谱的提取和表达，下面将从性能和复杂度等方面对算法进行分析和评估。

首先，假设  $n \times n$  输入二维离散信号傅里叶频谱有  $K$  个显著稀疏频率，并设其主要分布在零频分量附近的一个区间范围内，其大小为  $k_1 \times k_2$ ,  $k_1 > k_2$  并且有  $k_1 k_2 = CK$ , 其中  $C$  是一个常系数。要计算这样的频谱，普通快速二维傅里叶算法采用逐维拆解组合变换的方式，需要读取所有  $n \times n$  个信号。对于本章算法，如果是在无验证模式下，只需采样  $k_1$  个采样列，与  $a$  个采样行， $a$  是一个较小的标量。因此需要读取的二维离散信号个数为  $O(nk_1)$ ，在加入验证模式的算法中，需要读取  $O((k_1+k_2)n)$  个信号。例如对于  $2048 \times 2048$  的图像，假设其具有一个较大的稀疏度  $K=512 \times 256$ ，本章算法最多只需采样输入信号约三分之一的信号，比快速傅里叶变换少计算两百八十万次输入像素。

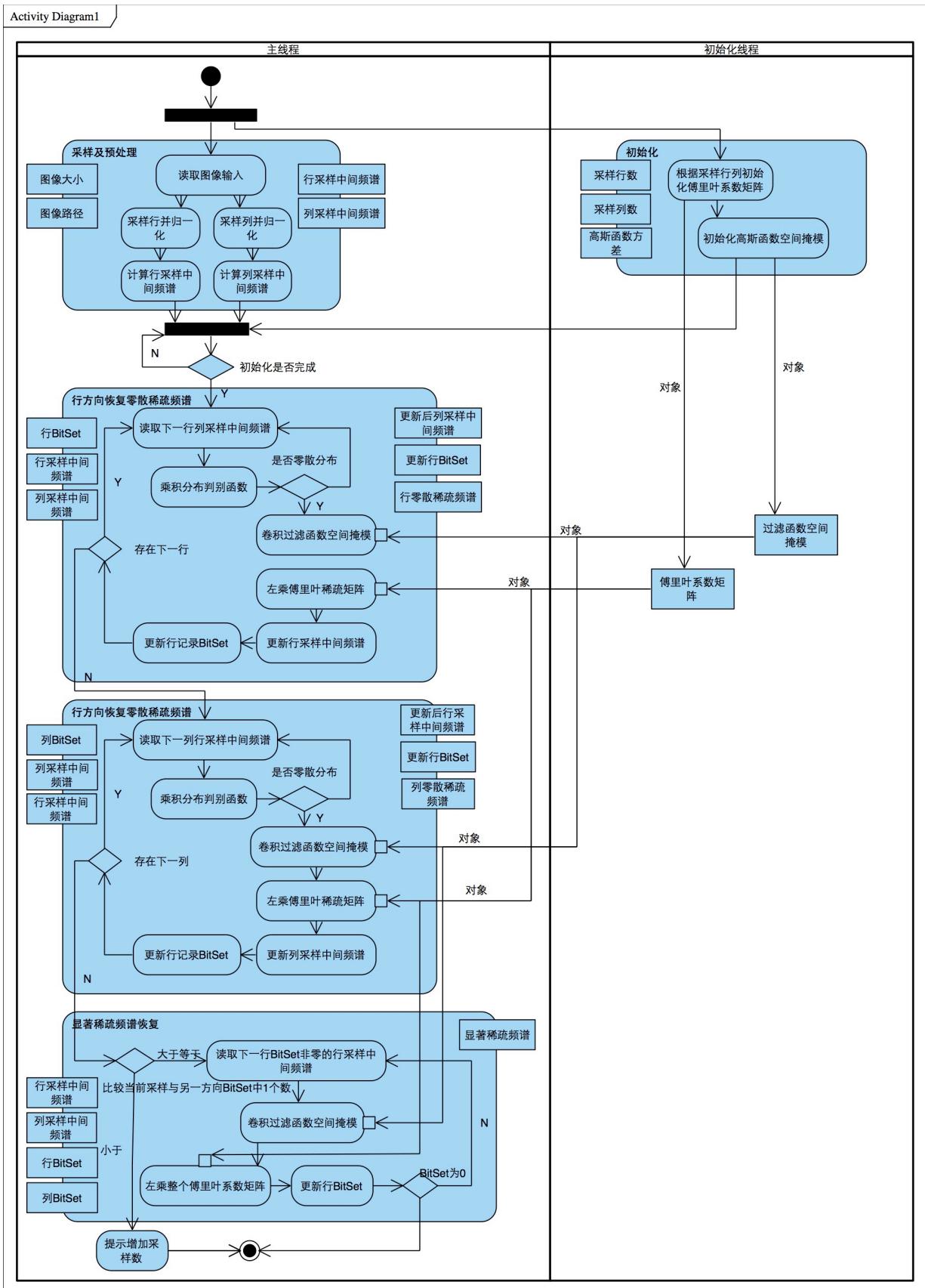


图 20 二维稀疏频谱表达算法流程

算法的时间消耗主要表现在：

- 对输入二维离散信号的采样，归一化和对采样信号的快速傅里叶变换。

由于二维离散信号在内存中都是按行存储成一维形式，因此，采样行信号要比采样列信号简单，原因在于采样行信号属于连续内存访问，可以很好的利用缓存局部性原理，每次读入行首的一个信号，其后一定范围内的信号都被加载入缓存，之后行信号可以直接从缓存中读取。而采样列信号属于跳跃式内存访问，每采样一个列信号都需要跳过一行信号所占的内存单元，由于缓存假设用户是连续读取的，只会加载当前读取内容的临近区域，所以下一个采样列信号并不在缓存中，因此程序只能每次都只能从内存中读取需要的列信号。虽然一次内存访问和一次缓存访问只有千分之一毫秒的差别，但是如果访问的次数很多时间性能就会明显下降。这在二维快速傅里叶变换中表现明显，因为其在做列变换时需要大量的跳跃式访问。本文中对此的优化是，在无验证模式时，对显著稀疏频谱只做行方向的迭代，从列方向恢复频谱。

不管是否能有效利用缓存，都假设成对内存的访问，由于访问内存任何位置的时间是相同的，采样的时间也是线性的，采样的同时进行归一化和频谱居中，因此整个采样及预处理操作的时间复杂度为采样信号数量的常数倍，无验证模式时间复杂度为  $O(k_1n)$ ，加验证模式时间复杂度为  $O((k_1+k_2)n)$ 。

一维快速傅里叶变换的时间复杂度为  $O(N \log N)$ ，因此，无验证模式计算中间频谱的时间复杂度为  $O(k_1 n \log n)$ ，加验证模式为  $O((k_1+k_2)n \log n)$ 。

总的来说，这一部分的时间消耗为  $O(k_1 n \log n)$  或  $O((k_1+k_2)n \log n)$ ，分别对应无验证和有验证模式。由于  $k_1 > k_2$ ，因此可以总的认为是  $O(k_1 n \log n)$ 。

## 2. 零散分布稀疏频谱恢复行列两次迭代过程：

以逐行读取列采样中间频谱从行方向恢复稀疏频谱为例，迭代需遍历列采样中间频谱的所有行，对每一行需计算其与列选择函数的乘积，来确定是否在本次迭代中恢复该行频谱，这个过程耗时  $O(k_1)$ 。对于满足条件的行，会根据零散分布稀疏频谱恢复策略，计算其相应行的稀疏频率，先是根过滤选择函数空间掩模的卷积，因为掩模大小是一个常数，所以耗时  $O(k_1)$ 。再是读取傅里叶逆变换稀疏矩阵并乘积列采样中间频谱的  $a$  列，由于  $a$  是一个较小的常数，且系数矩阵只需读取一次，所以这部分时间消耗为常数  $C$ 。

最后是更新行采样中间频谱，每次只需更新  $a$  列，每列更新的时间消耗为  $O(k_2)$ 。

所有满足条件的列采样中间频谱行都需要更新，因此更新行采样中间频谱总消耗  $O((n-k_1)k_2)$ 。

总的来说，每一行时间消耗  $O(k_1)$ ，加上必要的更新操作，整个行反向的迭代时间消耗为  $O(k_1 n)$ 。列方向的迭代同理，整个过程的时间复杂度  $O(k_1 n)$ 。

### 3. 显著稀疏频谱恢复过程：

这一过程中主要时间消耗在于对于列 *BitSet* 中状态为 1 的行计算相应二维频谱行的稀疏频率。对于每一行，需要先卷积过滤函数空间掩模，如上所述，时间消耗为  $O(k_1)$ ，然后是矩阵乘法，这里选择的是整个系数矩阵，因此时间消耗不再是一个常数。对于  $k_2 \times k_1$  的矩阵左成  $k_1$  维向量，其时间复杂度为  $O(k_1 k_2)$ ，共计算  $k_1$  行，所以这部分的时间复杂度为  $O(k_1^2 k_2)$ ，如果加入验证迭代，验证迭代的时间复杂度与此相仿。

综上所述，算法的时间复杂度为  $O(k_1 n \log n + k_1^2 k_2)$ 。

## 4.6 本章小结

本章从二维图像稀疏频谱特性出发，深入分析了二维图像稀疏表达的两个难点，并探索出了合理的解决办法。结合第三章组合行列稀疏频谱恢复算法思想，针对二维图像频谱的特性，提出一种新的二维图像稀疏频谱表达算法，并详细描述了算法逻辑，分析了算法的性能。

## 第五章 二维图像稀疏频谱表达实验及应用验证

第四章中提出了一种针对二维图像稀疏频谱的表达算法，对详细分析了算法的性能和时间复杂度。本章中主要是对该算法的实验验证，及在现有两个应用中的使用描述。

### 5.1 二维图像稀疏频谱表达算法实验分析

本节中将主要分析二维图像稀疏频谱表达算法在不同类型，不同大小图像上的应用。首先，根据输入图像，算法采样归一化二维图像离散信号，计算并输出其二维稀疏频谱，为方便分析和显示，实验中输出算法计算得到了二维稀疏频谱的对数图，以及该二维稀疏频谱通过傅里叶逆变换得到的二维空间图像，通过比较原空间图像和稀疏频谱表达的空间图像，来评价算法的得到的二维稀疏频谱表达的正确性。同时，对同一输入，比较在不同稀疏度下的比较算法得到的稀疏频谱的二维空间图像。

#### 5.1.1 图像质量评价方法

评价一个二维图像的质量可以从两个方面考虑，即目前主流的图像质量评价方法，从主观和客观两个角度。主观图像质量评价根据实验人员的主观感知对图像进行分析和理解，因此因人而异，不能够作为一个稳定的评价指标。而客观图像质量评价根据模型给出量化指标，通常的做法是通过比较评价图像和源图像的各种偏差来估算评价图像的好坏。两个经典的评价方法为最小均方误差( $MSE$ )<sup>[22]</sup>及峰值信噪比( $PSNR$ )<sup>[22,23]</sup>。

最小均方误差是一个常用的误差计算方法，在一些机器学习算法中将其作为损失函数。其作为图像评价方法的函数形式为：

$$R_{MSE} = \frac{1}{mn} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (x'_{i,j} - x_{i,j})^2 \quad (5.1)$$

其中， $n, m$  分别为二维图像的行宽和列高。 $x'$  为评价图像的二维像素矩阵， $x$  为源图像二维像素矩阵。需要说明的是，虽然  $MSE$  作为一种常用的误差计算方法，但却很少在图像质量中使用，究其原因主要在于其评价结果与人的主观感觉相差较大，相关性较低。本文中将偏向使用  $PSNR$  方法，其函数计算形式如下：

$$R_{PSNR} = 10 \lg \frac{Q^2 \cdot mn}{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (x'_{i,j} - x_{i,j})^2} (dB) \quad (5.2)$$

式中  $Q$  是图像量化灰度级数。由于自然图像具有特定的结构，每个邻近像素点之间

存在很强的从属关系，正是这些从属关系使不同的二维图像展现出不同的信息。基于这些原理提出了基于结构失真的图像质量评价方法，简称结构相似度评价(*SSIM*)<sup>[23]</sup>，直接评价图像信号的结构相似性。它将传统图像结构信息中的亮度，对比度，及图像中得物体结构分别用均值，标准差，协方差来替代，其公式形式为：

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\delta_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\delta_x^2 + \delta_y^2 + C_2)} \quad (5.3)$$

式中， $\mu_x, \mu_y$  分别是源图像和评价图像的像素均值。 $\delta_x, \delta_y$  分别是源图像和评价图像的像素值方差，最后  $\delta_{xy}$  是源图像和评价图像的协方差。具体的技术方法和图像空间卷积类似，使用一个  $8 \times 8$  窗口，从左至右，自顶向下在二维图像上滑动。每滑动一次根据上面公式计算一次窗口与图像重叠部分的 *SSIM* 值，最后去所有滑动计算所得 *SSIM* 值得均值作为最后的 *SSIM* 值。*SSIM* 值得取值区间是(0,1)，值越高表示所评价的图像的质量越高。

