分类号_____密级_____

UDC _____

本 科 毕 业 论 文

压缩感知数据重构算法与仿真

学生姓名 李豪 学号 12020022024

指导教师 任新敏 副教授

院、系、中心 信息科学与工程学院电子工程系

专业年级 电子信息工程 2012 级

论文答辩日期 年 月 日

中 国 海 洋 大 学

压缩感知数据重构算法与仿真

完成日期：_____

指导教师签字：_____

答辩小组成员签字：_____

_____

_____

_____

_____

_____

_____

摘要

作为一种新型的采样理论，压缩感知打破了传统奈奎斯特采样定律的约束，在信号稀疏的前提下通过采样矩阵实现对信号的采样和压缩。压缩感知不再以信号的频率作为采样条件，并且在采样的同时完成对信号的压缩，从而减少了采样得到的无效数据，降低了数据存储、传输的代价。

本文首先介绍压缩感知相关基本理论，包括信号的稀疏表示法，采样矩阵的设计，以及压缩感知数据重构三个部分。重点和关键在于重构算法的研究和仿真，本文选取了常用的一些重构算法，并对他们进行详细的理论研究，给出每个算法的运行原理和执行步骤，制作算法流程图，并利用一维信号和二维信号作为原始信号进行重构仿真实验。利用实验数据，分析比较算法的性能，包括重构时的运行时间、信号的峰值信噪比以及重构的误差，根据各算法表现出来的特性总结其在实际运用中的优点和缺点。

关键词：稀疏表示，采样矩阵，数据重构

Abstract

As a new sampling theory, compressed sensing breaks the limitations of traditional sampling theorem. It can sample and compress the signals which are sparse through the sampling matrix. Compressed sensing does not take the frequency of the signal as the sampling condition, and completes the compression and sampling of signals at the same time, thus reducing the redundant data from sampling, and reducing the cost of the data storage and transmission.

This article explains the basic theory of compressed sensing, including the sparse representation of signal, the design of sampling matrix, and the data reconstruction of compressed sensing. The key is to study the reconstruction algorithms and get their simulation data. Do some theoretical research on Basis Pursuit algorithm, Matching Pursuit algorithm, Orthogonal Matching Pursuit algorithm, Subspace Pursuit algorithm and Compressive sampling matching pursuit algorithm, analyze their principle and operation steps, and get the simulation data by using one dimensional and two dimensional signal to do experiment on reconstruction algorithms. After getting the simulation data of different reconstruction algorithms, compare the time complexity by using their running time data, and compare the reconstruction accuracy by using their reconstruction error data. At last, summarize the advantages and disadvantages of different reconstruction algorithms.

Keywords: sparse representation, sampling matrix, data reconstruction

目录

第一章 绪论

1.1 课题研究背景及意义

随着技术的发展，数字信号已进入人类社会生活的各个领域。例如，手机和计算机里的数码图片，MP3 音乐，MP4 视频文件等。但是在现实生活中人们接触到的往往是模拟信号，比如环境温度，空气湿度，广播电塔发射出的电磁波信号，有线电话通讯中传输的电压信号等。对这一类信号进行处理时通常依赖于现代计算机技术，由于计算机的离散化存储机制，在保存这些信息的时候，需要对其进行时域采样，将其转化为数字信息。

过去的几十年中，奈奎斯特采样定律一直在信号处理领域占据主导地位，通过对模拟信号进行时域间隔采样便可得到离散的样本数据，从而借助计算机进行存储，处理和传输。但该定律指出：若要完全精确的重建信号，采样的频率 F1 和信号频率 F2 需要满足条件：$F1 \geq 2 * F2$。随着科学技术的发展，需要观测的数据不断增大，比如在天文观测，医疗成像，雷达扫描等方面，信号的带宽很大，采样得到的数据量急剧增长，在传输，处理这些数据的时候，往往还要对其进行压缩，丢弃其中冗余的信息，这就带来了很大的时间代价。另一方面，当目的信号的频率过高，超出了现有技术所能采样的频率限制的时候，传统的奈奎斯特采样定律根本无法实现。

为了打破奈奎斯特采样定律的约束，降低信号采集过程带来的成本代价，Donoho 和 Candes 等人于 2006 年引出了一种新型的采样理论——压缩感知（Compressed Sensing，CS）。与传统采样定律先采样后压缩的过程不同，压缩感知采样理论基于信号的稀疏性，在采样的过程中完成对信号的压缩，采样不再需要很高的频率，这样就能获得更少的数据。在信号的重构方面，压缩感知也能借助于极少的采样数据，选取适当的重构算法，实现信号重建。目前常用的算法包括l_1范数法，贪婪类算法等。

对于奈奎斯特采样定律而言，采样频率 F1 和信号频率 F2 满足$F1 \geq 2 * F2$是先决条件，同样，信号的稀疏性也是压缩感知理论的先决条件。然而现实世界中的自然信号往往是非稀疏的，这时就需要利用信号稀疏表示方法对原始信号做

处理。虽然原始信号并不是稀疏的，但可以借助于一些正交变换基，将原始信号投影到正交基上，得到在变换域中表现为稀疏的数据。例如，对于一幅自然图片，其中的像素值几乎都是非零的，但是将这幅图片进行小波变换，得到的数据中大量的系数的绝对值都趋于零，这时便可利用压缩感知做进一步处理。

压缩感知理论自提出以来就受到广泛关注，目前该理论备受关注的方向除了上面所述的稀疏表示外，还有采样矩阵的设计和重构算法的研究。信号通过正交基变换可以转化为稀疏信息，再通过设计好的采样矩阵便能得到采样压缩后的数据，然后通过重构算法实现重构。信号的稀疏表示，采样矩阵的设计都很容易实现，但如何通过采样数据进行信号重建，并且保证较高的恢复效果，一直是研究的重点。

1.2 国内外研究现状

压缩感知于 2006 年被提出以后，受到了国内外的广泛关注，很多领域的学者都对其展开了研究。目前对压缩感知的研究主要集中在信号的稀疏表示，采样矩阵的选取，重构算法这三个方面。

1.2.1 信号稀疏表示理论

压缩感知理论要求所采样的信号必须是稀疏的，然而常见的信号通常都含有大量的冗余信息。这就需要对信号进行某种表示，剔除其冗余信息。目前常用的表示方法有傅里叶变换、离散余弦变换、小波变换、以及多尺度几何分析。傅里叶变换实现了信号由时域到频域的转换，并借助于一系列正弦函数对原始信号进行线性组合和逼近。傅里叶变换虽然应用很广泛，但是相对于小波变换而言缺乏对信号任意局部的表示。但小波信号在一维空间所具有的优势并不能应用于多维空间中，于是就引出了多尺度几何分析方法。

国外的 Meyer，Coifman，Donoho，Candes 等人先后提出了梳妆波变换，楔波变换，小线变换，曲线波变换，为高维信号的多尺度分析奠定了理论基础。目前，对信号的稀疏表示研究仍在继续，研究的热点问题主要是如何找到更好的变换域，使得目的信号在该变换域下的表示尽量稀疏，减少采样数据。

1.2.2 采样矩阵

采样矩阵是压缩感知数据获取中非常重要的一步。Candes 已经证明：采样矩阵的约束等距条件是重构信号的前提。目前国内外研究的采样矩阵中满足这一要求的有局部傅里叶矩阵、高斯矩阵、伯努利矩阵、二进制随机矩阵、循环矩阵等。

在压缩感知中，对于目的信号$\mathrm{X} \in R^N$，构造一个采样矩阵$\Phi(\mathrm{M} \times \mathrm{N}$，$\mathrm{M} \ll \mathrm{N})$。若信号 X 是稀疏的，则采样数据为$\mathrm{Y} = \Phi\mathrm{X}$。于是将 N×1 的目的信号采样后得到 M×1 的数据，当 M 远小于 N 时采样得到的冗余数据很少。因此打破了传统采样定律在高维高频率信号采样方面的局限。目前，国内外对采样矩阵的研究仍在继续。如何找到一种新型的采样矩阵使得采样得到的数据更少，并且在保证重构精度的前提下采样矩阵的性质更好，以及冗余字典和采样矩阵的匹配关系，一直是当前研究的热点问题。

1.2.3 重构算法

根据信号的稀疏表示方法，目前通常采用的重构算法包括l_1范数法，贪婪类算法、贝叶斯方法等。在凸优化类算法中，使用最早的是国外的 Mohimani 提出的 SLO 算法，该算法利用多次迭代，逼近求得最优解。国内的研究学者林婉娟等人在 2011 年提出了新的 NSLO 算法，NSLO 在速度上比 SLO 更快。目前 NSLO 已在 SAR 成像中被广泛应用。而在贪婪算法领域，国外的学者提出了 MP 算法，OMP 算法，CoSaMP 算法等。使得重构算法在时间复杂度方面获得了很大改善。

一种算法重构性能的好坏在于其是否有较低的时间复杂度和较高的重构精度。现有的相关算法中，在满足时间复杂度较低的时候重构精度会下降，但在保证重构精度的前提下往往又需要较高的复杂度，给后期的重构运算带来很大代价。所以怎样找到一种在保证较高重构精度的条件下让运算时间更少的重构算法，一直是国内外研究者的主要任务。

1.3 本文的主要研究内容和结构安排

本文主要工作在于对压缩感知重构算法进行研究，并利用一维和二维原始信

号对重构算法进行仿真实验，通过这些实验所得数据，分析比较各个算法的性能。

本文的内容安排如下：

第一章是绪论部分，介绍了本课题研究的相关背景情况及意义，阐述了国内外学者对该领域的研究趋势。

第二章是本文理论部分，讲述了信号的稀疏表示方法、采样矩阵的研究以及数据重构三个部分。

第三章是本文的主要部分，对常用的重构算法进行详细的理论分析，并利用MATLAB进行仿真实验，对比分析各个算法之间的复杂度和精度。

第四章是总结与展望部分。回顾本文所做的工作，总结研究内容，分析其中的不足。

第二章 压缩感知基本理论

2.1 信号的稀疏表示

压缩感知要求信号必须是稀疏的。然而常见的信号通常都含有大量的冗余信息，对这类信号做处理时，会因为采样数据的增加而带来较大的成本代价。虽然原始信号并不是稀疏的，但可以借助于一些正交变换基，将原始信号投影到正交基上，得到在变换域中表现为稀疏的数据。目前常用的稀疏表示方法如下：

(1)傅里叶变换

傅里叶变换实现了信号由时域到频域的转换，并借助于一系列正弦函数对原始信号进行线性组合和逼近，使信号的处理变得更加的灵活和容易。傅里叶变换虽然改变了信号的时域表示，但在频域中表示的信息与原信号所携带的信息并没有差别，不同的仅仅是表示的方式而已。有了信号的频域表示后，如果需要对频域信号进行离散化处理，这时就需要借助离散傅里叶变换（DFT）。离散傅里叶变换可以看做是傅里叶变换的离散采样，对于一个长度为 M 的有限长信号 x(n),其 N 点离散傅里叶变换（DFT[x(n)]）及离散傅里叶逆变换（IDFT[x(x)]）如下：

$$\begin{cases} X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2k\pi n}{N}} \quad \text{其中 } k = 0,1,\ldots\ldots,N-1 \quad (2\text{-}1) \\ x(n) = IDFT[X(k)] = \frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{-j2\pi k(m-n)}{N}} = \begin{cases} 1 & m = n + iN,\ \text{i为整数} \\ 0 & \text{m} \neq n + iN,\ \text{i为整数} \end{cases} \quad (2\text{-}2) \end{cases}$$

(2)离散余弦变换

同傅里叶变换一样，离散余弦变换(DCT)也是正交变换，但与傅里叶变换相比，DCT 变换不再限制于复数的运算，而是可以在实数域进行计算，在图像处理的相关方面，也比傅里叶变换具有更好的性能。对于一维和二维信号，DCT 变换及其反变换公式如下：

其中一维信号为 f(x),x=0,1,…N-1,二维信号为 f(x,y),x,y=0,1,…N-1。

正变换
$$\begin{cases} F(0)=\dfrac{1}{\sqrt{N}}\sum_{x=0}^{N-1}f(x), \quad u=0 & (2\text{-}3) \\ F(u)=\sqrt{\dfrac{2}{N}}\sum_{x=0}^{N-1}f(x)cos\left[\dfrac{\pi}{2N}(2x+1)u\right], \quad u=1,2\ldots N-1 & (2\text{-}4) \end{cases}$$

