Hyperspectral Image Denoising and SVD Acceleration

Weitong Liang

1 Brief Background on the Denoising Problem

Image denoising is one of the most basic inverse problems in image processing. Understanding the sources of image noise and its complex, often exaggerated effects is helpful in performing image denoising. If we can remove the noise that conforms to certain statistical norms, we're already doing quite well. In addition, deep learning has been widely used in image denoising, and the distributional characteristics of noise are crucial for producing high-quality training datasets.

We can abstract the denoising problem as solving $D = L + M$, where D is the noisy data, L is the denoised data, and M is the noise. Although this model is simple, in most real cases it is not easy to solve. The current mainstream image denoising methods can be roughly divided into three categories: Filtering-Based Methods, Model-Based Methods, and Learning-Based Methods.

There's another issue worth noting. For image noise, it is not just about spatial anomalies — a pixel that appears isolated from its neighborhood may not necessarily be noisy. A better definition of noise is "randomness," which is better illustrated over time. If repeated images show large variation at a pixel, we call it noisy. Hence, we often compute the variance in a local region to assess noise, treating the pixels in that region as multiple observations at the same point.

2 Denoising Models

2.1 Principal Component Analysis and Singular Value Decomposition

2.1.1 Basic Idea of PCA

PCA is a widely used tool in data analysis. Its main idea is to map an n-dimensional feature vector to a k-dimensional space, where the new features are orthogonal and known as principal components. These components are derived from the original variables and reconstructed based on them.

In dimensionality reduction, PCA seeks a decomposition that maximizes data "spread" and "uncorrelatedness." "Spread" can be measured by variance, and "uncorrelatedness" by

having near-zero covariances. The covariance matrix captures both, and thus we can use its eigen-decomposition to construct new features.

Let the original data matrix be X, then the covariance matrix is $XX^\top$, with diagonal elements representing variances and off-diagonals representing covariances. PCA proceeds by computing the eigenvalues and eigenvectors of this matrix, selecting the top k components that retain the most variance.

2.1.2 Using SVD for PCA

In PCA, we often encounter the need to decompose the covariance matrix, and when the dimension is large, eigenvalue computation becomes difficult. In practice, we usually use SVD to accomplish the decomposition.

Let $A \in \mathbb{R}^{m\times n}$, then the SVD of A is given by:

$$A = U\Sigma V^\top$$

where U and V are orthogonal matrices, and Σ contains the singular values. Then,

$$AA^\top = U\Sigma^2 U^\top$$

and so PCA projection directions can be obtained from columns of U corresponding to the largest singular values.

2.1.3 SVD via QR Factorization

Given $A \in \mathbb{R}^{m\times n}$, we compute QR decomposition $A = QR$. Since SVD of R is cheaper, we proceed by computing $R = Q^\top A$, then $\text{SVD}(R) = U_2\Sigma V^\top$, so

$$A = QR = QU_2\Sigma V^\top$$

This simplifies computation, although additional QR steps may increase complexity.

2.2 Variants of RPCA

2.2.1 Limitations of PCA and the Idea of RPCA

PCA performs low-rank approximation under an ℓ_2 objective. However, outliers can distort the solution. RPCA models data as $M = L + S$, where L is low-rank and S is sparse. The optimization is:

$$\begin{aligned} \min_{L,S} \quad & \|L\|_* + \lambda\|S\|_1 \\ \text{s.t.} \quad & M = L + S \end{aligned}$$

2.2.2 CTV-RPCA

In RPCA, we only use a low-rank prior. But images often have local smoothness, i.e., pixel values change smoothly in small regions. We introduce Total Variation (TV) penalties in CTV-RPCA to enforce local smoothness, especially in 3D image tensors. The resulting formulation:

$$\begin{aligned} \min_{X,S} \quad & \sum_{i=1}^{3} \|G_i X\|_1 + \lambda \|S\|_1 \\ \text{s.t.} \quad & M = X + S, \quad G_i = \nabla_i X \end{aligned}$$

2.2.3 WNNM-RPCA

WNNM introduces weighted nuclear norm minimization. Instead of shrinking all singular values equally, WNNM assigns different weights, allowing large singular values (signal) to shrink less and small ones (noise) to shrink more.