### 5.1.2 实验及结果分析

实验 1：对输入图像大小为  $512 \times 512$  的人物脸部图像，分别在不同的采样行列，以及不同的选择函数方差下使用第四章二维图像稀疏频谱表达算法的实验。以下分别是实验主要参数列表和相应的实验结果分析。



图 21 输入图像 lena 及其 FFTW 频谱对数灰度图

首先，上图是经 *FFTW* 变换后得到的频谱对数灰度图，统计 *FFTW* 计算频谱总数为 262144。计算模式为 *FFTW\_ESTIMATE*，共计耗时 12ms，内存占用 2056KB。*FFTW* 在计算规模较小的图像傅里叶变换时确实具有非常好的性能，源于其对程序实现的优化，

例如像内存对齐策略等。下表是利用稀疏频谱表达算法计算图 21 所示图像的稀疏频谱的主要参数说明：

表 3 实验对比 1 无验证稀疏频谱表达主要参数

图像规模	图像类型	采样行	采样列	选择函数方差	列验证
512×512	人物头像	10	202	80	无

如第四章中所述，表 3 中的参数表明，本次试验对比采用稀疏频谱表达算法计算图像稀疏频谱时，主要从行方向计算显著稀疏频谱，所以采样了较多列，并且，并没有同时从列方向计算显著稀疏频谱，来增强行方向计算的结果。在本章其他试验中，处于时间性能的考虑，默认都不加入列方向计算稀疏频谱过程。

下图是在表 2 所示参数下的稀疏表达频谱及其对应的空间图像：

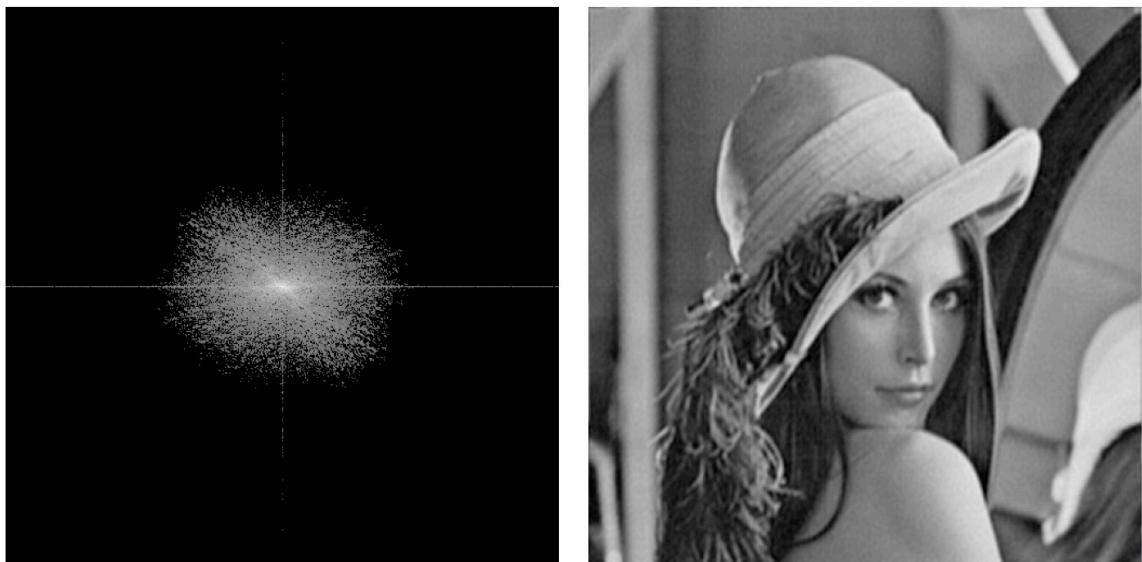


图 22 稀疏度为 18883 的频谱表达图像  $PSNR=26.8615dB, SSIM=0.9314$

实验对比 1 共采样了 202 列，根据这些采样列的快速傅里叶变换中间频谱，从行方向恢复二维稀疏频谱。因不采样用从列方向计算稀疏频谱来验证并增强行方向计算结果的方法，所以只采样 10 行，主要用于帮助计算非高斯区轴附近的稀疏频谱。输入图像规模较小，整个算法运行耗时  $231ms$ ，相比 FFTW 要慢一些。最后经稀疏频谱还原的二维空间图像相比源图的峰值信噪比  $PSNR$  为  $26.8615dB$ ，结构相似度  $SSIM$  为 0.9314，表明了稀疏频谱表达的图像和源图是近似相同的，可以用稀疏频谱表达的方式等价表示源图像，相比完整傅里叶频谱，稀疏表达的方式只用了完整频谱 7% 的频率。另外，在本章之外的实验中加入了从列方向计算频谱的模式，采样更多行，得到的稀疏频谱表达效果和本实验表达效果相差较小。

以下是其他几组不同参数下的实验和结果，由于输入图像规模较小，运行时间性能

上差距  $FFT W$  较大，但实验结果仍然可以验证稀疏频谱表达图像的准确性，表 4 中最后一行是加入列验证模式的稀疏频谱表达：

表 4 实验对比 2 不同参数下稀疏频谱表达

采样行	采样列	选择方差	内存使用(KB)	耗时(ms)	稀疏度	PSNR(dB)	SSIM
10	143	50	612	208	10076	24.9952	0.8698
10	151	55	644	212	10924	26.0088	0.8856
10	157	60	668	215	11852	26.0501	0.8935
10	166	70	704	221	13610	25.9540	0.9058
166	166	70	1328	276	13610	26.4214	0.9067

图 23 分别是上表中第一行和第四行参数实验的稀疏频谱和其表达的空间图像，限于篇幅，此处列出表 4 中的部分结果说明表达效果：

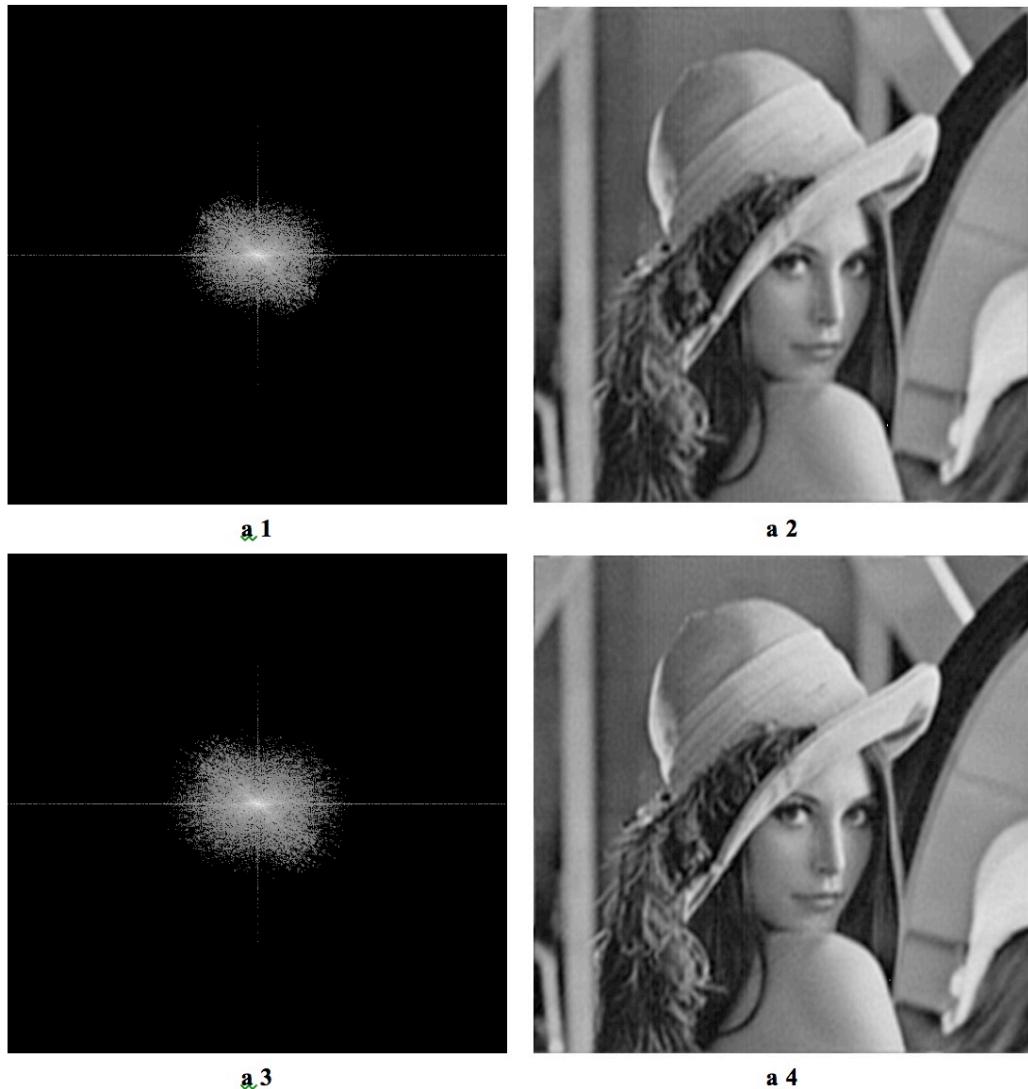


图 23 不同稀疏度下稀疏频谱表达效果

表 4 中得选择函数方差和采样行列成正线性关系，由第四章中对稀疏频谱计算过程的描述可知，选择函数用于确定高斯区的大小，以此判断稀疏频谱的分布，并采用不同的计算策略。选择函数是一个类高斯函数，其方差越大，高斯区的范围越大，显著稀疏频谱分布范围越大，因此需要采样更多的行和列。图 23(a1), (a2)为稀疏度为 10076 时的稀疏频谱及其表达的空间图像，图 23(a3), (a4)稀疏度为 13610 的稀疏频谱及其表达的空间图像。分别用了完整傅里叶频谱的 3.8% 和 5.1% 的频率来表达频域和空间域。其相应的峰值信噪比和结构相似度见表 4 第一行和第四行。可见越低的稀疏度会是稀疏表达图像的能力变差，图 23(a2)是用 3.8% 的频率稀疏表达，虽然清晰度和零度下降了，但是仍然可以作为人脸识别的输入，且并不会降低识别的正确性。

**实验 2：**对输入大小为  $2448 \times 3264$  像素的场景图像，分别在不同采样值及选择函数方差条件下，计算其二维图像稀疏频谱的实验，分析实验结果。图 24 是输入图像及经过 FFTW 变换计算得到对数频谱的灰度图像：



图 24 图像 school 及其 FFTW 频谱对数图

上图右图为左图空间图像经 FFTW 变换后得到的频谱图，统计 FFTW 共计算傅里叶变换约 799 万个频率，共耗时约 486ms，占用内存 62424KB。

以下是实验 2 在不同采样值和不同稀疏度下二维图像稀疏频谱表达算法的输出结果对比，下表是主要的参数和量化指标对比：

表 5 实验 2 图像 school 在不同稀疏度下得稀疏频谱表达对比

采样行	采样列	选择方差	内存使用(KB)	耗时 ms	稀疏度	PNSR(dB)	SSIM
25	658	500	10200	667	281321	10.8118	0.9895
25	711	600	14235	712	351435	11.6137	0.9935
25	817	700	16262	785	381801	12.2224	0.9946
25	1061	800	20929	934	419013	12.5983	0.9931

表 5 中列出了以 100 为间隔的选择函数方差，逐步扩大高斯区的大小，增加采样列的数量，同样，为了保证时间性能，不加入从列方向计算稀疏频谱的过程。表 4 中所有实验结果得到的稀疏频谱，其表达的空间图像和源图像的结构相似度都在 0.98 以上，充分验证了稀疏频谱表达方法的正确性。下图 25 分别是稀疏度为 281321 和稀疏度为 419013 时的稀疏频谱表达效果，对应表 5 中的第一行和第四行：