反变换
$$f(x)=\frac{1}{\sqrt{N}}F(0)+\sqrt{\frac{2}{N}}\sum_{u=1}^{N-1}F(u)cos\left[\frac{\pi}{2N}(2x+1)u\right], x=1,2\ldots N-1 \quad (2\text{-}5)$$

正变换
$$\begin{cases} F(0,0)=\dfrac{1}{\sqrt{N}}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y), \quad u=0,v=0 & (2\text{-}6) \\ F(u,0)=\dfrac{2}{\sqrt{N}}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\dfrac{\pi}{2N}(2x+1)u\right] & (2\text{-}7) \\ \quad v=0,u=1,2\ldots N-1 & \\ F(v,0)=\dfrac{2}{\sqrt{N}}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\dfrac{\pi}{2N}(2y+1)v\right] & (2\text{-}8) \\ \quad u=0,v=1,2\ldots N-1 & \\ F(u,v)=\dfrac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\dfrac{\pi}{2N}(2x+1)u\right]\cos\left[\dfrac{\pi}{2N}(2y+1)v\right] & \\ \quad u,v=1,2,\ldots N-1 & (2\text{-}9) \end{cases}$$

反变换
$$\begin{aligned} f(x,y)=&\frac{1}{\sqrt{N}}F(0,0) \\ &+\frac{2}{\sqrt{N}}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}F(u,0)\cos\left[\frac{\pi}{2N}(2x+1)u\right] \\ &+\frac{2}{\sqrt{N}}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}F(0,v)\cos\left[\frac{\pi}{2N}(2y+1)v\right] \\ &+\frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}F(u,v)\cos\left[\frac{\pi}{2N}(2x+1)u\right]\cos\left[\frac{\pi}{2N}(2y+1)v\right] \end{aligned} \quad (2\text{-}10)$$

(3) 小波变换

小波变换是一种在信号局部分析中被广泛运用的变换方法，与傅里叶对信号的全局表示不同，小波变换可以很好的反应信号在某一局部的特性，弥补了傅里叶变换在这方面的不足。二维离散小波相关变换公式如下：

$$W_{\varphi}(j_0,m,n)=\sqrt{\frac{1}{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\,\varphi_{j_0,m,n}(x,y) \tag{2-11}$$

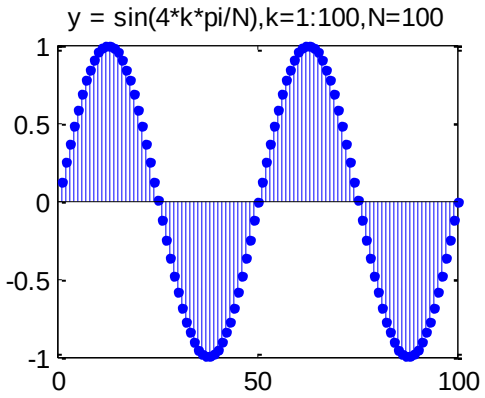
$$W_{\varphi}^{i}(j_0,m,n)=\sqrt{\frac{1}{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\,\psi_{j_0,m,n}^{i}(x,y),i=[H,V,D] \tag{2-12}$$

$$f(x,y)=\sqrt{\frac{1}{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}W_{\varphi}(j_0,m,n)\,\varphi_{j_0,m,n}(x,y) \tag{2-13}$$

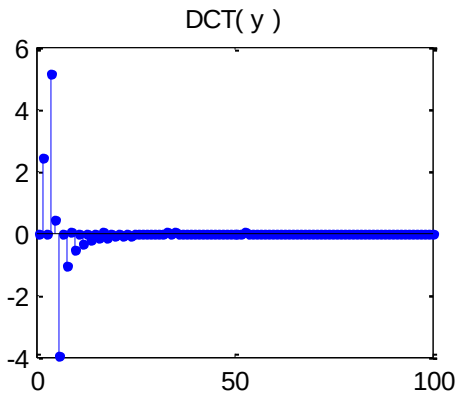
在式 2-12 中，$i=[H,V,D]$表示 3 个不同方向，$\varphi_{j,m,n}$和$\psi_{j,m,n}^{i}$如下：

$$\begin{cases}\varphi_{\mathrm{j},m,n}(\mathrm{x},\mathrm{y})=2^{j/2}\varphi\left(2^{j}x-m,2^{j}y-n\right) & (2\text{-}14)\\ \psi_{j,m,n}^{i}(x,y)=2^{j/2}\psi^{i}\left(2^{j}x-m,2^{j}y-n\right) & (2\text{-}15)\end{cases}$$

下面，以 DCT 变换为例，说明其在信号稀疏表示中的应用，如图 1 所示。



(a)一维信号 y



(b)一维信号 DCT 变换

(c) 原始图像

(d)DCT 变换重构图像

图 1 信号的 DCT 变换

图 1-(a)是对一维信号$y = \sin(4\pi t)$的连续采样，在两个周期的时间里获得了 100 个采样数据，由图(a)可知，该信号在时域中的非零数值较多，不满足压缩感知采样的要求。图 1-(b)是利用 DCT 变换对采样的数据进行处理的结果，采样数据做 DCT 变换后，绝大多数的数值都变为零，非零值的个数很少,于是便实现了非稀疏信号的稀疏表示。

图 1-(c) 是一幅 256×256 大小的 8 位灰度 bmp 格式图像，如果要无失真的保存该图像需要记录 65536 个像素点的像素值。如果对该图像做 DCT 变换，并将变换后的数据中数值趋于零的数设置为零，这里选择的条件为绝对值小于 15。再利用更改后的数据对图像进行重构，得到的重构结果如图 1-(d)所示。在进行离散余弦变换后非零点的个数为 19433，在保存的时候只需要保存这些非零点的值，极大的减少了存储的信息量，但重构的图像与原始图像差别却很小。

这样利用正交变换基对非稀疏信号进行稀疏表示，不仅减少了数据存储带来的成本代价，而且还满足了压缩感知理论对信号稀疏性的要求，在包括数据压缩在内的很多领域都有重大意义。

2.2 采样矩阵的研究

与传统采样理论不同，压缩感知打破了先采样后压缩的限制，通过采样矩阵实现了在采样的同时进行压缩。对于一个已知信号$x \in R^N$,稀疏度为 K，即 x 的 N 个数值中只有 K 个非零值，其中$K \ll N$。如果利用传统采样定律进行采样，为了不丢失信号 x 中的元素，需要获得 N 个测量值，这时可以理解为采样矩阵是 N 阶

单位矩阵。如图 2 所示.

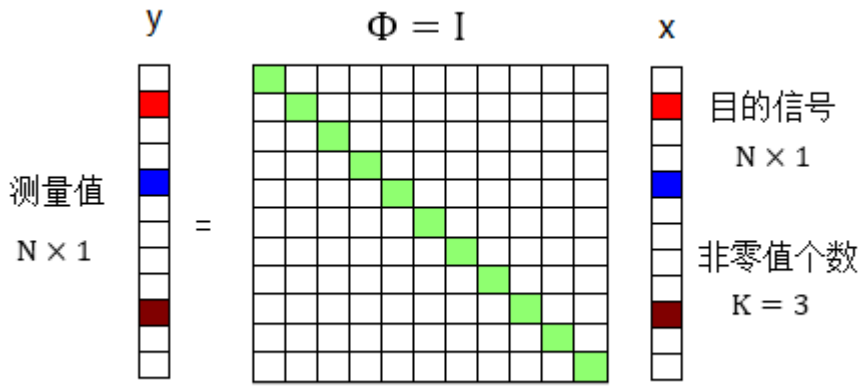


图 2 传统采样数学模型

在图 2 中，目的信号 x 共有 N = 12 个元素，其中非零值的个数为 K = 3,采样矩阵Φ为 12 阶单位矩阵，这样通过Φ和 x 得到的测量值 y 就包含了 x 中的所有信息，但除了非零数据，其他的都是多余的，如果对这些数字为零的数据也进行采样保存必然造成巨大的资源浪费。

解决这一问题的关键在于采样矩阵的选取，能否构造这样一个矩阵，使得采样后得到的数据 y 中只包含 x 的非零数据的信息，或者很少包含其他冗余数据信息。y 不一定是 x 中的所有非零元素的集合，而是 x 非零信息的一种表示，包含了 x 的全局信息。采样过程如图 3 所示。

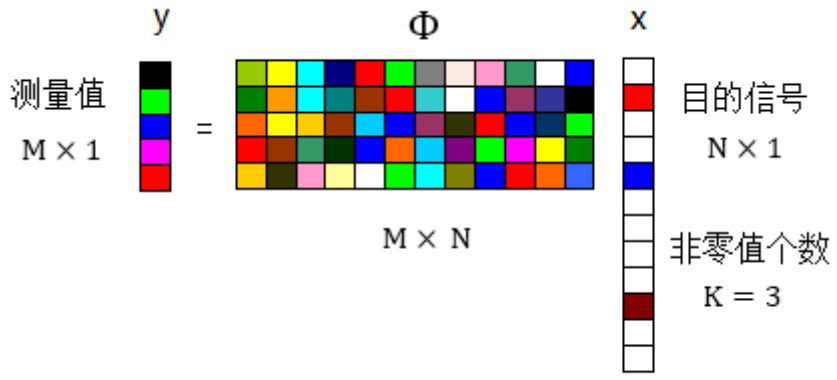


图 3 压缩感知采样数学模型

在图 3 中，对于稀疏信号 x 选取一个大小为$M \times N$的矩阵Φ作为采样矩阵。得到的数据 y 中共含有 M 个测量值，通常$M \ll N$。这样通过新的采样矩阵Φ将 x

的信息保存在了 y 中，并且 y 里面的测量值远小于原始信号 x 的元素个数，从而实现了对信号的压缩。

在上面的讨论中，默认信号 x 是稀疏的，其绝大多数元素的值都为零，给出了稀疏度 K。然而现实世界中的自然信号往往是非稀疏的，这时就需要利用 2.1 节中的信号稀疏表示方法对原始信号做处理，让目的信号在某个正交变换基Ψ下具有稀疏性，即$x = \Psi\alpha$。此时的采样模型如下：

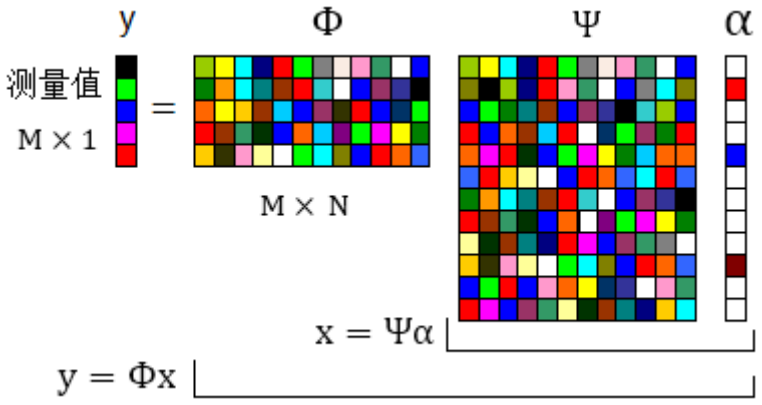


图 4 非稀疏信号采样模型

采样矩阵的好坏在于是否能够保存信号 x 中的有效信息，并且保证能够基于感知数据 y 对目的信号实现重构。为了实现这一点，采样矩阵通常需要具有一定的条件。下面以随机高斯矩阵为例，探讨其在采样和重构过程中表现出来的特性。

高斯矩阵元素满足高斯分布：

$$\Phi_{i,j} \sim N(0, \frac{1}{\sqrt{M}}) \qquad (2\text{-}16)$$

对于一幅 256×256 大小的 8 位 bmp 格式图像，稀疏度确定，利用高斯矩阵对其进行采样，并将采样的数据用 OMP 算法进行重构。以测量数 M 为自变量，观测高斯矩阵在变化的 M 值下重构结果的信噪比。由于高斯矩阵的随机性，每次测量时生成的数据不一定完全一样，为了避免偶然误差，对每个测量值做 300 次重复试验，对所有的数据求平均值，得到的结果如下：