Let the weight be $\omega_i = \frac{c}{\sigma_i(X)+\epsilon}$. The final optimization is:

$$X = \arg\min_X \|Y - X\|_F^2 + \sum_i \omega_i \sigma_i(X)$$

This formulation improves denoising quality.

3 Optimization Algorithms

3.1 EALM

For convex problems with equality constraints, the Augmented Lagrangian Method (ALM) adds quadratic penalties. The general form:

$$\begin{aligned} \min_x \quad & f(x) + \lambda \|Ax - b\|_2^2 \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

This results in a saddle-point problem. For RPCA, the augmented Lagrangian is:

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda \|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2$$

3.2 ADMM

ADMM is suitable for problems separable in two variables with linear constraints. The standard form is:

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c \end{aligned}$$

The iteration steps are:

$$x^{k+1} = \arg\min_x \mathcal{L}_\rho(x, z^k, u^k)$$
$$z^{k+1} = \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, u^k)$$
$$u^{k+1} = u^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$$

ADMM leads to efficient solutions using SVD and soft-thresholding.

4 Experiments and Results Analysis

4.1 4.1 Validation of Low-Rank and Local Smoothness Priors

We first validate whether the selected image datasets possess low-rank and local smoothness properties. For a dataset $D \in \mathbb{R}^{m\times n\times p}$, we unfold it into a matrix of size $mn \times p$. To verify low-rankness, we perform SVD on the unfolded matrix $\text{unfold}(D) = U\Sigma V^\top$, and examine the distribution of singular values.

For example, the dataset `OriH` exhibits a rapidly decaying singular value curve, concentrated in the first few singular values. Similar results are observed for `oriData3` and `simuindian`.

To verify local smoothness, we compute finite differences along image rows and columns. The resulting difference images reveal the object contours, consistent with the original image structure. Similar results are observed for `oriData3` and `simuindian`.

The corresponding visual results are provided in the appendix.

4.2 4.2 Denoising Experiments with Gaussian and Sparse Noise

We first tune parameters for WNNM and LRTV using `simuindian` as an example. Under Gaussian noise, the optimal parameter C for WNNM is 0.03, and for LRTV the best combination is $\tau = 0.008, \lambda = 0.06, r = 9$.

Next, we compare four methods (CTV, RPCA, WNNM, LRTV) under Gaussian noise levels $G = 0.1, 0.2, 0.3, 0.4$. Below is the evaluation for `simuindian`:

4.3 4.3 Speedup Experiments with rSVD

In randomized SVD (rSVD), the column dimension k of the sketch matrix significantly affects speed and accuracy. We set $k_i = \text{round}(\min(\text{size}(A))/5^i)$ and compare the SVD time under different k.

To further improve efficiency, we propose three accelerated rSVD algorithms: `ieeesvdr`, `method1r`, and `method2r`. Each is evaluated for $r = 8, 16, 24$ on the `OriH` dataset.

4.4 4.4 Denoising Experiments with Embedded Accelerated rSVD

Although randomized SVD algorithms can significantly reduce the cost of individual SVD computations, they may slow down convergence due to discarded smaller singular values. To balance runtime and image quality, we experiment with different sketch sizes R.

Table 1: Evaluation on `simuindian` with Gaussian Noise

G	Method	PSNR	SSIM	ERGAS	Time
0.1	RPCA	23.7734	0.6718	9.6092	4.1686
	CTV	50.8164	0.9994	0.4399	47.7877
	WNNM	23.5351	0.6631	9.8605	4.6269
	LRTV	29.2611	0.9265	5.1585	91.7575
0.2	RPCA	21.6823	0.5786	12.2029	3.7615
	CTV	50.8164	0.9994	0.4399	43.2092
	WNNM	22.3984	0.5620	11.2193	4.4203
	LRTV	26.9107	0.8867	6.7417	94.0694
0.3	RPCA	20.5133	0.5217	13.9619	3.7630
	CTV	50.8164	0.9994	0.4399	43.7176
	WNNM	21.5236	0.5012	12.3948	4.4095
	LRTV	25.2754	0.8512	8.1376	94.6543
0.4	RPCA	19.5999	0.4799	15.4780	3.9450
	CTV	50.8164	0.9994	0.4399	45.6125
	WNNM	20.7769	0.4565	13.4946	4.4466
	LRTV	24.0622	0.8172	9.3660	95.7970