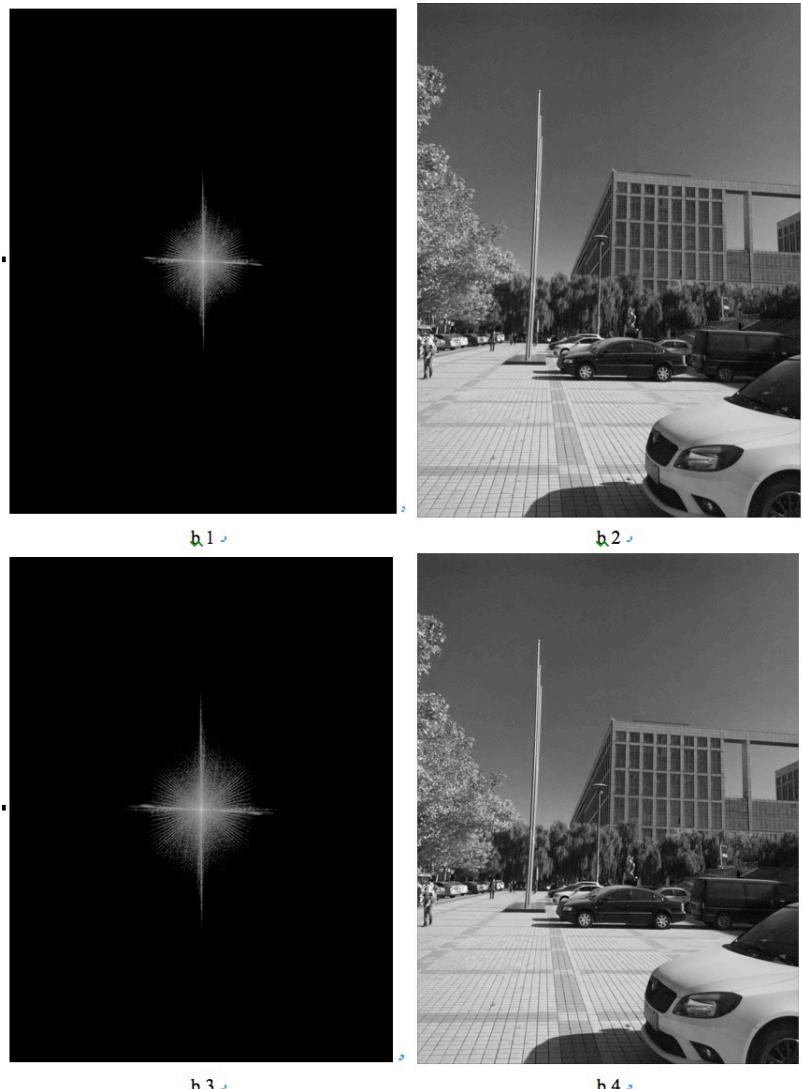


图 25 场景图像在不同稀疏度下得表达效果

相比  $FFT W$  计算得到了完整的傅里叶频谱共约 799 万个频率, 图 25(b3)中二维图像稀疏频谱表达算法值计算了约 40 万个频率, 约完整频谱的 5%, 图 25(b2),(b4)是在不同稀疏度下得稀疏频谱表达效果, 其中, 图 25(b2)是用完整频谱 3.5% 的频率表达的空间图像, 图 25(b4)是用完整频谱约 5.2% 的频率表达的空间图像, 两个稀疏频谱表达结果的空间图像与源图像的结构相似度都比较高, 均在 0.98 以上, 下图是对比图 24(b4)与源图的结构差异描述, 图中颜色越深的部分表示结构差异稍大的部分, 为方便显示, 对结构差异图调整了对比度, 如下图所示:

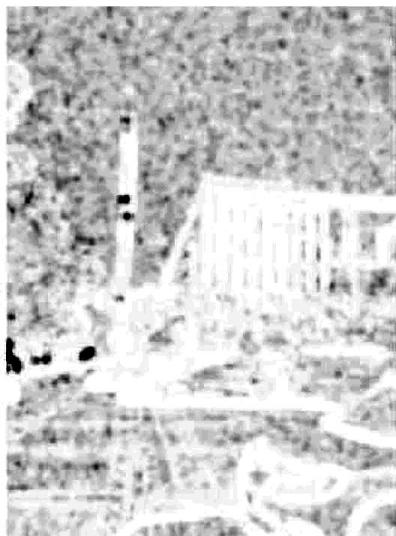


图 26 结构相似与差异部分显示

实验 3: 对输入大小为  $4578 \times 3446$  的图像, 在不同采样值和不同选择函数方差的条件下, 利用二维稀疏频谱表达算法计算图像稀疏频谱, 对比稀疏频谱的表达, 并分析实验结果。图 27 是源输入图像和其经  $FFT W$  计算傅里叶变换后得到对数频谱灰度图:



图 27 停车场场景大图及其  $FFT W$  频谱对数图

$FFT W$  计算二维离散傅里叶变换共约 1570 万个频率, 耗时约  $1786ms$ , 占用内存  $121MB$ , 以下是在不同实验参数下得稀疏频谱表达实验, 表 6 是主要的实验参数和量化对比指标:

表 6 实验 3 实验主要输入参数及输出

采样行	采样列	选择方差	内存使用(MB)	耗时 ms	稀疏度	PSNR(dB)	SSIM
50	813	550	29.71	1094	581385	15.8223	0.9765
50	995	650	36.06	1198	609275	16.9875	0.9821
50	1071	750	38.72	1476	622961	17.8702	0.9876
50	1203	850	43.33	1743	652963	17.9768	0.9901

表 6 所示的实验中，仍然主要从行方向计算二维稀疏频谱。表 6 中选择函数方差以 50 为间隔列出了五组对比实验，采样列和采样行共计约采样完整二维信号的 20%，实验结果得到的稀疏频谱中频率的个数约为完整傅里叶频谱的 3% 到 5%，并且此时算法的时间性能要好于 FFTW。下图 28 是表 6 中第一行和第四行实验结果得到的稀疏频谱，及稀疏频谱表达的空间图像：

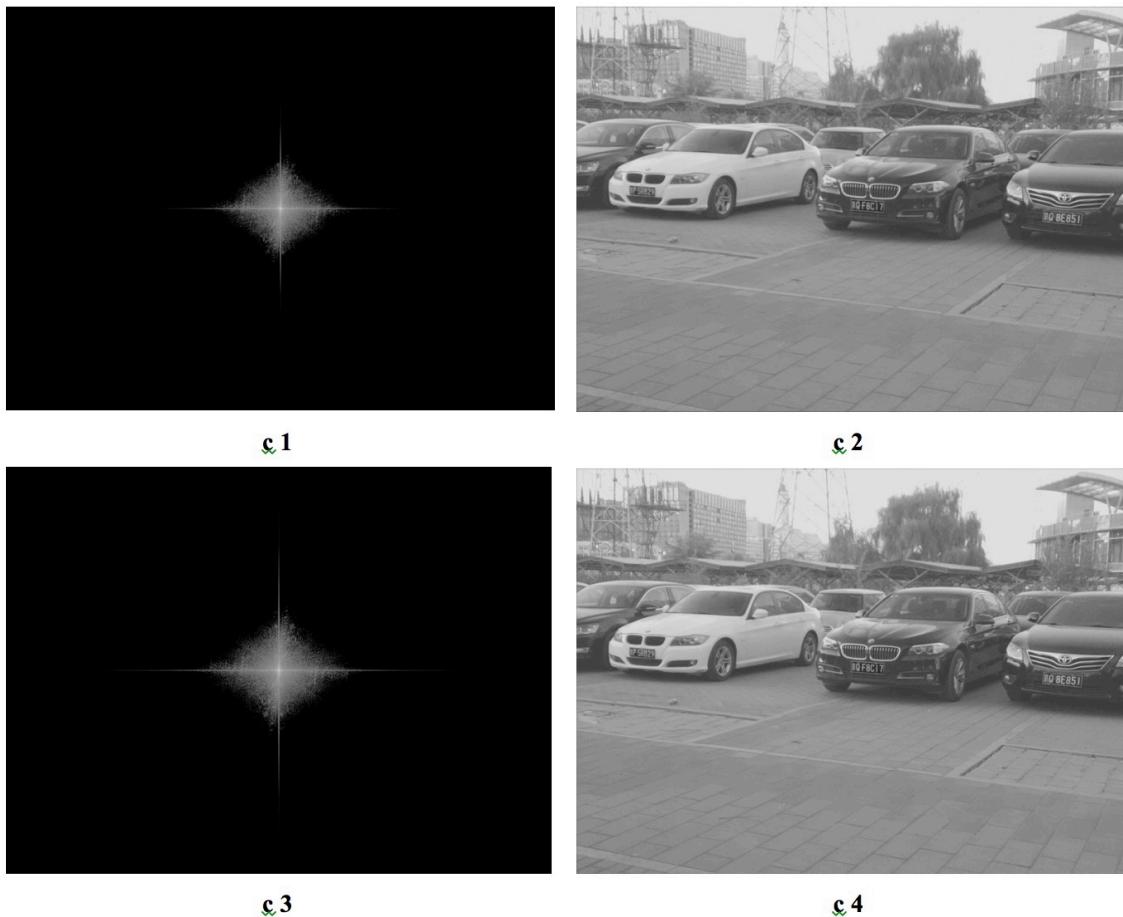


图 28 停车场图像稀疏频谱表达对比

上图 28 中，图 28(c1),(c2)是 581385 个频率构成的稀疏频谱及其对应的空间表达图像，由于图像较大，较难直接对比源图观察效果，根据表 6 中的量化指标，该稀疏频谱表达使用了 3.5% 的频率，其表达的空间图像和源图的结构相似度在 0.97 以上，而图

28(c3),c(4)使用了约完整频谱 4.1%的频率来表达空间图像，其与源图的结构相似度在 0.98 以上，具体见表 6 第一行和第四行。为了对比细节，下图对其中图 28 中车牌放大后的稀疏表达图像对比：

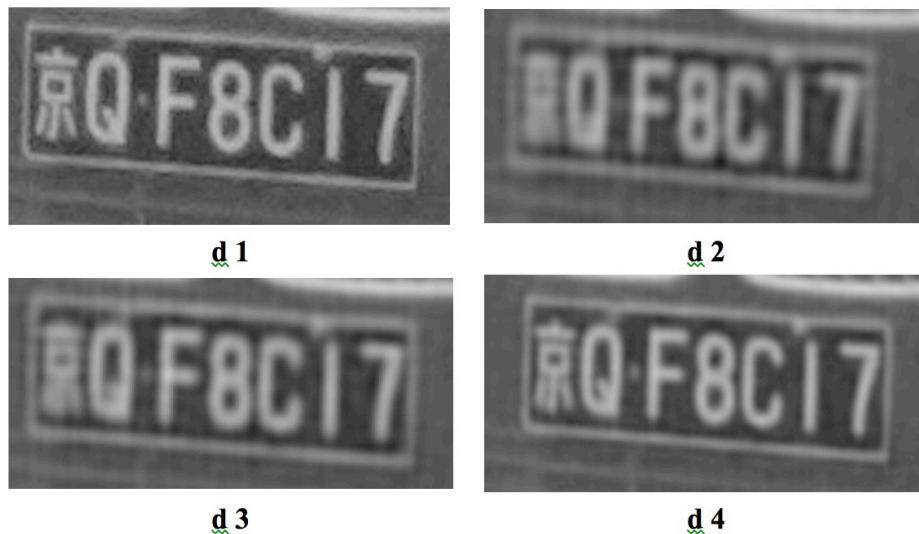


图 29 车牌大图稀疏表达对比

上图是从上面实验计算得到的稀疏频谱所表达的图像中截取的细节图像对比，其中，图 29(d1)为源图像中得车牌截图。图 29(d2)为表 6 中第一行实验参数下计算了约 58 万个稀疏频率，约完整频谱的 3.5%，表达的图像细节，对应大图为图 28(c2)。图 29(d3)为表 6 中第四行参数下计算得到约 64 万个稀疏频率，约完整频谱的 4.1%表达的停车场图像中的车牌截图，对应大图为图 28(c4)。图 29(d4)为 148 万个稀疏频率，约源图像频谱的 9.3%的稀疏频谱表达的停车场图像中得车牌截图。通过观察不同稀疏频谱表达的车牌截图表明，更多得稀疏频率可以使得稀疏频谱表达的二维图像的细节更为清晰，如果用于中对细节要求较高，那么应考虑更大的稀疏度，图 29(d4)中，9.3%的稀疏频谱对细节的表达效果已经和源图相差无几，并且，仔细观察可以发现，源图中得纹理噪声也到了较好的平滑。

## 5.2 基于稀疏频谱表达的地质图像频域处理

本文研究起源于“地质异常体分割和可视化算法库”项目，其主要运用在地质图像的预处理和分割部分。作为断层切面图像快速傅里叶变换和低通滤波算法的扩展，提高频谱获取和频谱表达能力，简化傅里叶频谱分析。

在地质异常体图像的分析和处理中，会遇到很多傅里叶频谱分析，包括一些预处理的增强和去噪。简单的图像平滑方法可以使用空间高斯核卷积平滑，或使用快速傅里叶

变换计算图像频谱，并在频域中进行低通滤波。本文对图像稀疏频谱的表达方法在快速计算图像频谱的同时，也可以作为一种平滑方法。相比空间卷积平滑，图像稀疏频谱表达方法平滑了图像的同时，得到了图像的稀疏频谱，其可以作为后续频域处理的输入，像高通，带通滤波等。