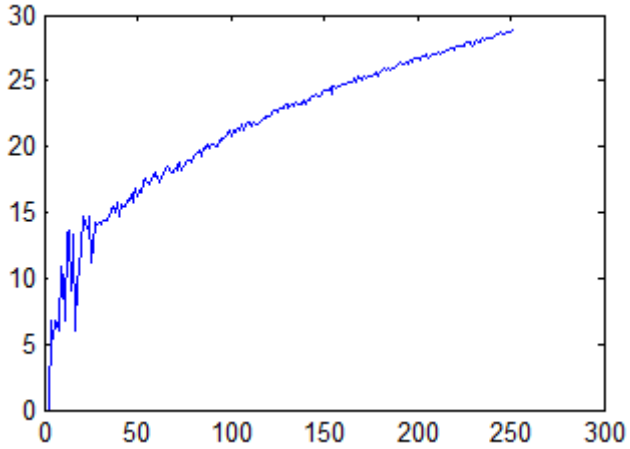


图 5 高斯采样矩阵恢复信噪比

由图 5 可知，在测量数 M 不断增大的过程中，重构的信噪比不断增大，重构效果更好。

2.3 压缩感知数据重构

信号$x(N\times 1)$，经过矩阵$\Phi(M\times N)$采样后得到$y(M\times 1)$，即$y=\Phi x$。因此，由采样数据 y 和采样矩阵Φ重构数据 x 的过程就是求方程$y=\Phi x$的解的过程。但在式$y=\Phi x$中，y 是长度为 M 的向量，x 是长度为 N 的向量，并且$M\ll N$，此方程属于欠定方程，在有解的情况下需要穷举信号 x 中的C_N^K种非零值的排列可能。因此，当 N 和 K 的值都较大时直接求解信号 x 是非常困难的。

压缩感知要求信号必须是稀疏的，这样，在求解$y=\Phi x$的时候，加上适当的限定条件，让重构的信号$\hat{x}$尽可能的稀疏，便能得到原始信号 x 的近似解。理论证明，可以通过求解最优l_0范数完成对信号 x 的求解：

$$\hat{x}=arg\ \ min\parallel x\parallel_0\ \ s.t\ \ \Phi x=y \qquad (2\text{-}17)$$

上式中，s.t 代表 subject to，信号 x 为稀疏度为 K 的稀疏信号，并且为了实现精确重构信号，测量结果 y 中的测量数 M 必须满足以下条件：

$$M=O(K\lg(N)) \qquad (2\text{-}18)$$

在求解式 2-17 的方法中，通常采用的是贪婪类算法，这些算法通过迭代的方式从感知矩阵中寻找与原始信号 x 最匹配的原子，并利用这些原子对信号进行稀疏表示，不停的向 x 逼近，并求出每次迭代的余量，然后针对余量继续进行最

为匹配原子的寻找。由于式 2-17 在求解的过程中带来的复杂性和不稳定性，贪婪类算法虽然能通过迭代的方式求出 x 的近似解，但重构的精度较低，在精度要求较高的情况下可以使用l_1范数代替，并且放宽约束条件，即：

$$\hat{x} = arg \ \ min \| x \|_1 \ \ s.t \ \ \Phi x = y \qquad (2\text{-}19)$$

这样，在知道 y 和Φ的情况下式 2-17 就转化成了一个凸最优化问题。通过一定的约束条件，进而转变为线性规划相关问题，完成从完备的基字典里找出原始信号 x 的稀疏表示，实现对压缩感知数据的重构。

第三章 压缩感知重构算法研究及仿真

3.1 基追踪算法(BP)

由 2.3 节可知，信号重构的过程转化为了一个求最小l_0范数的问题。但由于l_0范数问题是 N-P 问题，在实际应用的过程中几乎不可行，因此，通常利用l_1范数替代，通过求解最小l_1范数来实现对原始信号的重构。

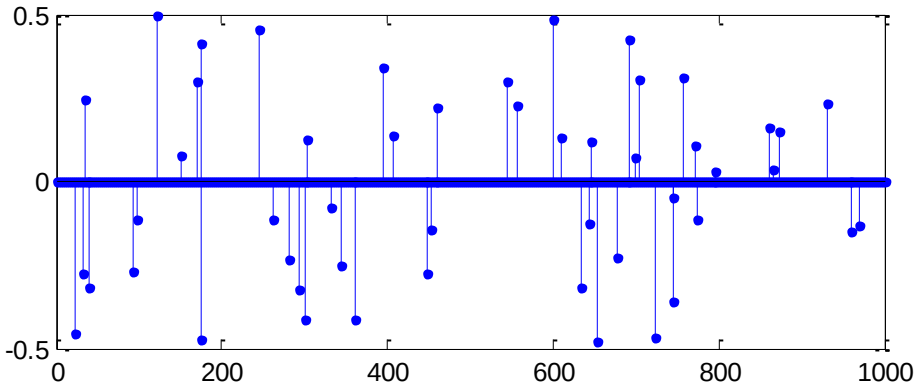
最小l_1范数法也被称为基追踪算法(Basis Pursuit,BP)。BP 算法通过迭代的方式从采样矩阵里寻找信号的最稀疏表示，用最少的基表示稀疏信号 x。考虑重构误差的因素，BP 算法可表示为：

$$\min \| \mathrm{x} \|_1 \quad s.t \quad \| x - y \|_2 \ll \varepsilon \qquad (3\text{-}1)$$

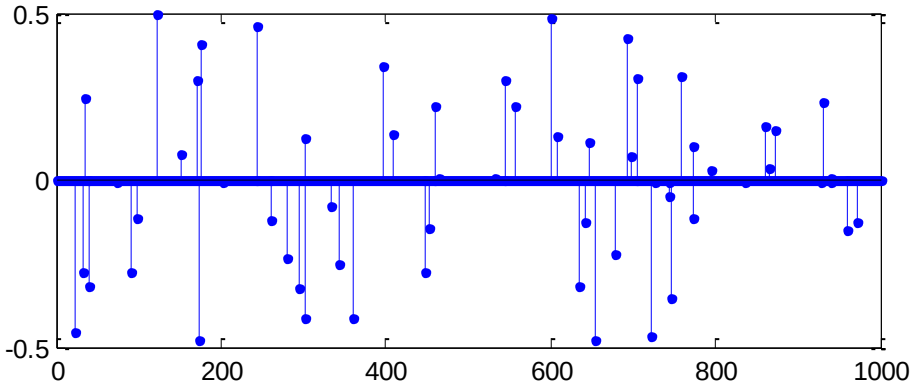
下面，利用 matlab 仿真软件，研究 BP 算法在一维和二维信号重构方面的特性。

(1)一维信号 BP 算法仿真实验

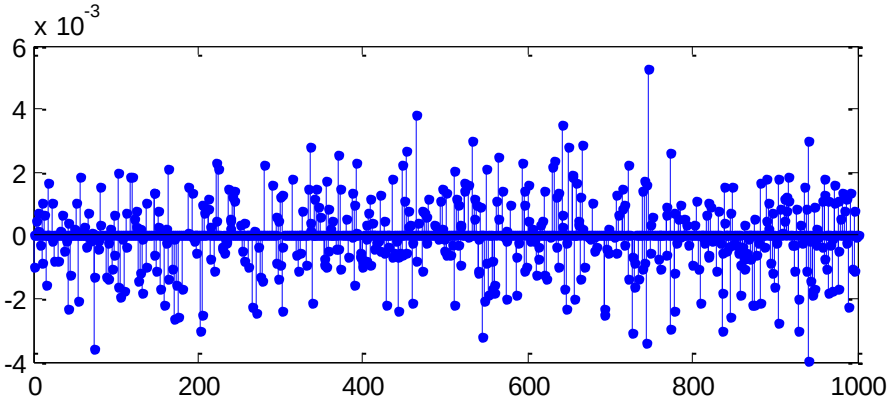
给定待测信号 x，其长度为 N=1000，并且 x 为稀疏度为 K=50 的稀疏信号，按采样率 M/N = 0.5 对信号 x 进行处理，获得测量数为 M=50 的采样数据 y，其中采样矩阵选择高斯矩阵 $\Phi(\mathrm{M} \times \mathrm{N})$，然后根据得到的采样数据 y，和高斯矩阵 Φ，利用 BP 算法进行重构，得到的实验结果如下：



(a)原始信号(x)



(b)重构信号(rec_x)



(c)重构误差(rec_x-x)

图 6 一维信号 BP 算法仿真结果

图 6-(a)为原始信号 x，其离散值随机分布在[-0.5,0.5]之间，图 6-(b)是经过基追踪算法重构后得到的信号 rec_x。图 6-(c)是原信号与重构信号的误差 recv_x-x,其中重构的绝对误差为 0.0258，相对误差为 0.0135。从图 6 可以看出，利用 BP 算法进行重构后，得到的重构结果与原来的信号几乎没有差别。

(2)二维信号 BP 算法仿真实验

以二维灰度图像作为仿真原始信号 x，高斯随机矩阵Φ作为采样矩阵，此处共选取 3 幅大小为 256×256 的 bmp 格式图像，仿真执行环境为：WINDOWS 7 64 位，CORE i5 CPU，MATLAB 8.3。仿真结果如下：

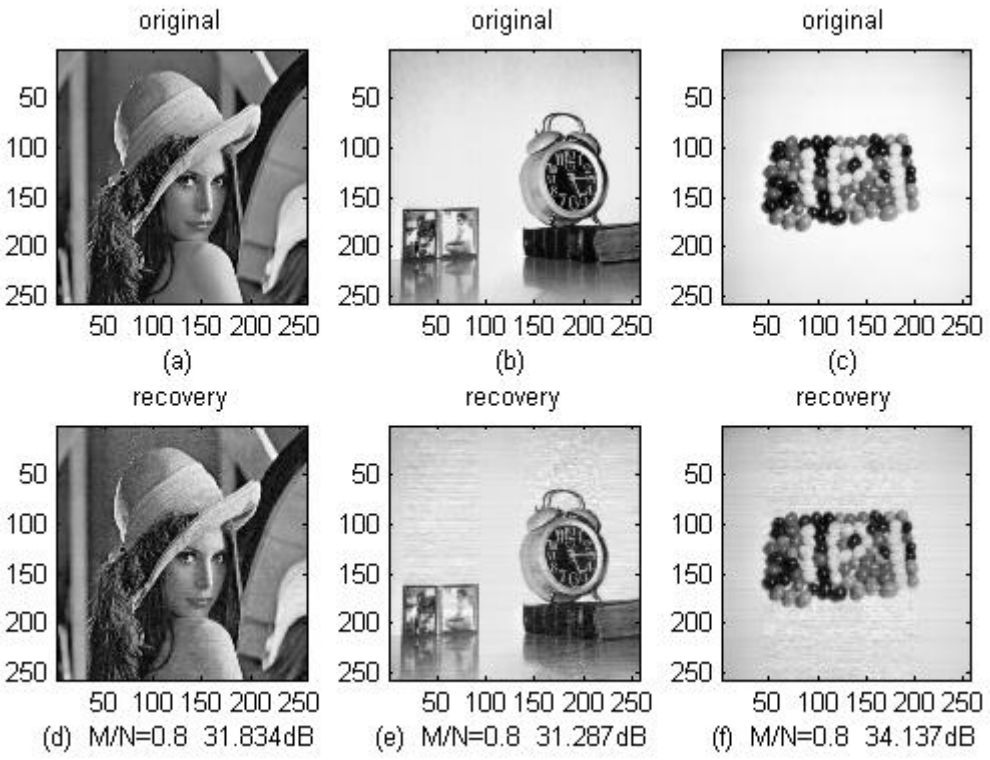


图 7 二维信号 BP 算法仿真结果

在上图中，(a)-(c)是原始图像"lena"，"clock"，"jellbeans"，(d)-(f)是相对应的重构图像，对原图像进行采样时选择的采样矩阵大小为 204×256，压缩比 M/N=0.8。记录重构的峰值信噪比(PSNR)。其中三幅图像重构的 PSNR 分别为：31.780dB，30.641dB，34.228dB。

由图 7 可以看出，当压缩比 M/N 为 0.8 的时候，利用高斯采样矩阵和 BP 算法能够很好的实现对原图像信号的重构，重构的精度较高。

上面所讨论的是 BP 算法在不同的原始图像信号下，以确定的测量数和压缩比对图像进行重构，下面探讨 BP 算法以不同测量数进行重构的时候所表现出来的特性，结果如图 8 所示。

在图 8 中，选取"jellbeans"作为原始图像信号，压缩比 M/N 取值分别为 0.4，0.5，0.6，0.7，0.8，分别以这些压缩比进行图像的重构，由图 8 可知，重构的 PSNR 跟压缩比是正相关的，并且图像重构的效果也随着 M/N 的增大而更加精确，但是由于测量数 M 的变大，BP 算法需要更长的运行时间。实际运用时需要根据对重构精度和运行时间的要求选择合适的压缩比。