Table 2: Time Cost of rSVD with Different k

Method	1	2	3	4	5	6	7	8	9	10
SVD	0.1513	0.1482	0.1460	0.1518	0.1512	0.1547	0.1565	0.1556	0.1466	0.1504
rSVD(k1)	0.0918	0.0964	0.0929	0.0934	0.0953	0.1034	0.1029	0.0989	0.0951	0.0947
rSVD(k2)	0.0549	0.0538	0.0554	0.0543	0.0544	0.0526	0.0525	0.0534	0.0538	0.0566
rSVD(k3)	0.0444	0.0443	0.0449	0.0438	0.0440	0.0473	0.0468	0.0448	0.0440	0.0438
rSVD(k4)	0.0404	0.0400	0.0425	0.0408	0.0381	0.0387	0.0390	0.0377	0.0373	0.0412

Table 3: Time Cost of Different rSVD Methods

Method	1	2	3	4	5	6	7	8	9	10
ieeer1	0.0156	0.0153	0.0158	0.0162	0.0159	0.0159	0.0157	0.0157	0.0157	0.0159
method1r1	0.0097	0.0103	0.0101	0.0103	0.0104	0.0096	0.0102	0.0093	0.0101	0.0106
method2r1	0.0093	0.0101	0.0102	0.0097	0.0106	0.0101	0.0100	0.0093	0.0098	0.0096
ieeer2	0.0200	0.0192	0.0194	0.0202	0.0195	0.0190	0.0197	0.0191	0.0194	0.0210
method1r2	0.0136	0.0134	0.0134	0.0140	0.0141	0.0140	0.0155	0.0149	0.0131	0.0141
method2r2	0.0140	0.0141	0.0150	0.0154	0.0138	0.0135	0.0153	0.0140	0.0141	0.0139
ieeer3	0.0296	0.0298	0.0334	0.0311	0.0302	0.0314	0.0318	0.0319	0.0308	0.0318
method1r3	0.0197	0.0193	0.0242	0.0207	0.0206	0.0219	0.0201	0.0206	0.0198	0.0211
method2r3	0.0191	0.0201	0.0245	0.0198	0.0202	0.0196	0.0201	0.0200	0.0198	0.0195

Table 4: Evaluation of `simuindian` with Salt and Pepper Noise (S = 0.1)

Method	PSNR	SSIM	ERGAS	Time
RPCA	43.1355	0.9975	1.0243	4.0816
CTV	50.8164	0.9994	0.4399	49.1001
WNNM	38.9460	0.9924	1.6783	5.2465
LRTV	41.4424	0.9949	1.2816	84.8005

Adaptive R-scheduling Results

To further improve efficiency, we apply stage-wise adjustment of sketch size R according to the precision of current iterations. Smaller R is used in early iterations, and larger R as the algorithm converges. Precision cutoffs are set as $C_1 = 10^{-4}, C_2 = 10^{-5}, C_3 = 10^{-6}$.

Table 5: Stage-wise rSVD Results on `simuindian` (Salt and Pepper Noise)

Method	ieee			method1			method2		
Cutoff	C1	C2	C3	C1	C2	C3	C1	C2	C3
PSNR	40.78	39.17	38.57	42.41	40.83	40.77	42.40	41.36	40.29
SSIM	0.993	0.990	0.988	0.996	0.993	0.993	0.996	0.994	0.992
ERGAS	1.42	1.69	1.82	1.19	1.41	1.40	1.17	1.32	1.49
Time	48.12	44.50	43.02	51.51	48.72	42.87	45.26	43.97	41.41