对地质断层图像的稀疏频谱表达，以大小为  $935 \times 746$  大小的断层图像为例，观察稀疏频谱的表达效果。

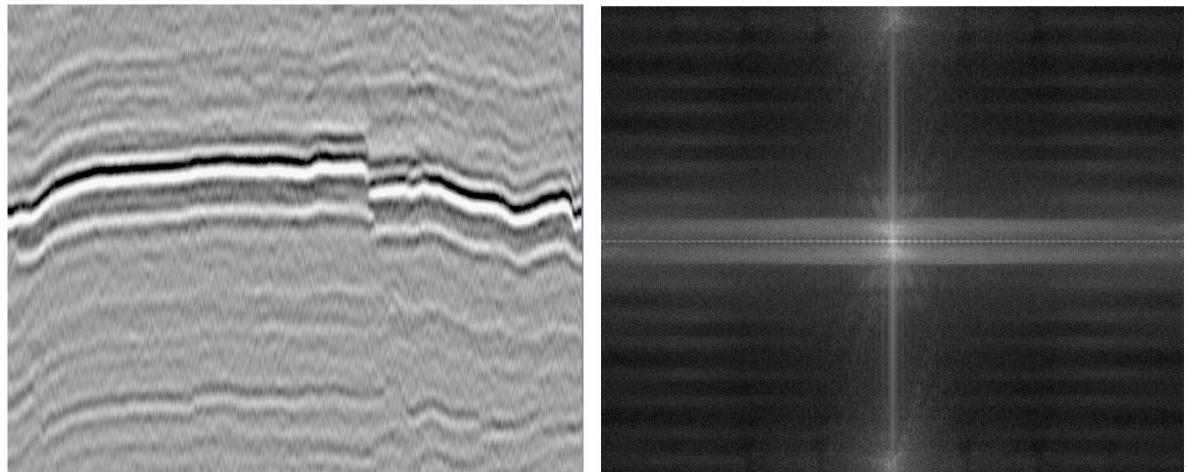


图 30 地质断层图像及其 FFTW 频谱对数图

图 30 的左边是地质断层图像灰度图。右边是其傅里叶频谱对数图。共计约 70 万个频率。可以看到，源图像中具有较强的纹理噪声，断层边缘不清晰，其傅里叶频谱图中可以观察到高频区间的噪声频率。但是使用简单低通滤波过滤频域，又会丢失非高斯区轴附近的频率信息。

下图是采样 80 列，65 行断层图像信号，计算得到其稀疏频谱，及通过该稀疏频谱表达的断层图像：

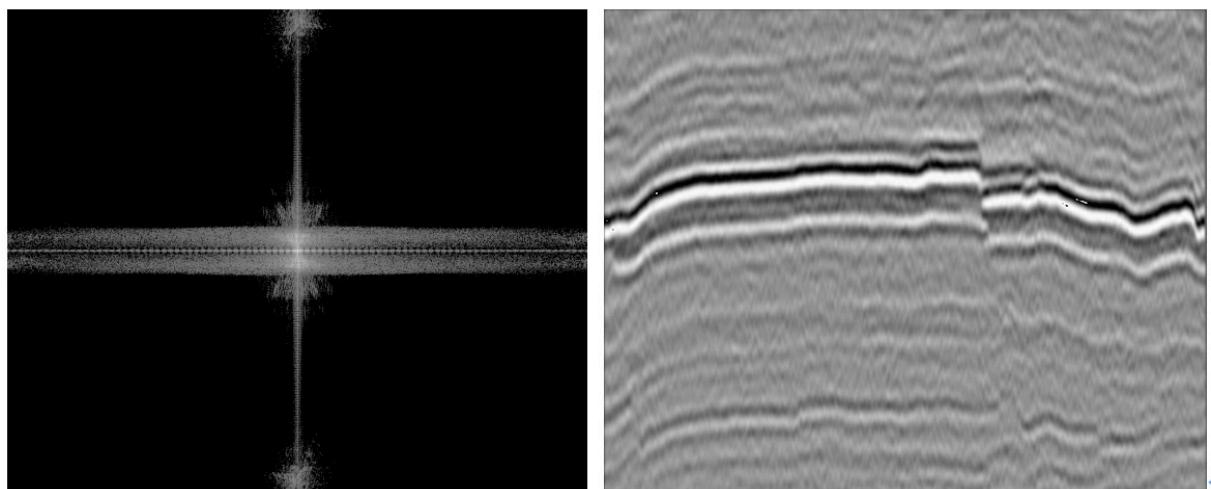


图 31 稀疏频谱表达地质断层图像

上图中，稀疏频谱共包括 90864 个频率，约源图像完整频谱的 13%，通过逆变换得到其稀疏表达的空间图像，与原图相比  $PSNR$  值为 22.3234， $SSIM$  值为 0.9744，下图是对断层局部图像放大后的对比效果：

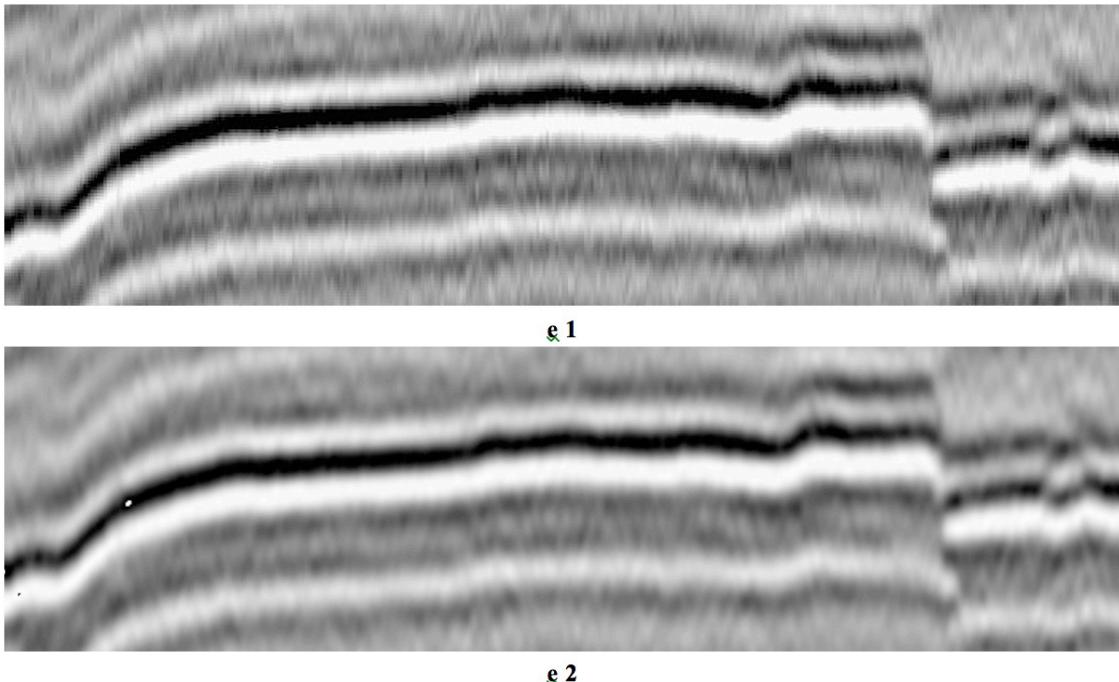


图 32 地质断层图像稀疏频谱表达断层对比

图 32(e1)是源地质图像的断层放大图，图 32(e2)是由二维稀疏频谱表达算法计算得到的源图像完整频谱 13% 的频率表达的断层图像。对比的结果可以发现，源图像的断层图像受噪声影响，并不清晰。相比稀疏频谱表达的断层图像，噪声得到了平滑。原因在于，二维稀疏频谱表达算法在计算图像稀疏频谱时，只恢复计算主要的稀疏频谱，噪声频率一般具有频率高，幅值小的特点，因此都不在稀疏频谱计算范围之内，因此就相当于过滤了噪声频率。但这也只是较为理想的一类情形，对于频率低，幅值大的一些噪声频率，稀疏频谱表达算法只能将其认为是一个显著稀疏频率对待，无法达到去噪的目的。所以，对于噪声较大的一类图像信号，其稀疏频谱表达图像具有一定的降噪能力，并且，由于稀疏频谱的大小要远小于二维空间图像信号和完整的傅里叶频谱信号。总的来说，使用图像的稀疏频谱表达达到了降维降噪的目的，建立在稀疏频谱表达上的信号处理使得傅里叶频谱分析和一些空间图像处理变得更为方便和简单。

### 5.3 本章小结

本章通过实验验证了通过第四章设计实现的二维稀疏频谱表达计算方法得到的图像稀疏频谱表达的正确性，对比不同大小的各类图像，其峰值信噪比的差异稍大，与源

图结构相似度都比较高，可以验证基于该稀疏频谱表达的图像是正确的。并且，由于作为图像表达的稀疏频谱都是幅值较大的显著稀疏频率，其作为噪声频率的概率较低，所以该表达相比源图像具有一定的降噪能力。并且，稀疏频谱表达的图像频谱大小远小于快速傅里叶变换频谱，使得无论是获取傅里叶频谱还是傅里叶频谱分析变得更为简单高效。最后，在性能上，*FFTW*在图像规模偏小时都具有更好的性能，统计发现，当输入图像的大小达到 $2^{24}$ ，稀疏度为空间信号大小的约5%以内或采样行列总数为所有行列的25%以内时，稀疏傅里叶频谱的计算方法由于*FFTW*，并随着图像大小的增大，这个比例也随之提高。

最后，本文稀疏频谱表达的方法在实验室两个项目中得到了运用。在地质体图像的频域处理中，稀疏频谱表达算法能够快速的获取大型地质图像的稀疏频谱，该频谱可以用于各种频域分析，并且，稀疏频谱具备降维降噪的能力，使得该稀疏频谱表达的图像变得平滑，另外降维后的频谱也更利于傅里叶分析。

## 第六章 人脸识别流水线系统的设计与实现

本文对图像稀疏频谱表达的研究直接用于实验室基于卷积神经网络的火车站监控视频人脸识别项目，作为其其中的一个组成部分，主要用于将图像的稀疏频谱表达作为分类人脸图像的频域特征，提高卷积神经网络的分类准确度。其原理在于，图像的稀疏频谱表达具有降维降噪的能力，由于频域信号和空间信号的等价性，稀疏频谱表达相当于提取了图像的主要特征，虽然缺失了一些图像的细节，但并不影响图像分类和识别的准确性，因为分类和识别更注重图像的整体语义<sup>[24]</sup>。因此，稀疏频谱表达在图像的分类和识别中的作用非常类似于主成分分析方法(PCA)。本节首先描述卷积神经网络人脸识别系统的整体框架，然后详细介绍每个部分的功能和设计实现。最后根据实验结果分析整个人脸识别系统的识别能力。

### 6.1 系统需求分析概述

系统主要完成在提供的人脸库上训练人脸图像分类器，并对监控视频中截取的特殊人脸进行识别，从而达到判断对火车站内是否出现通缉犯的目的。并且，由于现在实行实名制的火车乘车方式，为了杜绝使用他人身份证件购买火车票进站的情况，系统也期望在过进站时能够自动检测该乘车人是否和票面上身份证件号所对应的乘车人相符合。由于本文项目主要涉及算法库的设计实现，因此本章简要概括系统的需求，如下图所示：

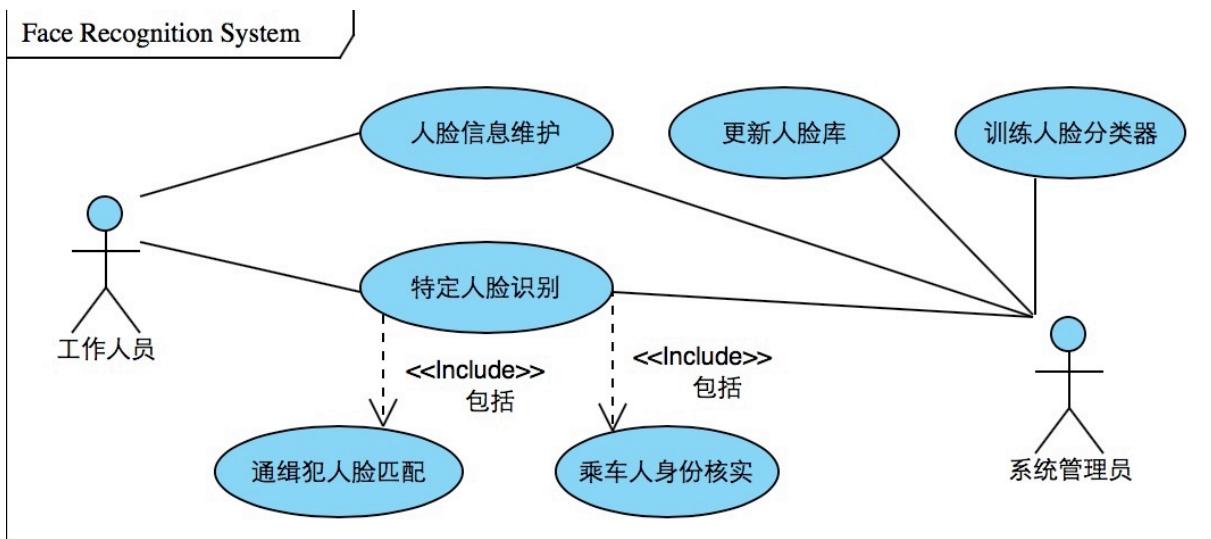


图 33 人脸识别系统总体需求

具体的系统平台实现会具备更多的用户管理，信息管理等模块功能，但这些不属于本文的主要内容。图 33 用例图中主要列出了和本章项目紧密关联的需求，也是整个人脸识别系统的核心需求。

## 6.2 系统人脸识别总体结构

本章所述系统主要用于从火车站内的监控视频中检测人脸图像并识别，主要涉及的关键技术包括人脸检测与提取，人脸特征对齐，人脸分类，人脸匹配识别等。可以分为训练和识别两个过程。以下是两个过程的主要程序处理流程：