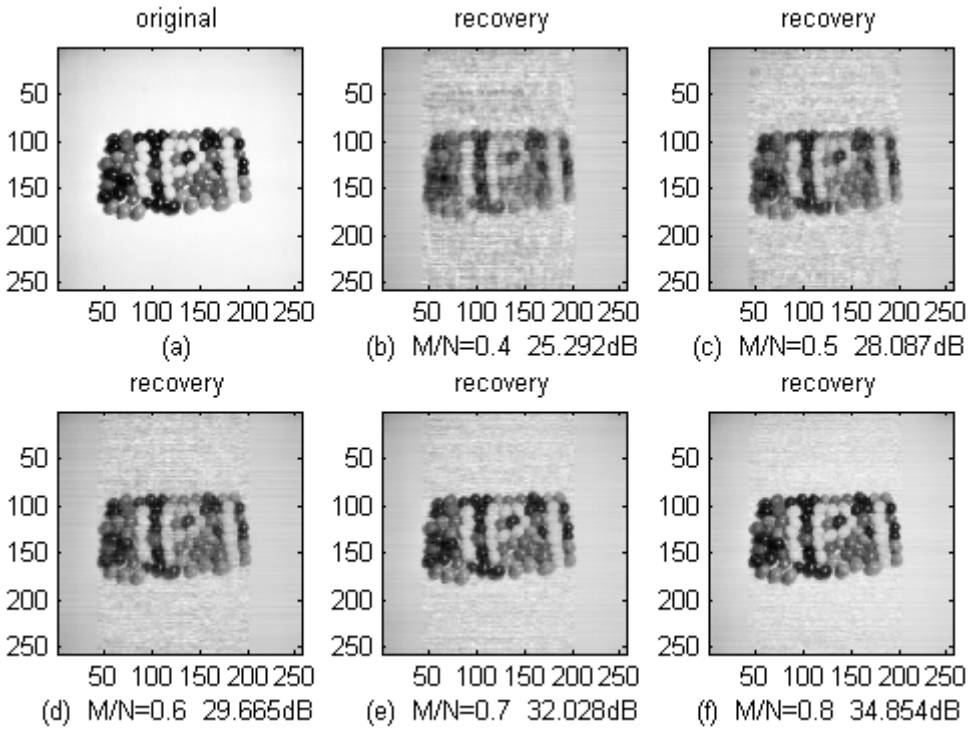


图 8 不同压缩比下的 BP 算法仿真结果

3.2 匹配追踪算法(MP)

如果原始信号 x 是稀疏的,则采样数据可直接由采样矩阵和 x 得到:y = Φx;但是当 x 是非稀疏信号时，需要利用正交变换基对非稀疏信号 x 进行稀疏表示，这时采样数据为：y = ΦΨα。通常将ΦΨ称为感知矩阵。匹配追踪算法在对信号进行重构的时候,将感知矩阵视为字典,感知矩阵中的每一列元素视为原子信号,然后对这些信号进行选择组合，从而完成对信号 y 的描述。

MP 算法在重构的时候通过迭代的方式每次从感知矩阵中找出与信号最为匹配的原子信号组合，对信号进行逼近，并得到信号与原子基之间的残差，找到残差之后继续寻找跟其最匹配的原子信号,重复这一步骤直到信号由一些原子信号线性表示。

MP 算法详细步骤如下:(y：测量数据，ϕ：感知矩阵，r_0：初始余量，k：迭代计数器，Λ：为索引集合，$\hat{x}$：重建信号，K：稀疏度)

(1)令r_0 = y，$\hat{x}_0 = 0$，$\Lambda = \emptyset$，k = 1;

(2)求系数$\lambda_k = arg_\lambda max\left\{\frac{<r_{k-1},\ \phi_\lambda>\phi_\lambda}{\|\phi_\lambda\|^2}\right\}$;

(3)索引集$\Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\}$;

(4)$\hat{x} = \hat{x} + \frac{<r_{k-1},\ \phi_\lambda>\phi_\lambda}{\|\phi_\lambda\|^2}$;

(5)余量 r = r - ϕx;

(6)如果 k<K，跳转到步骤(2)，否则输出重构信号$\hat{x}$。

算法流程图如下：

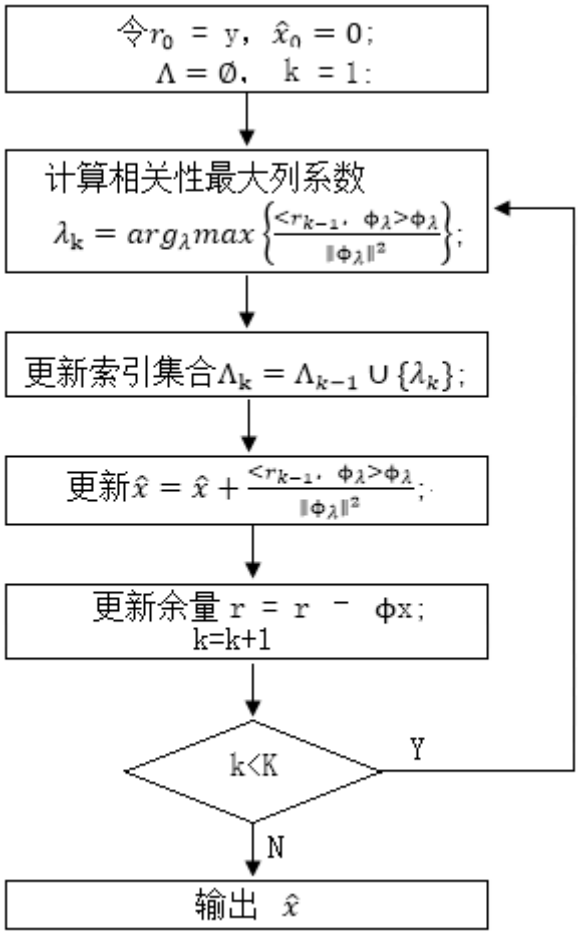
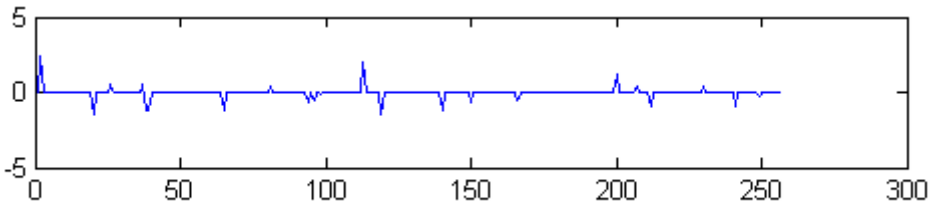


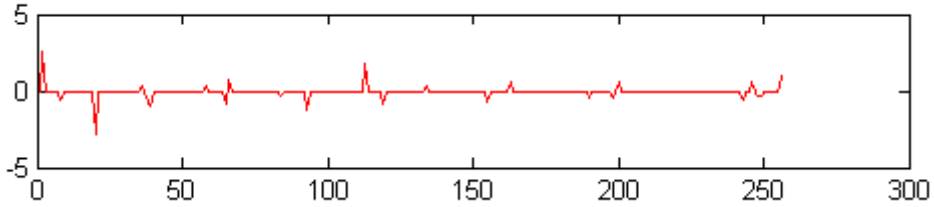
图 9 MP 算法流程图

下面给出 MP 算法在一维目的信号下的仿真结果。

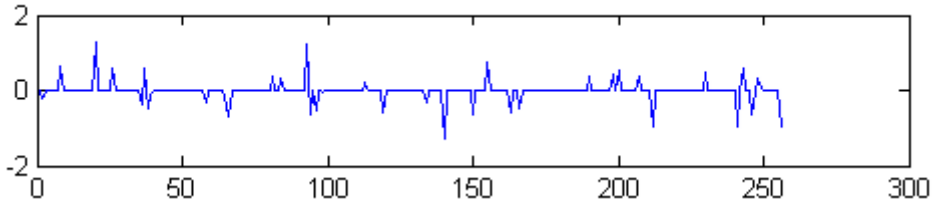
给定原始稀疏信号 x，长度为$N = 256$，稀疏度为K=23，采样测量数据为M=80，压缩比为 M/N=0.3，采样矩阵选择高斯矩阵。从图 10-(c)可以看出 MP 算法不能保证重构的误差足够小，重构精度不是特别高。另外在二维图像信号的重构中，MP 算法需要很多的循环计算步骤，给整个重构过程带来了很大的时间代价。因此，通常采用的是正交匹配追踪算法而不是 MP 算法。



(a)原始信号(x)



(b)重构信号(rec_x)



(c)重构误差(rec_x-x)

图 10 MP 算法一维仿真结果

3.3 正交匹配追踪算法(OMP)

与 MP 算法一样，正交匹配追踪算法(OMP)在对信号进行重构的时候，将感知矩阵视为字典，感知矩阵中的每一列元素视为原子信号然后选用一个原子信号进行信号的逼近，获得残差，然后不断的寻找跟残差最匹配的列元素，直到该信号可以由一些列元素线性表示。但 MP 算法的迭代次数较大，需要很大的循环次数才能逼近原始信号。为了解决这一问题，OMP 算法通过格拉姆-施密特正交化的方式生成一个正交的列集合，这样每次更新残差的时候都是和正交的列进行比较，大大的减少的循环次数。

OMP 算法详细步骤如下:(y：测量数据，A：感知矩阵，r_0：初始余量，n：迭代次数，t：迭代计数器，Λ，J：索引集合，$\hat{x}$：重建信号，K：信号稀疏度)

(1)令r_0 = y，$\hat{x}_0 = 0$，$\Lambda = \emptyset$，n = 2*K，t = 1;

(2)r_0与 A 的内积：$C_k = A^T r_{k-1}$;

(3)求 max{C_k},及其位置 P;

(4)$\Lambda_k = \Lambda_{k-1} \cup \{P\}$，$A_{\Lambda_k} = A_{\Lambda_{k-1}} \cup \{A(:,P\}$;

(5)求解方程$\hat{x}_k = (A_{\Lambda_k}^T A_{\Lambda_k})^{-1} A_{\Lambda_k}^T y$;

(6)r_k = y - A$\hat{x}_k$，t = t + 1;

(7)如果迭代次数小于 n，跳转到步骤(2)；否则算法停止，输出$\hat{x}$ = $\hat{x}_k$。

算法流程图如下：

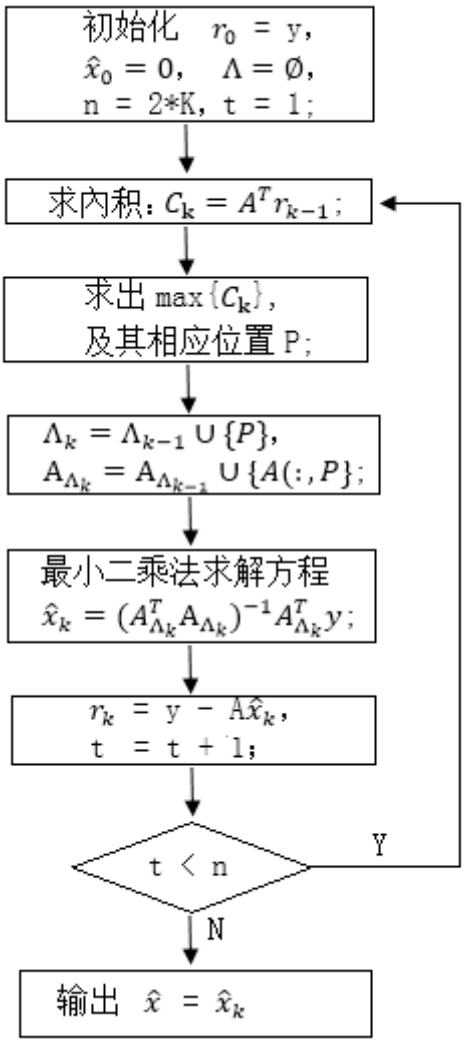


图 11 OMP 算法流程图

下面给出 OMP 算法在一维和二维信号下的仿真结果

(1)一维信号 OMP 算法仿真实验

一维信号仿真结果如下：

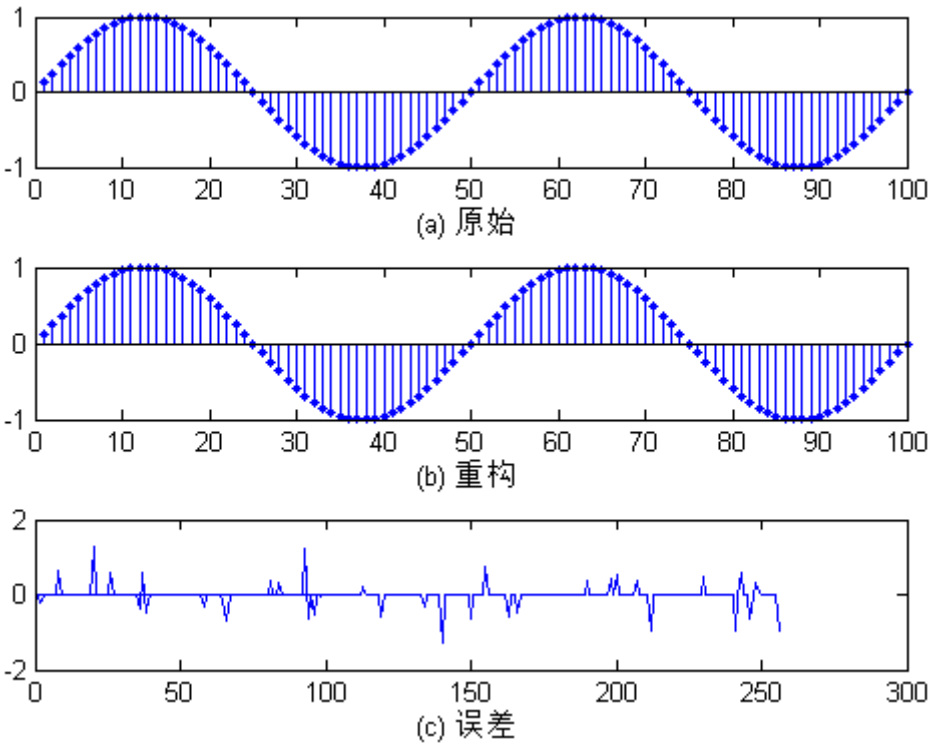


图 12 一维信号 OMP 算法仿真

图12-(a)是长度为100的离散信号x(sin(4*k*pi/N)， 其中k = 1:100; N = 100)，图12-(b)是OMP算法重建后的信号$\hat{x}$。采样过程中的的矩阵为高斯矩阵，正交变换基为傅里叶变换矩阵，正交变换后的稀疏度为K=20，采样测量值为M = 50。从图12-(c)中可以看出，当用OMP算法进行仿真重构时，重构带来的误差几乎可以忽略，重构的精度很高。

(2)二维信号OMP算法仿真实验

对二维信号进行仿真时所选取的原始信号和仿真执行的环境同3.1节。仿真结果如图11所示。

图13中(a)～(f)分别给出了原始二维信号及经过OMP算法重构后的信号。三种信号仿真中的压缩比都为0.8，对应的PSNR为26.765dB，26.322dB和32.715dB。可以看出OMP算法的仿真效果比BP算法要差，但OMP算法的执行时间更短。

下面探讨OMP算法以不同测量数进行重构的时候所表现出来的特性，结果如图14所示。由图可知，重构精度随压缩比的增大变得更高。

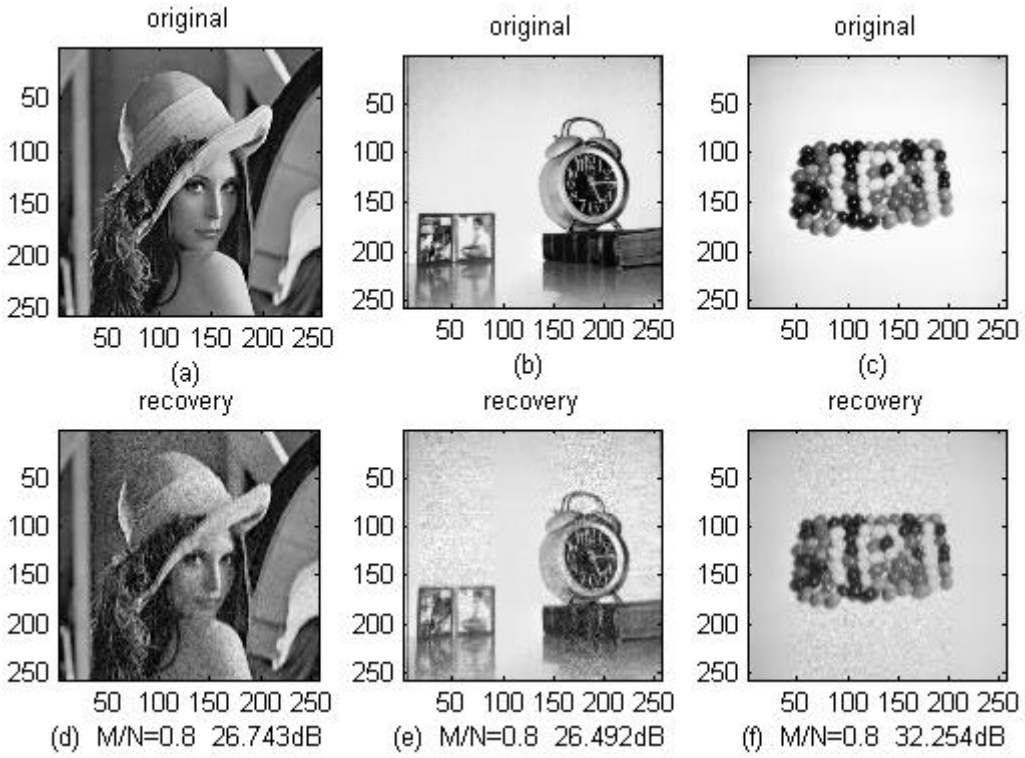


图13 二维信号OMP算法仿真

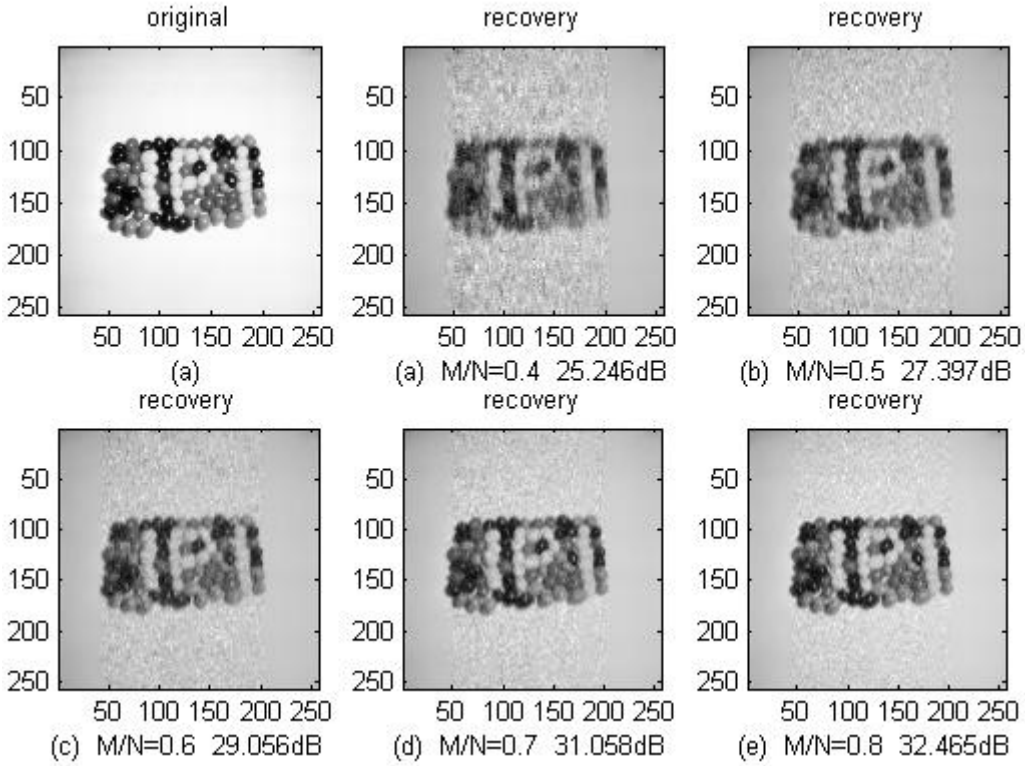


图 14 不同压缩比下的 OMP 算法仿真结果

3.4 子空间追踪算法(SP)

子空间追踪算法(SP)在执行的时候，需要根据感知矩阵的 K 个列向量生成子空间，确定测量信号 y 存在于哪个子空间中。SP 算法维护了一个索引集，初始值为与信号 y 的 K 个最大相关列，然后在每次的迭代过程中检测这 K 个列向量的子集，如果重构信号与测量信号之间的余量较大，则需要更新这个列表。这样，SP 算法会保留可靠地候选值，丢弃不可靠的候选值，在丢弃的时候同时增加相同数量的新值来更新列表，更新的前提是新的子空间重构后带来的余量要比原来的子空间产生的余量要小，于是，SP 算法就能在每次的迭代中找到比上一次更好的子空间，完成对原始信号的重建。

SP 算法详细步骤如下：(y 为测量数据，Φ为感知矩阵，y_0为初始余量，m 为迭代次数，l 为迭代计数器，T为索引集合，$\hat{x}$为重建信号，K 为信号稀疏度)

(1)$T^0 = arg_K \max(|\Phi^T|)$;

$y_r^0 = y - \Phi_{T^0}(\Phi_{T^0}^+ y)$;

(2)$\hat{T}^{l-1} = arg_K \max(|\Phi^T y_r^{l-1}|), \tilde{T}^l = T^{l-1} \cup \tilde{T}^{l-1}$;

(3)$T^l = arg_K \max\left(\left|\Phi_{\tilde{T}^l}^+ y\right|\right)$;

(4)$y_r^l = y - \Phi_{T^l}(\Phi_{T^l}^+ y)$; $y_r^l = y - \Phi_{T^l}(\Phi_{T^l}^+ y)$

(5)如果$\| y_r^l \|_2 \geq \| y_r^{l-1} \|_2$，则$T^l = T^{l-1}$，算法结束，输出$\hat{x}$；否则跳转到步骤(2)。

SP 算法流程图如下：

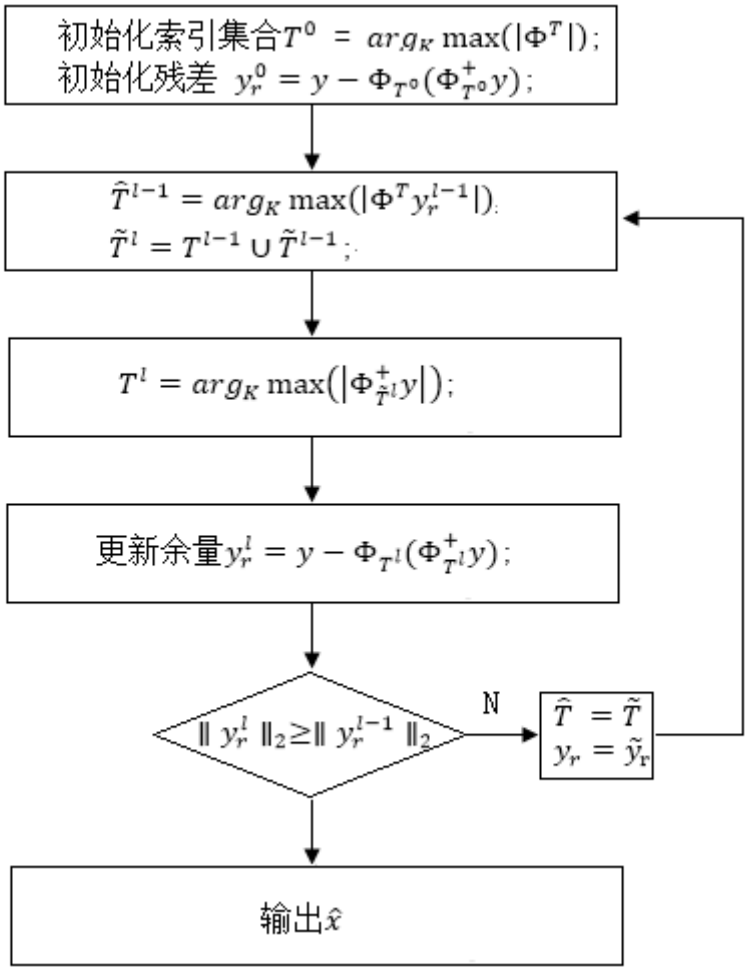


图 15 SP 算法流程图

下面给出 SP 算法在一维和二维信号下的仿真结果

(1)一维信号 SP 算法仿真实验

给定稀疏度为 K=50 的信号 x，长度为$N = 256$，采样矩阵为高斯随机矩阵，测量数据量为 M=204，压缩比为$N/M = 0.8$。仿真执行环境同 3.3 节。仿真实验结果如图 16 所示。