人脸图像库训练分类过程：

- Step1. 读取分类好的人脸图像库。
- Step2. 计算每个人脸图像的稀疏频谱，并将其作为训练人脸图像的频域特征。
- Step3. 在经处理好的人脸图像库训练卷积神经网络分类。
- Step4. 每一类人脸图像经卷积神经网络的输出作为该人脸的特征向量字典，本地存储。

待检测人脸识别匹配过程：

- Step1. 人脸采集，从视频信号中按帧截取图像。
- Step2. 基于图像的 *Haar-like* 特征检测并提取人脸图像，直方图均衡化，归一化。
- Step3. 计算待识别人脸图像的稀疏频谱。作为频域特征与原人脸图像一起作为分类器的输入。
- Step4. 计算待识别人脸图像和稀疏频谱特征经分类器后的输出，作为特征向量。
- Step5. 相似人脸识别，基于特征向量匹配，在训练好的人脸特征向量字典中找到与待识别人脸特征向量最相似的特征向量分类。

其他跟平台相关平台管理流程和模块不在本章的内容范围之内。上述两个过程构成了整个人脸识别系统的核心部分，按功能和实现总体划分为五个模块，分别为：图像采集模块，人脸检测与提取模块，预处理和特征提取模块，分类器模块，人脸匹配识别模块。图 33 简要说明了各模块的工作内容和模块之间的流水线关系，图中，人脸采集部分主要包含了图像采集，人脸检测提取，预处理和利用本文算法提取稀疏频域特征等模块的内容，从系统流程来看，属于整个识别过程的输入预处理角色。训练识别部分包含了在人脸库上训练分类器，以及在训练好的分类器上计算待识别人脸的特征向量，和基于特征向量的人脸匹配等过程内容，属于分类器模块和识别模块，也是整个系统的核心部分。图中人脸采集和训练识别是两个低耦合的部分，逻辑上是流水线关系。详细各模块的具体实现细节在下一节介绍：

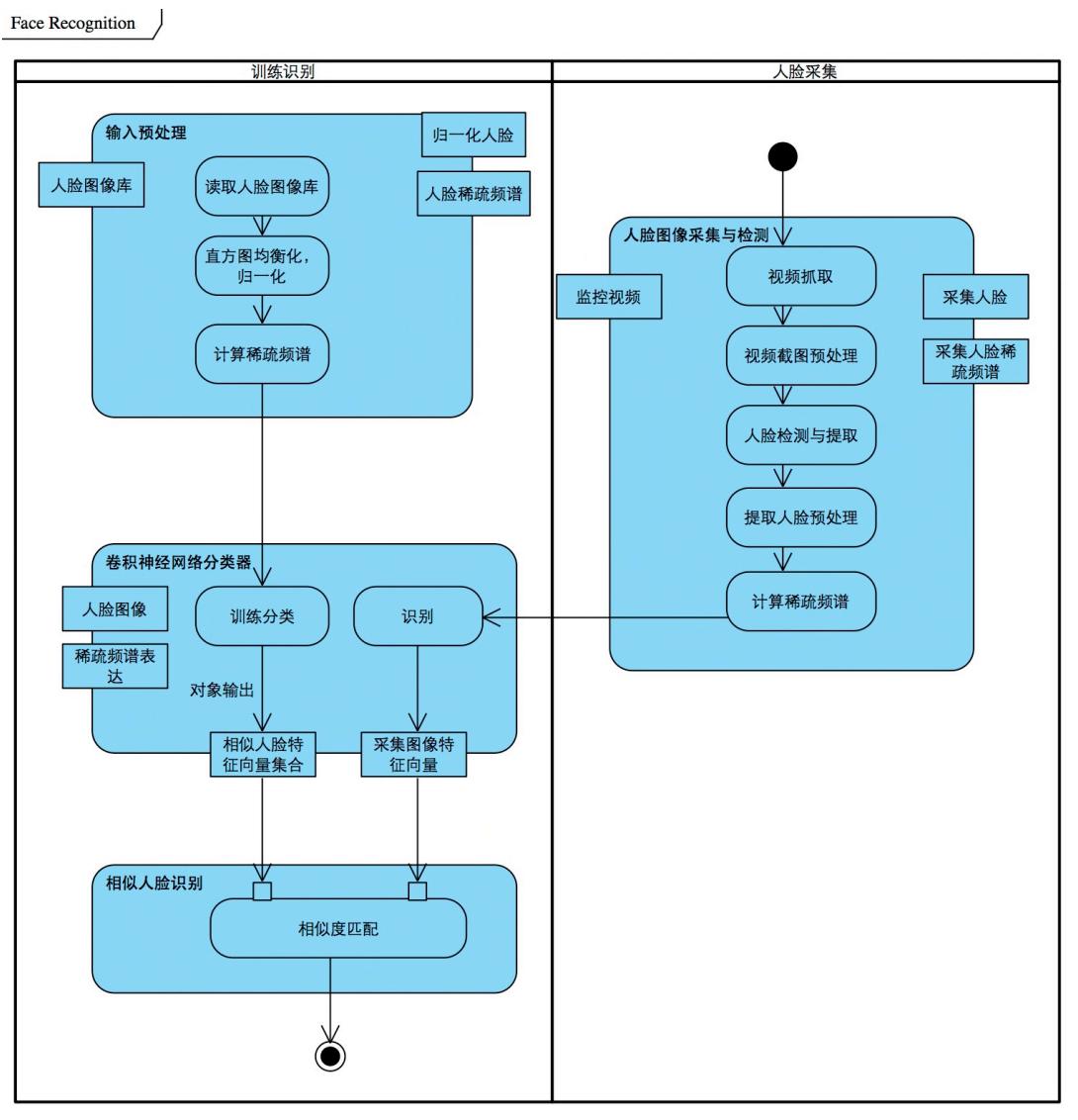


图 34 人脸识别系统处理流程

### 6.3 系统各模块的实现

本节主要介绍从视频图像截取到人脸识别的整个流水线过程中涉及的主要模块，以及详细的设计实现，整个过程借助了一些开源代码和算法库。

#### 1. 人脸图像采集模块

待识别人脸来自于西站火车站的监控视频，为了从中提取到合适的人脸图像，系统逐帧截取视频，将视频切割为很多的连续图像，保存到本地。这部分程序实现为 C++ 程序，依赖开源计算机视觉库 *OpenCV*，主要是组合使用 *cvQueryFrame* 和 *cvGrabFrame* 函数从摄像头或视频文件中抓取帧。

下表是主要的实现函数及其相应的功能：

表 7 人脸采集模块主要函数功能

<i>FaceProcess()</i>	从视频中采集图像，调用 FaceTracking 类提取人脸图像
<i>videoPlayerThread()</i>	多线程按帧抓取视频
<i>VideoForm_load()</i>	加载视频数据
<i>VideoForm_FormClosing()</i>	停止视频数据加载，清除缓存

## 2. 人脸检测模块

采集模块中抓取的图像是从视频中按帧截取的，因此得到的很多图像中并不包含人脸，或人脸特征不明显。为了抓取人脸图像，本系统中采用了基于 *Haar-like* 特征的目标检测方法。检测过程中构建一组简单的分类器，称为级联分类器，并通过人脸样本的 *Haar-like* 特征作为正例来训练来强化级联分类器的分类能力。训练完成的级联分类器可以用于检测输入图像中的感兴趣区域，即人脸区域。本节系统实现是采用开源 *OpenCV* 实现，主要依靠其中级联分类器的实现 *CascadeClassifier* 类，并借助其配置文件 *haarcascade\_frontalface\_default.xml* 来初始化来构造一个训练好的人脸检测分类器，整个模块基于 C# 实现。以下是程序实现主要类和函数，表中函数均省略参数。

表 8 人脸提取主要函数及功能

<i>IsTracked()</i>	判断是否检测出人脸；
<i>TrackedCount()</i>	返回检测出人脸的数量；
<i>Starts()</i>	开始检测人脸，主要调用 <i>startTracking()</i> ；
<i>startTracking()</i>	控制人脸检测整体流程，调用 <i>prepareTrack()</i> , <i>doingTrack()</i> ；
<i>doingTrack()</i>	创建级联分类器，基于 <i>harr-like</i> 特征检测图像中的人脸；
<i>prepareTrack()</i>	人脸检测前的一些初始化操作；
<i>DrawFaces()</i>	把检测到的包含人脸的矩形绘制到原始图像上；
<i>singleAngle()</i>	单一角度检测；
<i>multiAngle()</i>	多角度检测；
<i>rotateDetection()</i>	将图像旋转一个角度后进行检测；
<i>saveImg()</i>	保存提取出的人脸图像；
<i>GetFaces()</i>	获取人脸图像；

下图是采集和检测模块的主要实现类及其主要函数：

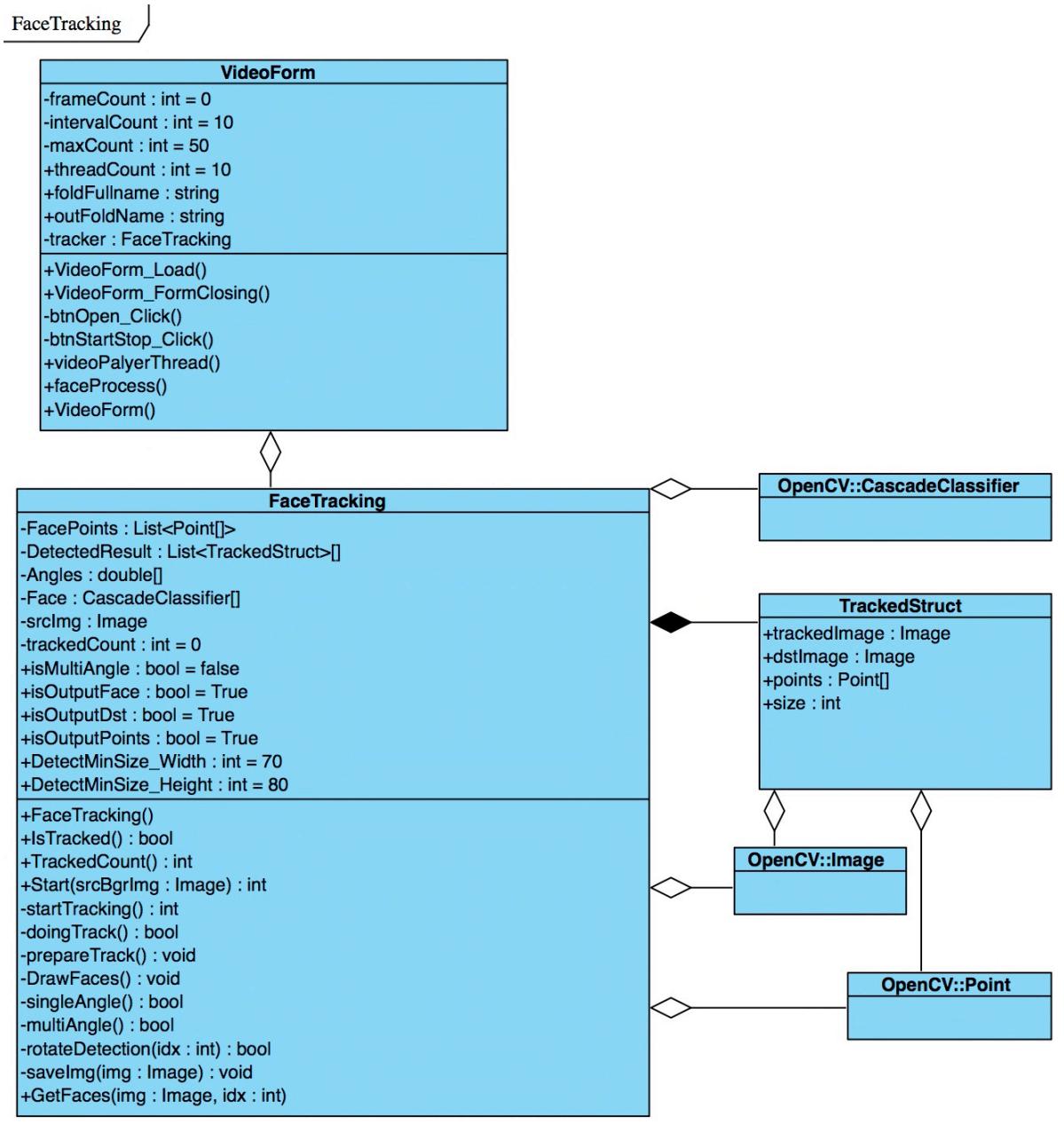


图 35 人脸检测与提取设计实现

### 3. 预处理和频域特征提取模块

该模块主要计算上一模块抓取的人脸图像的稀疏频谱，通过该稀疏频谱表达的人脸作为预处理后的人脸图像，从而平滑图像，并通过直方图均衡化，归一化人脸图像。最后，空间稀疏频谱表达图像作为空间特征，稀疏频谱作为频域特征，一起作为分类器的输入。稀疏频谱作为人脸图像的一种频域特征，由于其稀疏性质，稀疏频谱特征相比空