图 16-(a)～(c)分别为原始信号 x，重构信号$\hat{x}$以及两者之间的误差。在图(c)中，重构误差维持在10^{-16}数量级，重构的效果比 MP 算法要好。

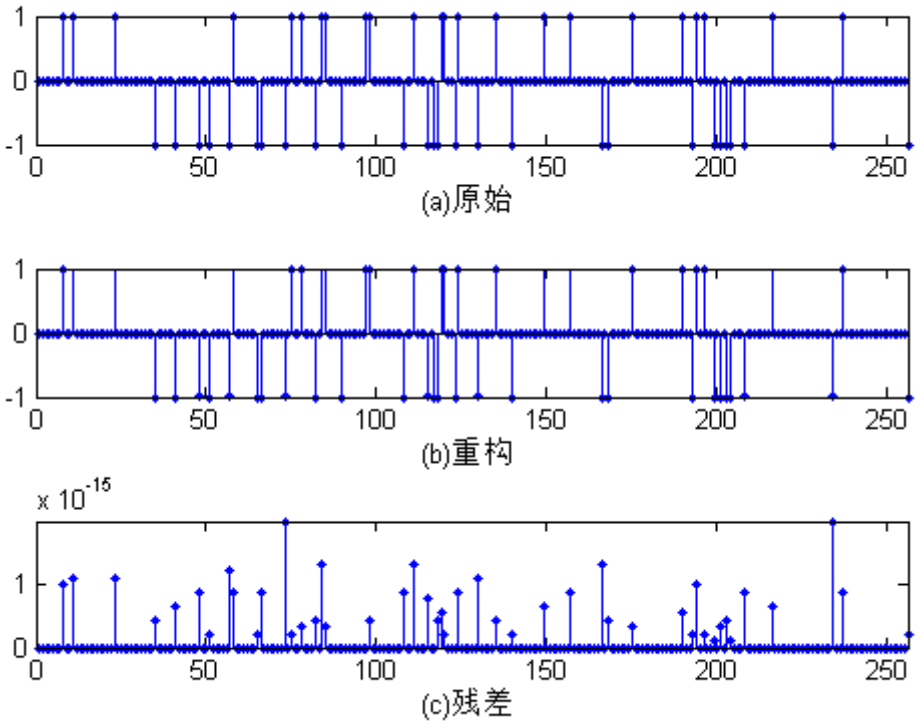


图 16 一维信号 SP 算法仿真

(2)二维信号 SP 算法仿真实验

以 245×256 大小的 8 位 bmp 格式灰度图像作为带采样信号 x，高斯随机矩阵作为采样矩阵，正交变换基选择 DCT 变换，按压缩比 N/M=0.8 进行采样测量，然后利用 SP 算法进行重构，记录每次重构的 PSNR。仿真实验结果如图 17 所示。

图 17-(d)～(f)为 SP 算法重构后的图像，对应的峰值信噪比为 27.739dB，27.611dB 和 33.728dB。从图 17 可以看出，在同样的压缩比和采样情况下，SP 算法在对二维图像信号重构的时候其精度要比 MP 算法高。

下面探讨 SP 算法以不同测量数进行重构的时候所表现出来的特性，结果如图 18 所示。从图可知，当压缩比不断增大的时候，峰值信噪比逐渐变大，信号重构的精度更高。

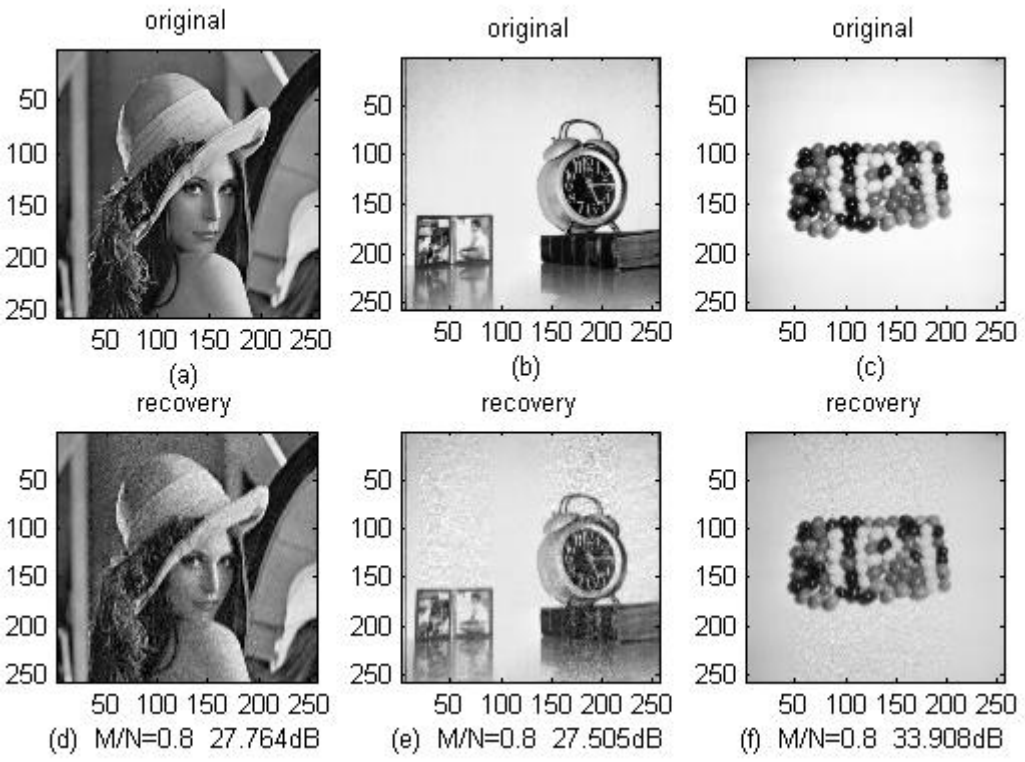


图 17 二维信号 SP 算法仿真

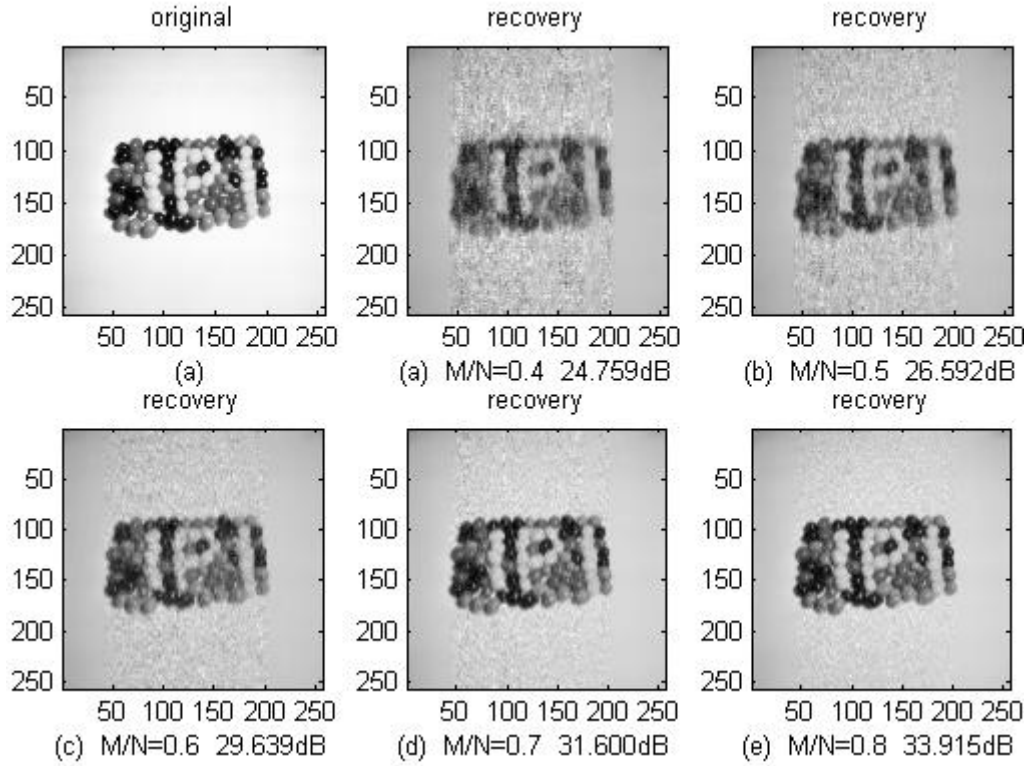


图 18 不同压缩比下 SP 算法仿真

3.5 压缩采样匹配追踪算法(CoSaMP)

在 MP 和 OMP 算法中,每次只选用一个原子信号进行信号的逼近,获得残差,算法的效率较低，在 SP 算法中，每次选取 K 个列向量，然后从这 K 个列向量的子空间中寻找最匹配原子基，这样就提高了收敛速度。CoSaMP 算法的主要思想也是不断的筛选原子基，并剔除部分原子，但是保证了索引集合里的原子个数小于等于 3K 个，被移除的原子个数小于等于 K 个，并且每次迭代索引集包含了 2K 个原子。

CoSaMP 算法详细步骤如下：(y：测量数据，Φ：感知矩阵，r_0：初始余量，k：迭代计数器，Λ，J：索引集，$\hat{x}$：重建信号，K：信号稀疏度)

(1)r_0 = y，k = 1，$\Lambda = \emptyset$，J = ∅;

(2)计算系数C_k = $\Phi^T r_{k-1}$;

(3)求出C_k中最大 2K 个值的索引存入 J;

(4)更新$\Lambda_k = \Lambda_{k-1} \cup J$;

(5)求解方程$\hat{x}_k = (\Phi_{\Lambda_k}^T \Phi_{\Lambda_k})^{-1} \Phi_{\Lambda_k}^T y$;

更新余量r_k = y - A$\hat{x}_k$;

(6)如果$|\Lambda| \geq 2K$，算法结束；否则 k=k+1，跳转到步骤(2)。

CoSaMP 算法流程图如下：

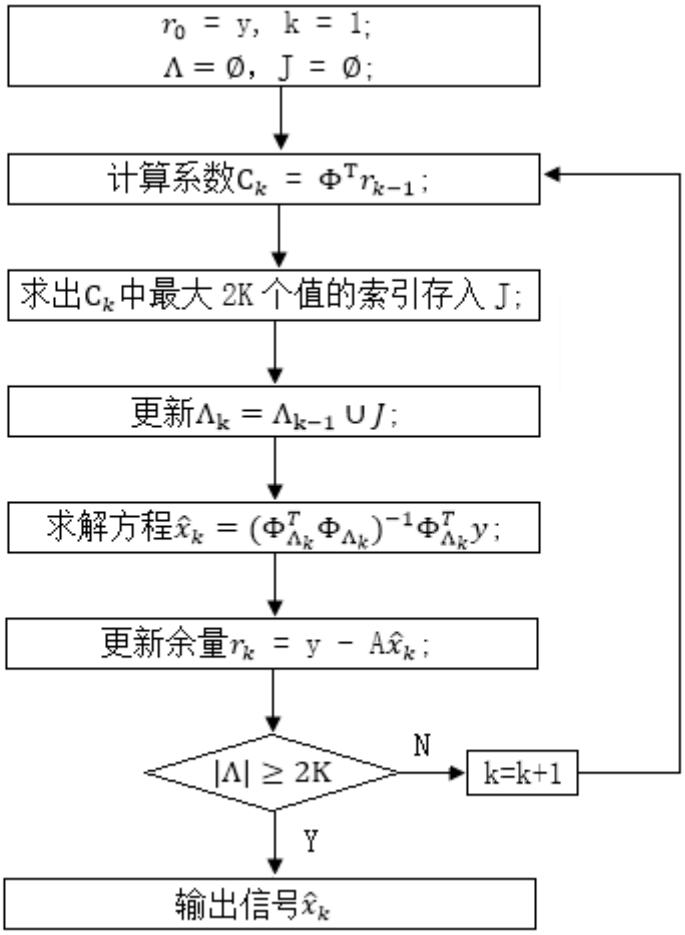


图 19 CoSaMP 算法流程图

(1)一维信号 CoSaMP 算法仿真实验

一维信号仿真时信号长度，测量压缩比，采样矩阵等条件都与 3.4 节相同，得到的结果如图 20 所示。从图 20-(c)可以看出，CoSaMP 算法的重构误差很小，数量级在10^{-16}～10^{-15}，近似为 0。

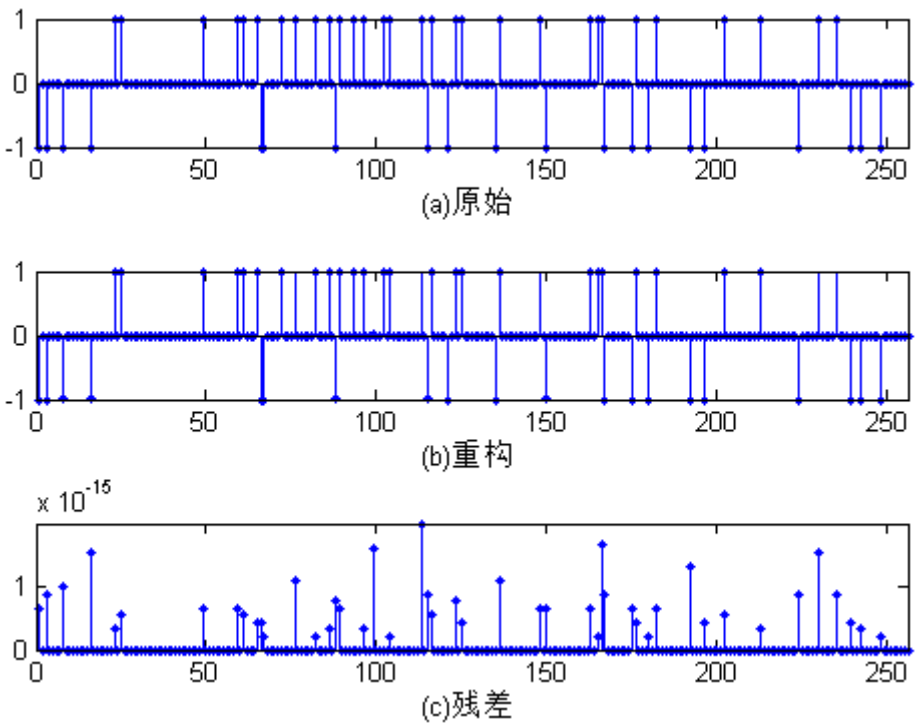


图 20 一维信号 CoSaMP 算法仿真

(2)二维信号 CoSaMP 算法仿真实验

选取三幅 256×256 大小的 bmp 格式图片，正交变换选择 DCT 变换，压缩比 M/N 为 0.8，并利用 CoSaMP 算法进行信号重构，结果如下图所示。

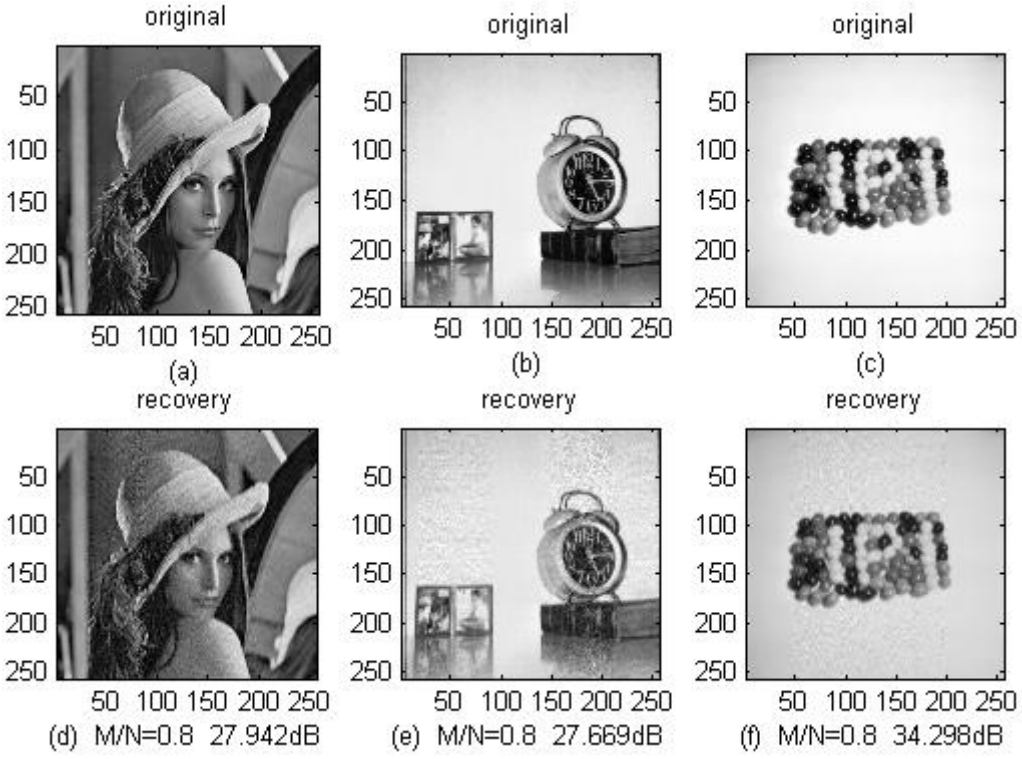


图 21 二维信号 CoSaMP 算法仿真

在同样的采样条件下，CoSaMP 算法在重构时的峰值信噪比与 SP 算法相差较小，两者的重构精度都比较高。下面给出了 SP 算法以不同测量数进行重构的时候所表现出来的特性，结果如图 22 所示。

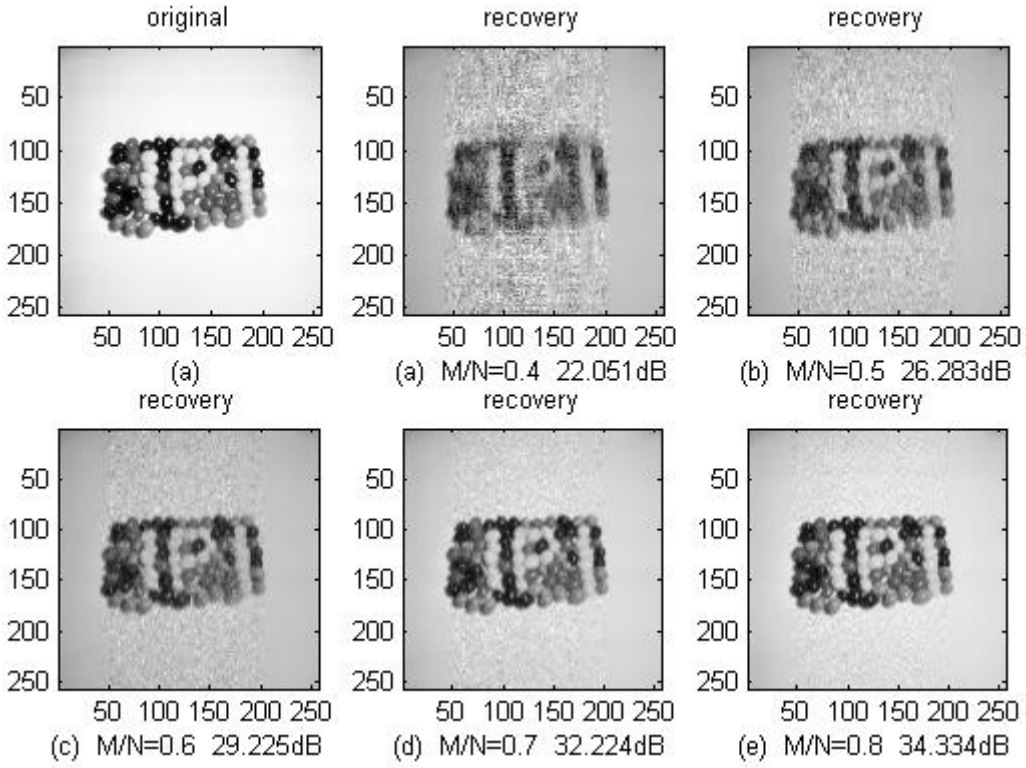


图 22 不同压缩比下的 CoSaMP 算法仿真结果

3.6 算法性能比较

一种算法重构性能的好坏在于其是否有较低的时间复杂度和较高的重构精度，时间复杂度越低带来的运行时间代价越小，重构精度越高带来的误差就越少。因此在对算法性能进行分析的时候，通过研究算法的运行时间的长短来评判算法时间复杂度的好坏，通过研究算法在恢复时的峰值信噪比和重构误差的大小来评判算法重构精度的高低。

本节将对以上的重构算法进行比较，基于运行时间、峰值信噪比和重构误差三个方面，分析各个算法在信号重构过程中的精度和时间复杂度，总结各个算法的优缺点。

(1)不同采样率下的运行时间比较

在表 1 中，给出了 BP 算法，OMP 算法，SP 算法和 CoSaMP 算法在相同的目的

信号不同的采样率下的运行时间，单位为秒(s)，采样率为 0.1～0.8。

在图 23 中，以压缩比 M/N 为横坐标，运行时间为纵坐标，绘制了不同算法运行时间随压缩比变化的二维曲线。从图中可以看出，BP 算法在运行时间上比 OMP 算法要长，CoSaMP 算法运行时间比 SP 算法长。

表 1 算法运行时间(s)

算法 / 采样率	BP	OMP	SP	CoSaMP
0.1	0.749	0.078	0.437	0.406
0.2	0.85	0.1795	0.858	1.014
0.3	1.014	0.3355	1.716	2.1915
0.4	1.24	0.5145	2.73	3.947
0.5	1.451	0.8035	4.438	6.6145
0.6	2.27	1.139	7.0825	10.249
0.7	2.7065	1.5675	9.5865	15.1555
0.8	3.229	2.0905	13.361	22.893

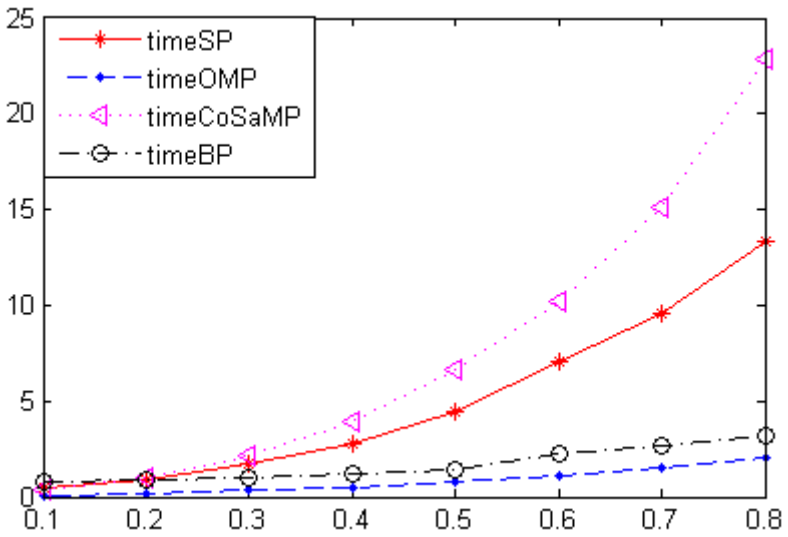


图 23 不同采样率下的运行时间

(2)不同压缩比下的 PSNR 比较

表 2 给出了各个算法分别在相同的目的信号不同的采样率下的重构峰值信噪比，单位为分贝(dB)，采样率为 0.1～0.8。

在图 24 中，以采样率为横坐标，峰值信噪比为纵坐标，绘制出各个算法峰

值信噪比随采样率变化的二维曲线。从图中可以看出，BP 算法在重构的精度方面要比其他算法高，而 SP 算法和 CoSaMP 算法在重构精度方面差别不大。

表 2 峰值信噪比 PSNR(dB)

算法 / 采样率	BP	OMP	SP	CoSaMP
0.1	17.101576	16.8750618	15.4537245	13.9159704
0.2	19.563315	18.9563759	18.4944196	16.7670693
0.3	22.2229559	21.2317177	20.2307241	19.0417974
0.4	24.4868604	23.2738142	22.5569135	21.6834668
0.5	26.5345038	25.0536905	24.5991328	24.1985496
0.6	28.3480559	26.5566198	26.951376	26.4117078
0.7	30.5339896	28.0013884	28.5878874	28.5262477
0.8	33.5151931	29.2402972	30.0386322	30.6389808