间图像具有较低的维度，并且，稀疏频谱由空间图像的主要频率构成，去除了空间图像中的无关噪声，在这一点上非常类似于主成分分析方法。本模块主要由本文给出的二维稀疏频谱表达方法构成，采用无验证模式。能够在较快时间内计算得到人脸图像的稀疏频谱。相比快速傅里叶变换，本文方法计算得到的稀疏频谱是过滤了一定的噪声，留下的稀疏频率都是空间人脸图像的主要特征体现。本模块主要为 C++ 实现，其稀疏频谱计算的实现与本文相同。下表是主要的实现函数与其相应功能的描述：

表 9 初始化模块主要函数

<i>detect_face()</i>	基于 haar-like 特征检测人脸并提取；
<i>show_landmarks()</i>	显示人脸特征 <i>landmarks</i> ；
<i>align()</i>	将检测到的人脸对齐；
<i>detectLandmarks()</i>	删除人脸特征 <i>landmarks</i> ；
<i>detect_align()</i>	检测人脸图像是否对齐；
<i>getSparseRepr</i>	计算人脸图像的稀疏频谱表达；
<i>preprocess()</i>	提取人脸预处理，直方图均衡化，归一化；

下图是和表 9 中函数对应的实现类和相关成员变量设计。

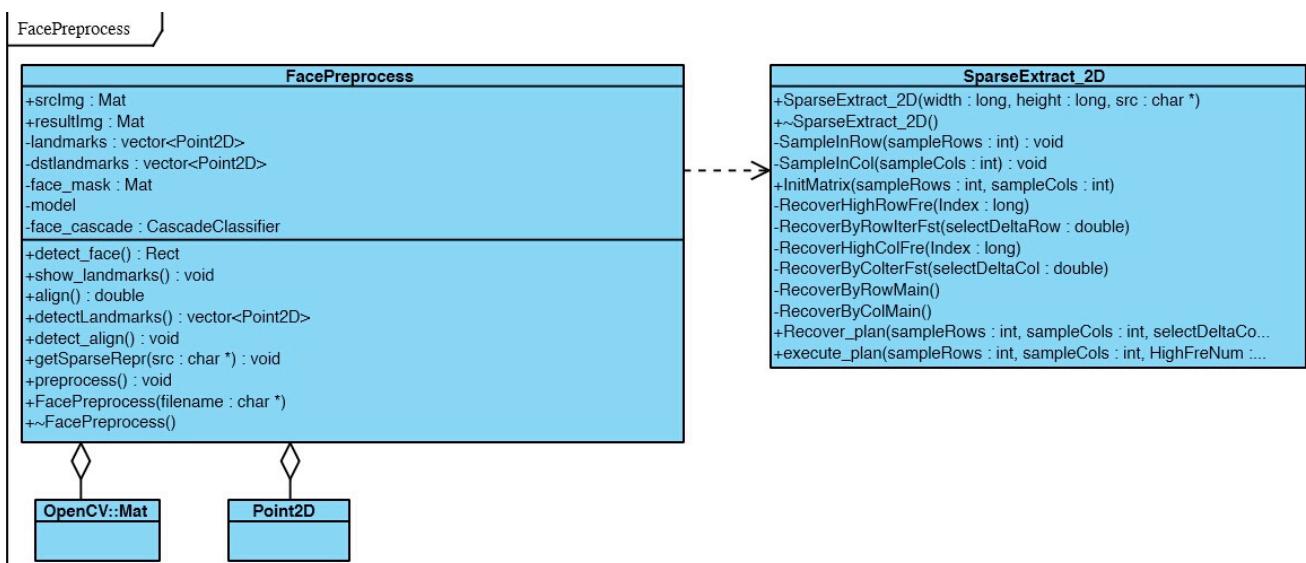


图 36 人脸图像初始化主要设计与实现

#### 4. 分类器模块

本模块是整个系统的核心模块，具有相当高的复杂度。本节主要介绍基于卷积神经网络分类器的实现人脸的分类和识别。卷积神经网络是近年来深度学习的热门，广泛运

用于机器学习，图像识别等领域。其基本原理如下图所示：

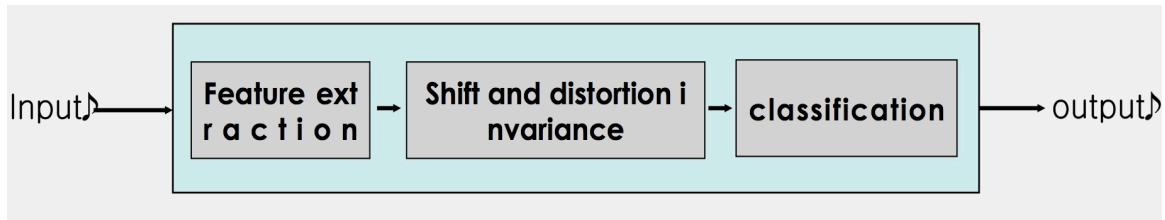


图 37 单层卷积分类

输入图像经过不同的卷积窗口做空间卷积形成不同的图像，这些图像可以认为是经卷积窗口提取的特征，这一层称为特征映射层，接下来是一个子采样过程，其减小了特征映射层中每个卷积特征的空间分辨率，最后在子采样后的特征上输出一个特征向量，并以该特征向量作为分类依据，卷积神经网络具有多层这样的结构，上一层作为下一层的输入，组成神经网络，因此具有很高的分类能力，如下图所示：

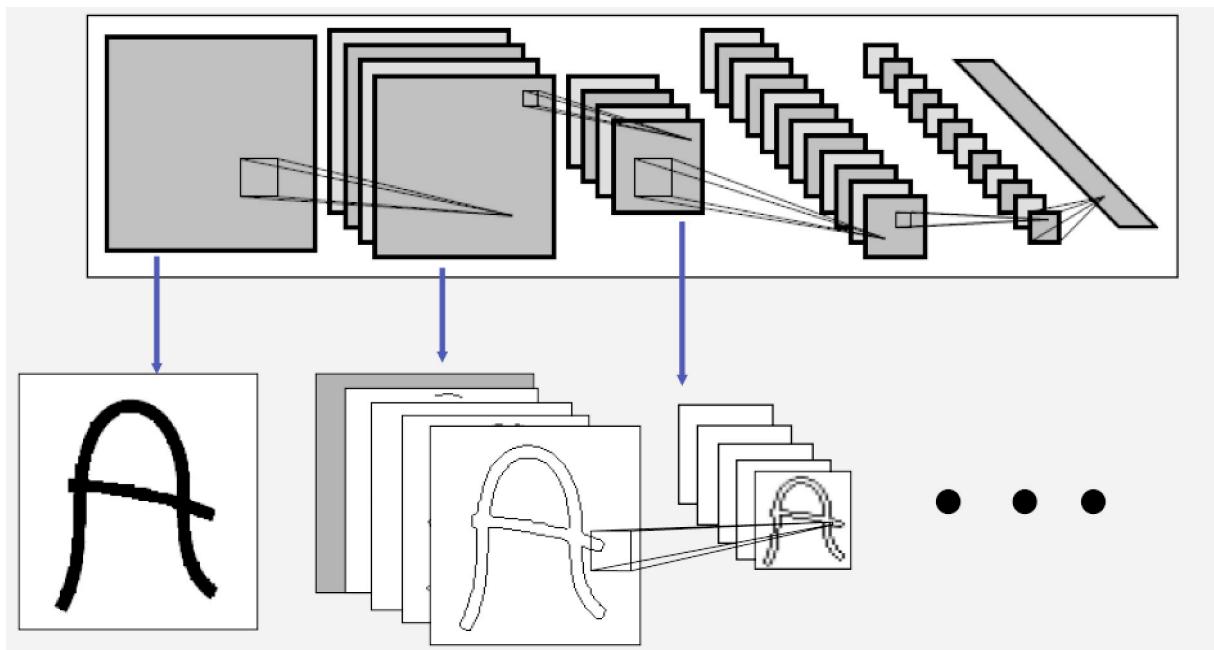


图 38 多层卷积神经网络

本节系统中使用了开源的卷积神经网络实现 *Cuda convnet2*，并在已有的人脸图像库上训练神经网络。本系统的创新之处在于将人脸图像的稀疏频谱与人脸图像捆绑作为卷积神经网络的输入，输出的特征向量为空间人脸图像特征向量和稀疏频谱特征向量的组合。在一定程度上提高系统的分类和识别能力。

由于这部分依赖了开源的实现，因此不详细展开关于卷积神经网络的具体实现细节，本章简要介绍系统在此处的调用过程，并简单介绍其中的与系统调用程序直接相关的几个开源类和开源函数的功能。下图 39 是主要的卷积神经网络分类器调用类图：

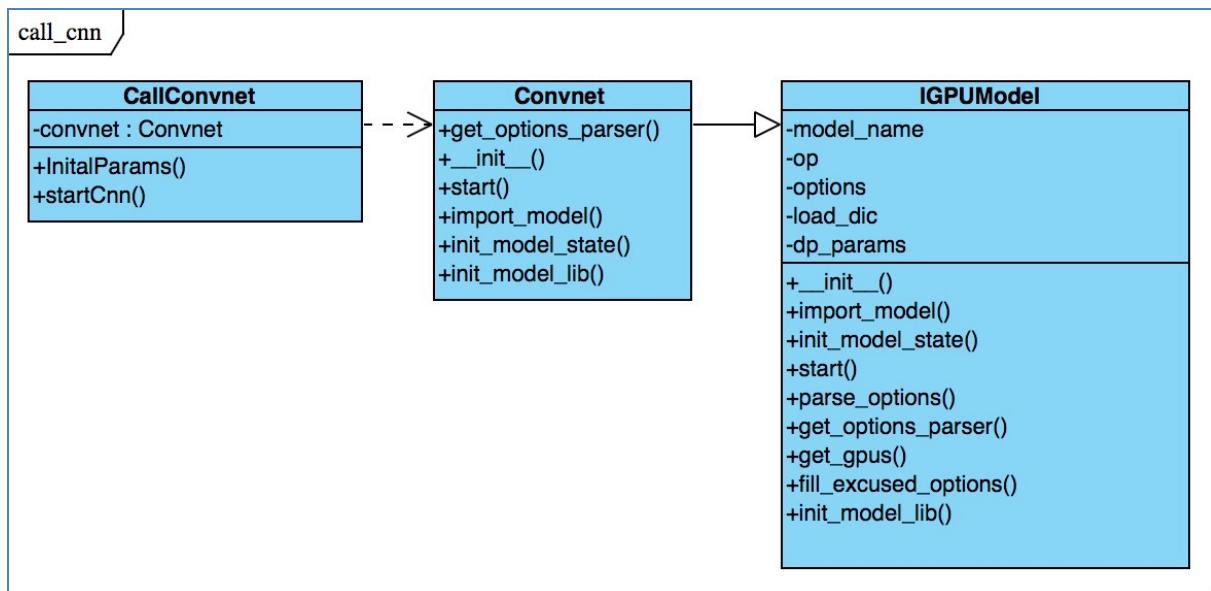


图 39 卷积神经网络分类器调用类图

如图所示，类 Convnet 是整个分类器的主类，并且是 IGPUModel 的子类，都是 Python 实现，源文件中这两个类都有较多函数和成员变量。图 39 中只列出了和系统调用相关的主要函数，并省略了参数，图中 Convnet 类列出的只是其复写了父类的函数。详细内容可见源码文件 Convnet.py 和 gpumodel.py，下面简要描述相关函数的功能的调用过程：

表 10 开源 CNN 主要接口调用函数

CallConvnet 类	
initialParams()	调用 get_options_parser 函数获取 OptionParser 对象，调用 parse_options 函数读取 Checkponit 文件配置到 load_dic，解析参数到参数集合，实例化 Convnet 类，初始化分类器主要参数。
startCnn()	调用 Convnet 对象的 start() 函数，开始迭代。
Convnet 类	
__init__()	类构造函数，初始化系统参数，包括参数对象集合，存储模型运行状态字典等，同时调用其他函数帮助初始化。
get_options_parser()	获取 OptionParser 对象，用于参数解析。
init_model_state()	初始化模型状态字典
import_model_lib()	导入 c++核心库

## 5. 相似人脸识别模块

在人脸图像库上训练卷积神经网络分类器完成后，得到每个人脸图像的一个特征向

量，相同人脸的特征向量是相似的，因此可以将其归为一类。对采集到得新人脸进行识别时，同样将直方图均衡化和归一化的图像与其稀疏频谱作为卷积神经网络输入，得到一组特征向量输出，识别该人脸的方法就是在从以训练分类的人脸图像库特征向量集合中找到和该人脸特征向量相似的向量，认为相似向量所属的人脸和识别人脸相同，从而达到识别目的。但是，由于人脸图像库较为庞大，其特征向量集合中得向量也非常多，这使得寻找相似向量变得困难。本节系统采用局部敏感哈希方法，目前也是采样了开源的 *E2LSH* 库。

## 6.4 实验分类识别结果

以下对比直接人脸图像输入和人脸图像加频谱特征输入两种情况下卷积神经网络的分类识别能力。

实验进行时，*batch size* 均为 256，实验的 *batch num* 为 49，实验的测试区间为 41-49。迭代次数均为 380 次。详细参数见附录。下表是对 10 类人脸分类的结果：