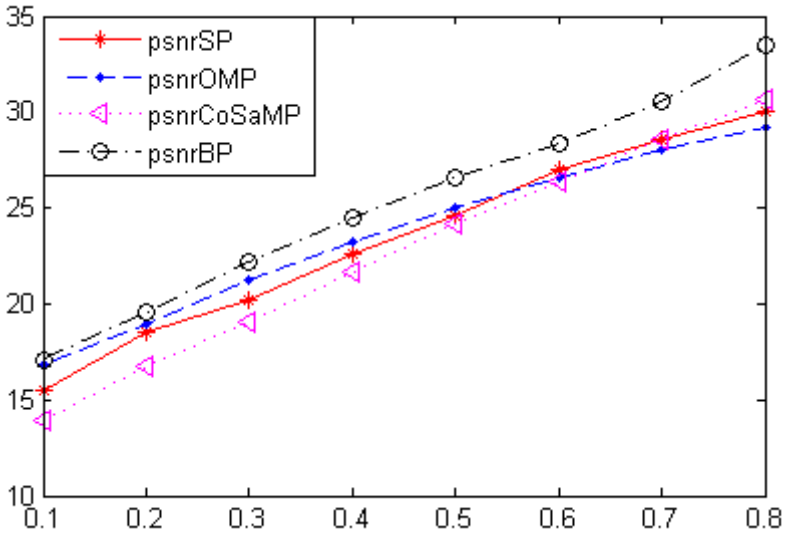


图 24 不同压缩比下的 PSNR

(3)重构误差比较

在比较重构误差时，主要比较均方误差、相对误差和绝对误差。

① 均方误差：$\mathrm{MSE} = (\| x - \hat{x} \|_2^2)/(M \times N)$;

② 相对误差：$\mathrm{ERR} = (\| x - \hat{x} \|_2)/(\| \hat{x} \|_2)$;

③ 绝对误差：$\mathrm{ERR} = (\| x - \hat{x} \|_2)$;

结果如表 3 所示。从表 3 可知，BP 算法在重构精度方面比其他算法要高，OMP 算法的重构误差较大，而 SP 算法和 CoSaMP 算法在重构误差方面差别较小。

表 3 算法重构误差

算法 / 误差	BP	OMP	SP	CoSaMP
均方误差	30.5948	75.0833	60.206	54.841
相对误差	0.0392	0.0616	0.0553	0.0527
绝对误差	1.42E+03	2.22E+03	1.99E+03	1.90E+03

下面，通过这些实验所得数据，分析各个算法的性能，根据各算法表现出来的特性总结其在实际运用中的优点和缺点。：

(1)BP 算法基于求最小l_1范数问题，通过迭代的方式从采样矩阵里寻找信号的最稀疏表示，用最少的基表示稀疏信号 x。优点在于重构的时候所需的测量个数少，重构的精度最高，缺点是运算量较大，执行效率较低。

(2)MP 算法在对信号进行重构的时候，将感知矩阵视为字典，感知矩阵中的每一列元素视为原子信号，然后对这些信号进行选择组合，从而完成对信号 y 的重构。MP 算法相对 BP 算法而言，时间复杂度低，但重构的精度不高。

(3)OMP 算法通过格拉姆-施密特正交化的方式生成一个正交的列集合，减少了循环次数。但 OMP 算法所需要的测量数更多，并且每次只用一个原子信号对原子集合进行更新，带来了重建时间代价。

(4)SP 算法在每次的迭代中找到比上一次更好的子空间，完成对原始信号的重建。SP 算提高了收敛速度。但因为每次迭代都必须对 K 个向量的子集进行处理，因此 SP 算法在时间复杂度上比 OMP 算法要高。重构的误差相对较低。

(5)CoSaMP 算法不是选择一个原子信号，也不会维护 K 个向量的集合，而是保证了每次选择集合原子个数小于等于 3K 个，迭代索引集合原子个数为 2K 个，每次剔除原子个数小于等于 K 个，进一步提高收敛速度，CoSaMP 算法的重构精度跟 SP 算法相差较小，但低于 BP 算法和 OMP 算法。

第四章 总结与展望

本文主要介绍了压缩感知技术相关知识，探讨了目前为止国内外学者对压缩感知中信号的稀疏表示、采样矩阵、重构算法等主要方向所做的研究，并阐述了这些方面的基本原理。

本文的核心研究内容和关键点在于对现有的一些压缩感知重构算法进行原理分析和仿真，列举了常用的 BP 算法、MP 算法发、OMP 算法、SP 算法和 CoSaMP 算法，对这些算法进行了详细的理论分析，给出了每个算法的运行原理和执行步骤，制作了算法流程图，并利用一维和二维原始信号对重构算法进行仿真实验，得到各个算法的运行时间、峰值信噪比、重构误差等实验数据。

通过这些实验所得数据，分析比较了各个算法的性能，包括重构时的运行时间、信号的峰值信噪比以及重构的误差，根据各算法表现出来的特性总结了在实际运用中的优点和缺点。

当然，由于作者能力水平有限，本文还有很多相关知识没能涉及到，比如在稀疏表示中有关字典学习的知识、采样矩阵改进方法的研究以及其他复杂的重构算法等，本人将会在以后的学习中不断的完善。

参考文献

[1]Candes EJ，Tao T. Near-optimal signal recovery from random projections: Universal encoding strategies[J].IEEE Transactions on Information Theory，2006，52(12)：5406-6425.

[2]Prony R. Essai experimental[J]. de l’Ecole Polytechnique(Paris)，1795，1(2)：24-76.

[3]Donoho DL. Compressed sensing[J]. IEEE Transactions on Information Theory. 2006，52(4)：1289-1306

[4]Baraniuk R. Compressed sensing[J]. IEEE signal processing magazine，2007,24(4).

[5]Candes EJ. Compressed sampling[C]. Proceedings oh the International Congress of Mathematicians：Madrid，August 22-30，invited lectures，2006，pp. 1433-1452.

[6]李峰，郭毅. 压缩感知浅析[M]. 北京：科学出版社，2015.

[7]闫敬文，刘蕾，屈小波. 压缩感知及应用[M]. 北京：国防工业出版社，2015.

[8]高西全，丁玉美. 数字信号处理[M].第三版. 西安：西安电子科技大学出版社，2008.

[9]焦李成，杨淑媛，刘芳，侯彪. 压缩感知回顾与展望[N]. 电子学报，2011，39(7)：1651-1662.

[10]石光明，刘丹华，高大华，刘哲，林杰王，良君. 压缩感知理论及其研究进展[N]. 电子学报，2009，37(5)：1070-1081.

[11]戴琼海，付长军，季向阳. 压缩感知研究[N]. 计算机学报，2011,34(3)：425-434.

[12]谢晓春,张云华. 基于压缩感知的二维雷达成像算法[N]. 电子与信息学报,2010,32(5)：1234-1238.

[13]江海，林月冠，张冰尘，洪文. 基于压缩感知的随机噪声成像雷达[N]. 电子与信息学报，2011,33(3)：672-676.

[14]方红，杨海蓉. 贪婪算法与压缩感知理论[N]. 自动化学报，2011,37(12)：1413-1421.

[15]朱明,高文,郭立强. 压缩感知理论在图像处理领域的应用[N]. 中国光学,2011,4(05)：441-447.

[16]周灿梅. 基于压缩感知的信号重建算法研究[D]. 北京交通大学，2010.

[17]李维明. 基于矩阵分解的压缩感知重构算法的研究[D]. 安徽大学，2013.

[18]王智慧. 基于压缩感知的图像重构算法研究[D]. 西安电子科技大学，2012.

[19]曹离然. 面向压缩感知的稀疏信号重构算法研究[D]. 哈尔滨工业大学，2011.

[20]庞茜. 压缩感知恢复算法研究[D]. 天津大学，2011.

[21]屈冉. 压缩感知算法及其应用研究[D]. 南京邮电大学，2013.

[22]马春晖. 压缩感知重构算法研究[D]. 杭州电子科技大学，2011.

致谢

值此论文完成之时，向本人的指导老师任新敏老师表示深深的感谢和敬意。任老师从本课题的制定到实施以及论文的编写都提供了很大的帮助，在任老师的指导下完成了对课题相关内容的研究，掌握了课题内容的基本原理，顺利完成了论文的撰写。同时也对在仿真实验过程中提供帮助的同学表示感谢，感谢他们与我探讨软件程序相关问题。

附录

程序 2-1

```
function ONE_DIM_SAMP()
%一维离散信号 DCT 变换
    k = 1:100;
    N = 100;
    y = sin(4*k*pi/N);
    y1 = dct2(y);
    figure;
    stem(y,'.');
    title('y = sin(4*k*pi/N),k=1:100,N=100');
    figure;
    stem(y1,'.');
    title('DCT( y )');
end
```

程序 2-2

```
function DCT()
%二维离散信号 DCT 变换
    img=imread('lena.bmp');
    img_data=dct2(img);
    figure;
    imshow(img);
    title('original');
    xlabel('(a)');
    img_data(abs(img_data)<10)=0;
    img_rec=idct2(img_data)./255;
    figure;
    imshow(img_rec) ;
    title('recovery');
    xlabel('(b)');
end
```

程序 3-1

```
function Gauss_Mat = Get_Gauss_Mat( M, N )
%GET_ GAUSS_MAT  生成大小为 M×N 的随机高斯矩阵
Gauss_Mat=randn(M,N);
Gauss_Mat = Gauss_Mat./repmat(sqrt(sum(Gauss_Mat.^2,1)),[M,1]);
end
```

程序 3-2

```
function Dct_Mat = Get_Dct_Basis()
    %获取 DCT 变换基矩阵
```

```
    Dct_Mat = zeros(256,256);
    for k=0:255
        tmp=cos([0:255]'*k*pi/256);
        if k>0
            tmp=tmp-mean(tmp);
        end;
        Dct_Mat(:,k+1)=tmp/norm(tmp);
    end
end
```

程序 3-3

```
Function [rec_x0] = bp(x,y,Phi,N)
    %bp 算法
    Phi2 = [Phi,-Phi];
    b = y;
    for i =1 : N
        u(i,1) = max(0,x(i));
        v(i,1) = max(0,-x(i));
    end
    x = [ u ; v];
    c = [ ones(1,length(u)) ones(1,length(v)) ];
    x = linprog(c',[],[],Phi2,b,zeros(size(x)),[]);
    rec_x0 = x(1:length(u)) - x(length(u)+1:length(u)+length(v));
end
```

程序 3-4

```
function [x_rec] = mp()
    %mp 算法
    x_rec = zeros(N,1);
    times =    K;
    g = zeros(N,1);
    r    = y;
    for n=0:times
        g = Phi' * r;
        [val,K] = max( abs(g) ) ;
        x_rec(K,1) = x_rec(K,1) + g(K,1);
        r = r - g(K,1) * Phi(:,K);
    end
end
```

程序 3-5

```
function X=omp(Y,Compressed_Mat,m)
    %omp 算法
    X=zeros(1,m);
```

```
    A=[];
    r_n=Y;
    N=length(Y);
    s=floor(n/4);
    for t=1:s;
        product=abs(Compressed_Mat'*r_n);
        [v,p]=max(product);
        A=[A,Compressed_Mat(:,p)];
        Compressed_Mat(:,p)=zeros(n,1);
        a=(A'*A)^(-1)*A'*Y;
        r_n=Y-A*a;
        p_a(t)=p;
    end
    X(p_a)=a;
end
```

程序 3-6

```
function X=sp(Y,Compressed_Mat,m)
%sp 算法
r_n=Y;
s_p_l=[];
N=length(Y);
s=floor(N/4);
for t=1:s
    pd=abs(Compressed_Mat'*r_n);
    [v,p]=sort(pd,'descend');
    s_p_c=p(1:s);
    s_p=union(s_p_c,s_p_l);
    A_t=Compressed_Mat(:,s_p);
    a_x_c=zeros(m,1);
    a_x_c(s_p)=(A_t'*A_t)^(-1)*A_t'*Y;
    [v,p]=sort(abs(a_x_c),'descend');
    X=zeros(1,m);
    X(p(1:s))=a_x_c(p(1:s));
    s_p_l=p(1:s);
    r_n=Y-Compressed_Mat*X';
end
```