表 11 10 类人脸对比系统分类识别结果

	Top-1 error	Top-5 error	Multiview top-1 error	Multiview top-5 error
空间图像特征	29.0901%	4.2904%	26.9684%	3.4418%
加入稀疏频谱特征	23.6681%	2.0754%	21.6879%	1.3673%

实验表明，将稀疏频谱作为分类的附加特征向量，能够提高卷积神经网络对人脸的分类能力，实验中，Top1 分类误差降低了 6 个百分点。系统整体的识别正确率为 77% 左右，图 40 是分类误差随迭代次数的变化曲线。

在对西站视频图像中截取并人工分类的 29 人的人脸图像，平均每人的人脸图像数在 20 左右的人脸识别实验中，系统分类识别的正确率为 77%+，图 39 是系统的分类误差随迭代次数的变化曲线图，随着迭代次数的增多，识别正确率最终稳定在 77% 左右。

表 12 29 类人脸对比系统分类识别结果

	Top-1 error	Top-5 error	Multiview top-1 error	Multiview top-5 error
空间图像特征	30.1375%	4.1708%	28.3761%	3.5314%
加入稀疏频谱特征	23.2619%	2.3597%	22.7622%	1.4983%

对比车站 29 类人脸的分类识别，加入稀疏频谱特征后系统分类识别的正确率提高约 6%+。图 41 是分类误差随迭代次数的变化曲线，从图中可以看出系统识别正确率的变化趋势。

根据以上实验，可以得出结论：加入人脸的频域特征能够提高人脸分类识别的准确性。

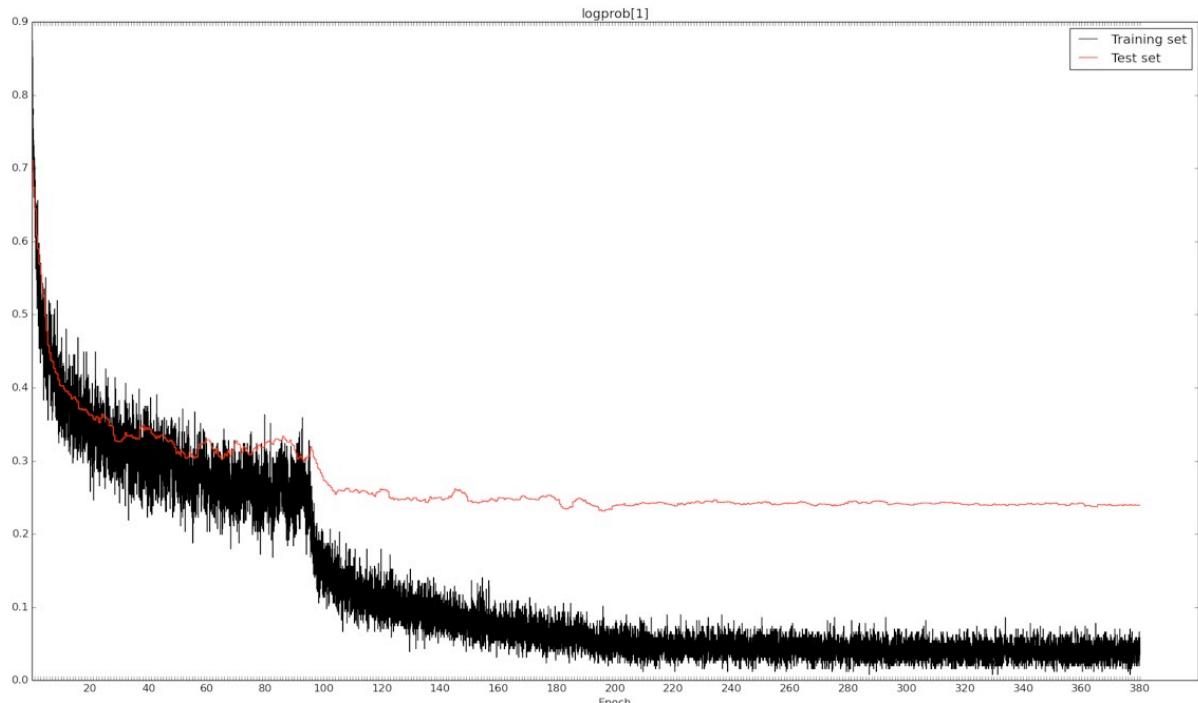


图 40 人脸库 10 类人脸系统分类差误差变化曲线

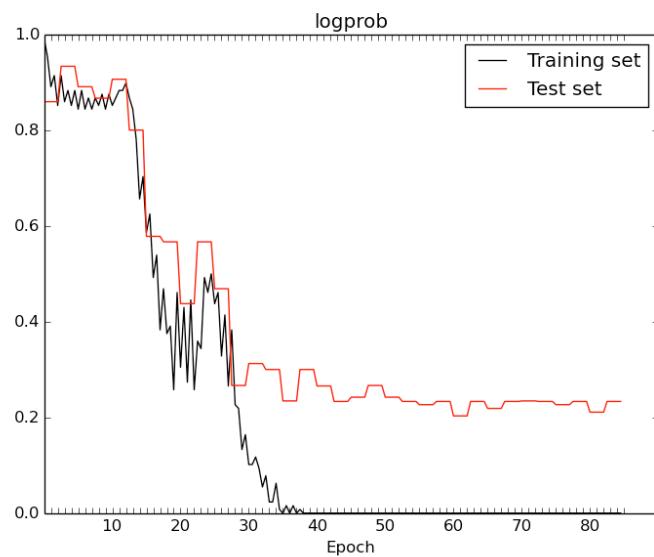


图 41 火车站 29 类人脸图像系统分类识别误差

## 6.5 本章小结

在人脸识别应用中，稀疏频谱表达的图像包含完整的图像信息，但具有较小的维度，使得其可以作为图像分类和识别的特征，可以看做是一种基于稀疏频域的主成分分析方法。在本章所述系统中，综合使用了图像的空间特征和稀疏频谱特征作为分类器的输入，在利用卷积神经网络作为分类器的系统中，提高了空间图像分类识别的准确率。目前这方面的运用仍在进一步完善和提高。

## 总结与展望

### 结论

随着数字信号处理领域的深入发展，原本很多基于的正交基变换的信号表达方法越难越难以满足日新月异的应用场景。从最开始的傅里叶变换分析，到为了更好的适应需求的小波变换，多尺度变换等。最近几年，对使用稀疏的方式来表达信号的研究取得了很多成果，例如基于过完备字典集的稀疏信号表示等。而傅里叶变换作为基本的信号处理分析方法，在今天其重要性仍然是无可替代的。近两年对稀疏傅里叶变换算法的研究取得了重大突破，使得计算离散信号的傅里叶变换的时间复杂度从快速傅里叶变换算法的对数级变为亚线性。这使得对很大规模的信号的傅里叶变换变得简单可行，很快应用到很多工业应用中，绪论部分讲述了其在某些领域的重大贡献。但目前大部分的研究都主要集中在对一维信号的稀疏频谱表达。在图像处理领域，傅里叶变换在和图像的频域信号直接相关，因此也非常需要一种简单可行的稀疏频谱表达算法，能够通过稀疏频谱很好的表达图像信号。

本文的目的，就是实现这样一个适用于二维离散信号，尤其是二维图像信号的稀疏频谱表达算法。通过该算法，可以在不计算二维傅里叶变换的条件下，求解出二维离散信号的稀疏频谱。本文中，首先从现有的理想二维严格稀疏傅里叶变换算法出发，分析了其适用的条件，及在非理想环境无法使用的原因，并针对其不足之处，找到了更好解决的方案。改进后算法适用于二维严格稀疏频谱，并且，对稀疏度也有一定的约束，但适用范围较原来的理想算法广了很多。由于改进算法对严格稀疏频谱的强假设，导致其并不能用于二维图像的稀疏频谱表达，其中一个原因在于图像频谱包含大量噪声频率，因此图像的稀疏频谱是一种噪声稀疏频谱，另一个原因在于图像频谱的分布也不满足理想算法和改进算法的频谱分布的假设。因此，在第四章中，针对二维图像频谱分布的特点，对改进二维严格稀疏频谱表达方法做了进一步修改，除了延用原算法组合采样行列进行稀疏频谱的策略外，修改后算法在逻辑上变化较大，但主要的稀疏频谱计算方法还是借鉴了第二章中基于线性编码理论的一位稀疏傅里叶变换算法和第三章中得严格稀疏频谱恢复方法。最后，在第五章中，通过实验完成稀疏频谱表达方法的验证，通过图像质量评价指标，可以证明基于本文算法的稀疏频谱表达确实能够较好的表达二维图像，但是仍由较多不足之处。

不足之处之一在于针对二维图像的稀疏频谱表达算法还的实现还不够优化，以至于在实验和应用中，在图像的稀疏频谱表达并没有达到预想的性能，甚至在很大部分图像频谱计算上其运行时间大于  $FFTW$ 。另外较为明显的一点是，对噪声频率的控制还不够完善，使得存在一定的概率将一些非噪声的稀疏频率判断为噪声。

综上所述，本文主要概况为以下结论：

1. 研究了现有稀疏表示理论及稀疏傅里叶变换算法，改进了理想二维严格稀疏傅里叶变换算法。
2. 调研了图像频谱的稀疏性质，证明了稀疏频谱表达在图像信号表达上的可行性。
3. 设计并实现了一种二维稀疏图像频谱表达方法，验证了稀疏频谱表达图像可行性。
4. 在地质图像处理和人脸识别系统应用中验证了本文设计的图像稀疏频谱表达方法。

## 展望

由于时间等问题影响，本文仍有若干问题需要解决，首要的一点是优化程序实现，使之能够达到算法预想的性能，在所有满足条件的图像频谱计算中无论在时间还是空间性能上都由于  $FFTW$ 。另外的一点是，本文设计并实现的图像稀疏频谱表达方法仍然是一种串行的算法程序，没有利用好现代计算机多核多线程的优势，而事实上算法逻辑中很多地方是可并行的。最后的一点优化是期望能找到一个更好的噪声控制过滤函数，能够在时间域内很好的鉴别频域噪声并在稀疏频谱计算时排除其影响。当然，还有很多其他细节问题，不一一赘述。

除了对算法程序的优化之外，期望基于稀疏频谱的图像表达运用到更多的图像处理和应用中去。由于空间像素表达的图像往往是完备的，其输入规模有时会偏大难以处理，而对大部分图像来说，其频谱都是非常稀疏的，利用稀疏频谱表达图像来代替使用空间图像作为输入，可以大大减小输入规模，伴随而来的就是算法或程序性能的提升。例如在图像分类领域，往往带分类空间图像的很多像素是不需要作为输入的，使用稀疏频谱作为分类的输入，也许可以降低分类的复杂度。总之，稀疏频谱表达为我们在空间像素表达之外开辟了一个广阔的新空间，带着新的视角去看待现在图像处理的很多问题，也许会有很多新的发现，这会是一个有趣的探险。

## 参考文献

- [1] Hubel D H, Wiesel T N. Receptive fields of single neurones in the cat's striate cortex[J]. The Journal of physiology, 1959, 148(3): 574.
- [2] B. A. Olshausen, D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images[J]. Nature, 1996, 381(13): 607-609.
- [3] Ghazi B, Hassanieh H, Indyk P, et al. Sample-optimal average-case sparse fourier transform in two dimensions[J]. arXiv preprint arXiv:1303.1209, 2013.
- [4] Rauh A, Arce G R. Sparse 2D Fast Fourier Transform[J]. Proceedings of the 10th International Conference on Sampling Theory and Applications, 2013: 248-251.
- [5] Kushilevitz E, Mansour Y. Learning decision trees using the Fourier spectrum[J]. SIAM Journal on Computing, 1993, 22(6): 1331-1348.
- [6] Mansour Y. Randomized interpolation and approximation of sparse polynomials[J]. SIAM Journal on Computing, 1995, 24(2): 357-368.
- [7] Gilbert A C, Guha S, Indyk P, et al. Near-optimal sparse Fourier representations via sampling[C]. Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. ACM, 2002: 152-161.
- [8] Gilbert A C, Muthukrishnan S, Strauss M. Improved time bounds for near-optimal sparse Fourier representations[C]. Optics & Photonics 2005. International Society for Optics and Photonics, 2005: 59141A-59141A-15.
- [9] Hassanieh H, Indyk P, Katabi D, et al. Simple and practical algorithm for sparse Fourier transform[C]. Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2012: 1183-1194.
- [10] Hassanieh H, Indyk P, Katabi D, et al. Nearly optimal sparse Fouriertransform[C]. Proceedings of the forty-fourth annual ACM symposium on Theory of computing. ACM, 2012: 563-578.e
- [11] Hassanieh H, Shi L, Abari O, et al. Ghz-wide sensing and decoding using the sparse fourier transform[C]. INFOCOM, 2014 Proceedings IEEE. IEEE, 2014: 2256-2264.
- [12] Hassanieh H, Adib F, Katabi D, et al. Faster gps via the sparse fourier transform [C]. Proceedings of the 18th annual international conference on Mobile computing and networking.

ACM, 2012: 353-364.

- [13] Shi L, Andronesi O C, Hassanieh H, et al. MRS Sparse-FFT: Reducing Acquisition Time and Artifacts for In Vivo 2D Correlation Spectroscopy[C].ISMRM'13, Int. Society for Magnetic Resonance in Medicine Annual Meeting and Exhibition. 2013.
- [14] Donoho D L. Compressed sensing[J]. IEEE Transactions on Information theory, 2006,(04):1289-1306.doi:10.1109/TIT.2006.871582.
- [15] Elad M. Sparse and redundant representations: from theory to applications in signal and image processing[M]. Springer, 2010.
- [16] Chen S S, Donoho D L, Saunders M A. Atomic decomposition by basis pursuit[J]. SIAM journal on scientific computing, 1998, 20(1): 33-61.
- [17] Tropp J A, Gilbert A C. Signal recovery from random measurements via orthogonal matching pursuit[J]. Information Theory, IEEE Transactions on, 2007, 53(12): 4655-4666.
- [18] Sudan M. Decoding of Reed Solomon codes beyond the error-correction bound[J]. Journal of complexity, 1997, 13(1): 180-193.
- [19] Reed-Solomon codes and their applications[M]. John Wiley & Sons, 1999:230-245.
- [20] MacWilliams F J, Sloane N J A. The theory of error-correcting codes[M]. Elsevier, 1977.
- [21] Akçakaya M, Tarokh V. A frame construction and a universal distortion bound for sparse representations[J]. Signal Processing, IEEE Transactions on, 2008, 56(6): 2443-2450.
- [22] 周景超, 戴汝为, 肖柏华. 图像质量评价研究综述[J]. 计算机科学, 2008, (7):1-4.
- [15] Akavia A, Goldwasser S, Safra S. Proving hard-core predicates using list decodingFOCS. 2003, 44: 146-159.
- [23] 郑德品. 基于结构相似度的图像质量评价方法研究[D]. 浙江大学电气工程学院, 2007.
- [24] Wright J, Ma Y, Mairal J, et al. Sparse representation for computer vision and pattern recognition[J]. Proceedings of the IEEE, 2010, 98(6): 1031-1044.
- [25] E.J.Candes and T.Tao, Decoding by linear programming, IEEE Trans.Inf.Theory,vol.52, no. 1, pp. 4203-4215, Dec,2005;
- [26] F.J.MacWilliams and N.J.A.Sloane, The Theory of Error Correcting Codes,9th ed Amsterdam, The Netherlands:Elsevier Sci.1996;
- [27] M.Sudan, Decoding of Reed Solomon codes beyond the error-correction bound, in

- Proc.37th Ann. Symp. Foundations Comput.Sci.,1996;
- [28] J.A.Tropp, Greed is good: Algorithmic results for sparse approximation, IEEE Trans.Inf. Theory,vol.50, no. 10, pp. 2231-2242;
- [29] J.A.Tropp and A.C.Gilbert, Signal recovery from partial information via orthogonal matching pursuit, IEEE Trans.Inf. Theory, vol. 53, no. 11, pp. 4655-4666, Dec.2007;
- [30] Sarah Spence Adams, Introduction to Algebraic Coding Theory
- [31] V.Y. Pan. Univariate polynamials: Nearly optimal algorithms for numerical factorization and root-finding. J. Symbolic Computation, 2002;
- [32] J.Massey.Shift-register synthesis and bch decoding. Information Theory, IEEE Transactions on, 15(1):122-127, jan 1969;
- [33] Rafael C.Gonzalez , Digital Image Processing[M].第三版.北京:电子工业出版社,2010.1.1 : 1-300;
- [34] P. Duhamel and M. Vetterli,, Fast Fourier transforms: a tutorial review and a state of the art, 1990, Signal Processing 19: 259–299;
- [35] Bracewell, R. N., The Fourier Transform and Its Applications (3rd ed.), Boston: McGraw-Hill 2000 : 1-249;
- [36] E. Porat and M. Strauss. Sublinear time,measurement-optimal, sparse recovery for all. Manuscript, 2010.
- [37] Y.Mansour.Randomized interpolation and approximation of sparse polynomials. ICALP, 1992;
- [38] I.S.Reed, G.Solomon , Polynomial codes over certain finite fields, J. Soc. Indust. Appl. Math., 8 (1960), pp. 300-304;
- [39] F. MacWilliams, N.Sloane, The Theory of Error Correcting Codes, North-Holland Publishing Company, Amsterdam, 1977;
- [40] 方维,孙广中,吴超,陈国良. 一种三维快速傅里叶变换并行算法. 计算机研究与发展, 2011.03:440-446;
- [41] 陈国良. 并行算法设计与分析 [ M ]. 北京 : 高等教育出版社, 2002;
- [42] compressiveSensingResources,RiceUniversity, Accessed 2010. [Online]. Available: <http://www.dsp.ece.rice.edu/cs/>
- [43] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional

- neural networks[C].Advances in neural information processing systems.2012: 1097-1105.
- [44] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010.  
www.image- net.org/challenges. 2010.
- [45] L. Breiman. Random forests. Machine learning[M], 45(1):5–32, 2001

## 附录

二维图像稀疏频谱表达算法伪代码：

#1.重要计算变量初始化伪代码

*InitMatrix(width,height,sampleRow,sampleCol)*

{

*wn1=cos(2\*pi/width)+isin(2\*pi/width);*       #1 的 n 阶复根

*For(i=0;i<sampleCol;i++)*

    {

*For(j=0;j<sampleCol;j++)*

        {

*RowMatrix[i,j]<—pow(wn1,i\*j);*

        }

    }

#矩阵求逆

*RowMatrix <—InverseMatrix(RowMatrix);*

*wn1=cos(2\*pi/height)+isin(2\*pi/height);*       #1 的 n 阶复根

*For(i=0;i<sampleRow;i++)*

    {

*For(j=0;j<sampleRow;j++)*

        {

*ColMatrix[i,j]<—pow(wn1,i\*j);*

        }

    }

#矩阵求逆

*ColMatrix <—InverseMatrix(ColMatrix);*

#计算选择函数序列

*Gauss<—DiscreteGaussSignal(0,selectDelta);*

#选择函数与矩形函数乘积序列；

*FilterWindows <—Sample(Gauss\*Box\_signal,C);*

```

}

#行采样信号， 预处理并计算一维傅里叶变换
SampleInRow(src,width,height,sampleRow)
{
    OneRow <—Zeros(1,width);
    For(i=0;i<sampleRow;i++)
    {
        OneRow <—GetSrcByRow(src,i);
        OneRow <—Normalization();
        FreOneRow <—FFT(OneRow);
        #中间频谱指针数组赋值;
        RowMiddleFre(i) <—FreOneRow;
    }
}

#列采样信号， 预处理并计算一维傅里叶变换
SampleInCol(src,width,height,sampleCol)
{
    OneCol <—Zeros(1,height);
    For(i=0;i<sampleCol;i++)
    {
        OneCol <—GetSrcByCol(src,i);
        OneRow <—Normalization();
        FreOneCol <—FFT(OneCol);
        #中间频谱指针数组赋值;
        ColMiddleFre(i) <—FreOneCol;
    }
}

RecoverHighRowFre(RowMiddleFre,ColMiddleFre,Index)
{
    VextorXcd row(ROW_HIGH);

```

```

For(i=0;i<ROW_HIGH;i++)
{
    row[i]← ColMiddleFre[i]/Index];
}

row← Convolution(row,FilterWindow);
#读取傅系数逆矩阵的常数行常数列。行列下标标准见第四章算法描述;
LittleRowMatrix ← RowMatrix(ROW_HIGH,ROW_HIGH);
VextorXcd rowSfre← LittleRowMatrix * row;
wcol← conj(cos(2*pi*Index/height+isin(2*pi*Index/height));
#更新行采样中间频谱
For(m=0;m<ROW_HIGH;m++)
{
    SparseFre[Index][m]=rowSfre[m];
    For(n=0;n<sampleRow;n++)
    {
        Wupdate ← pow(wcol,n)*rowSfre[m];
        RowMiddleFre[n][m] -= Wupdate;
    }
    RowBitSet[Index] = 0;
}

#逐行计算零散稀疏频谱
RecoverByRowIterFst(RowMiddleFre,ColMiddleFre,selectDelta)
{
    For(i=0;i<height;i++)
    {
        #逐行计算是否在行方向高斯滤波函数截止范围外;
        If (GetGaussByRow(seletDelta,i) == True)
        {
}

```

```

    RecoverHighRowFre(RowMiddleFre,ColMiddleFre,i);

}

}

}

RecoverHighColFre(RowMiddleFre,ColMiddleFre,Index)

{

VectorXd col(COL_WID);

For(i=0;i<COL_WID;i++)

{

col[i]<- RowMiddleFre[i][Index];

}

col<- Convolution(col,FilterWindow);

LittleColMatrix <- ColMatrix(COL_WID,COL_WID);

VextorXcd colSfre<- LittleColMatrix * col;

wrow<- conj(cos(2*pi*Index/width),isin(2*pi*Index/width));

For(m=0;m<COL_WID;m++)

{

SparseFre[m][Index] = colSfre[m];

For(n=0;n<sampleCols;n++)

{

Wupdate <- pow(wrow,n)*colSfre[m];

ColMiddleFre[n][m] -= Wupdate;

}

}

ColBitSet[Index] = 0;

}

```

```

RecoverByColIterFst(RowMiddle,ColMiddleFre,selectDelta)

{

For(i=0;i<width;i++)

}

```

```

{
    If(GetGaussByCol(selectDelta,i)==True)
    {
        RecoverHighColFre(RowMiddle,ColMiddleFre,i);
    }
}

#计算高斯区显著稀疏频率-按行方向。列方向同理
RecoverByRowMain(RowMiddleFre)
{
    For(i=0;i<height;i++)
    {
        If(RowBitSet[i] == 1)
        {
            sparseK<— count(ColBitSet,1);
            #检查采样是否足够计算稀疏频率。
            if(sparseK > sampleCols)
            {
                Print “sparseK may be a good sample cols”;
                System.exit();
            }
        }
    }
}

#构建主傅系数逆矩阵
MainRowMatrix <— RowMatrix(sparseK,sparseK);
VectorXcd row(sparseK);
For(j=0;j<sparseK;j++)
{
    row[j] <— ColMiddleFre[j][i];
}
row<— Convolution(row,FilterWindow);
VectorXcd rowFre<— MainRowMatrix *row;

```

```

For(m=0;m<sparseK;m++)
{
    SparseFre[i][m] = rowFre[m];
}
RowBitSet[i]=0;
}

}

SparseExtract(src,width,height,sampleRow,sampleCol,selectDelta,bool flag)
{
    SparseFre <— Zeros(width,height);
    RowBitSet <— 2height-1;
    ColBitSet <— 2width-1;
    #初始化零矩阵;
    RowMatirx <— Zeros(sampleCol,sampleCol);
    ColMatrix <— Zeros(sampleRow,SampleRow);
    #初始化行列傅系数矩阵
    (RowMatrix,ColMatrix.FilterWindow)
        <— InitMatrix(width,height,sampleRow,sampleCol);
    RowMiddleFre <— RowFreSet(sampleRow);  #设置行中间频谱集合
    ColMiddleFre <— ColFreSet(sampleCol);   #设置列中间频谱集合
    RowMiddleFre <— SampleInRow(src,width,height,sampleRow);
    ColMiddleFre <— SampleInCol(src,width,height,sampleCol);
    RecoverByRowIterFst(RowMiddleFre,ColMiddleFre,selectDelta);
    RecoverByColIterFst(RowMiddleFre,ColMiddleFre,selectDelta);
    RecoverByRowMain(RowMiddleFre);
    If(flag == True)
        RecverByColMain(ColMiddleFre);
}

```

## 致谢

在北京航空航天大学的两年多的研究生学习期间，得到了很多老师和同学的帮助，使自己快速的进步，并最终完成毕业设计。其中，有一些人给予了非常大支持，帮助和鼓励。

首先，要感谢的是导师王华锋老师，是王老师带领我们走入了数字图像处理领域，在王老师的悉心教导，严格要求下，不仅完成了一个又一个图像处理难题，也使我们快速的成长，在学业和技术上都有了突破。在整个毕业设计的进行过程中，王老师倾注了大量的时间和精力，并一直以他渊博的知识，开阔的思维，严谨的治学态度指引这我们前进的方向。没有王老师的指导和点拨，就没有我现在的成绩。另外，王老师正直，认真的品格，强烈的事业心和责任感也让我获益匪浅。对王老师的感激之情溢于言表，在此向尊敬的王老师表达衷心的感激。

同时，需要感谢实验室的师兄弟们，感谢主要参与人脸识别项目的师弟蔡叶荷，张延详，和黄江等，是他们不断的改进和测试实验，并为本文提供整理了数据和代码。另外，还要感谢宿舍的舍友韦祖河，尚永宽等为本文撰写和排版提供了很多建议与帮助。