# Hyperspectral Images Denoising via a Tensor Dictionary
# EECS 556 Project Report

Mingshuang Zhang, Jiachen Jiang, Chengtian Zhang, Jason Hu

July 5, 2022

Honor code: Our team will give attribution for any figures used in our documents and will cite all code sources beyond those already built into the Julia Base or Matlab toolboxes.

**Abstract**

As 3rd order tensors, Hyperspectral images (HSIs) can deliver more authentic representations for real scenes, enhancing the performance of many computer vision tasks when compared with traditional RGB or gray-scale images. In practice, the quality of an HSI is always affected by various noises. In this paper we propose an effective HSI denoising approach based on the Tensor Dictionary Model by considering two intrinsic characteristics underlying an HSI: the nonlocal similarity over space and the global correlation across spectrum. We modify the original method of grouping similar patches into clusters and optimizing over each cluster by designing an optimization problem with an objective function targeted at representing the HSI with a sparse global dictionary representation. Iteration over variables is used to solve the nonconvex Lagrange dual problem. Experimental results show that our method outperforms some basic state-of-the-art HSI denoising methods under comprehensive quantitative performance measures due to convergence issues. At the end we analyze our results and explore other possibly better methods.

## 1 Introduction

Hyperspectral images (HSIs) consist of measurements made from across the electromagnetic spectrum, with two spatial dimensions and one spectral dimension for each pixel. HSIs can deliver more faithful representation for real scenes with its abundant spectral information and are used in various fields, including land use analysis, environment monitoring, and field surveillance [1].

In real cases, however, an HSI is always corrupted by some noise, generally from equipment limitations such as sensor sensitivity, photon effects and calibration error. Furthermore, since the radiance energy is limited and bandwidth can be fairly narrow, the energy captured by each sensor might be low, inevitably leading to shot noise and thermal noise. The denoising problem for HSI is thus still of acute and growing importance [2].

To solve this problem, the most common approach is to break the spatial dimensions, usually consisting of several hundred pixels, into much smaller patches, while keeping all the spectral dimensions together,

preserving the low rank properties of the spectrum. This also eliminates the need to process matrices with large numbers of rows, algorithms whose complexity generally scale with high powers of the matrix size, instead offering a divide and conquer approach to each patch. To avoid boundary mismatches between patches, generally, some overlap between the patches is chosen. Patterns across different patches can be exploited for higher assurances of quality.

One recent and interesting paper that addresses the problem of removing Gaussian noise is [1]. This paper tackles the issue of three dimensional HSIs head on by representing patches as third order tensors. Furthermore, the paper exploits the fact that images generally contain similar patterns in different locations, and creates clusters of patches that can be represented sparsely by a dictionary. The dictionary can be trained with the noisy image and contains more columns than rows, allowing for this sparse representation. This method can readily be adapted to work when reshaping the spatial dimensions into one vector as well.

For our innovation approach, we use a global dictionary on every patch, instead of a smaller dictionary on each cluster. This should lead to less overfitting to the noisy image, as well as resolve the issue of choosing an appropriate parameter for clustering. Furthermore, we choose to use a convex norm to measure the sparsity of the dictionary representation, rather than impose a hard constraint on the zeros of the representation. This allows for more flexibility of representations and expands the tools that can be used to solve the optimization problem.

## 2 Quantitative Performance Prediction

We will assess the quantitative performance of our method as follows.

- We can first add noise to clean images, implement our developed dictionary into hyperspectral image denoising, and compare the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) with the original dictionary-based methods as well as our new method on the CAVE dataset.

  The PSNR is most easily defined via the mean squared error (MSE). Given a noise-free $m \times n$ monochrome image I and its noisy approximation K, MSE is defined as:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j)] - K(i,j)]^2. \tag{1}$$

  The PSNR (in dB) is defined as:

$$\begin{aligned} \text{PSNR} &= 10 \times \log_{10} \left( \frac{\text{MAX}_I^2}{\text{MSE}} \right) \\ &= 10 \times \log_{10}(\text{MAX}_I) - 10 \times \log_{10}(\text{MSE}). \end{aligned} \tag{2}$$

  Here, $\text{MAX}_I$ is the maximum possible pixel value of the image.

  The SSIM formula is based on three comparison measurements between the samples of $x$ and $y$:

luminance ($l$), contrast ($c$) and structure($s$). The individual comparison functions are:

$$\begin{aligned} l(x,y) &= \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \\ c(x,y) &= \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \\ s(x,y) &= \frac{2\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}, \\ c_3 &= c_2/2, \end{aligned} \tag{3}$$

where $\mu_x$, $\mu_y$ is the average of $x$ and $y$; $\sigma_x$, $\sigma_y$ is the variance of $x$ and $y$; $\sigma_{xy}$ is covariance of $x$ and $y$; $c_1$, $c_2$, $c_3$ are variables to stabilize the division with weak denominator. SSIM is then a weighted combination of those comparative measures:

$$\text{SSIM} = [l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma], \tag{4}$$

where $\alpha, \beta, \gamma$ are weights.

After setting the weights $\alpha, \beta, \gamma$ to 1, the formula can be reduced to:

$$\text{SSIM} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{5}$$

Since the method applied in paper [3] is similar to the innovation method we derived, we make a prediction of the PSNR and SSIM results based on the results in the paper [3]. In particular, we plan to get 5dB higher PSNR and a value of SSIM which is the average of the K-SVD algorithm [4] and ideal denoising (SSIM of 1) on the same dataset.

For example, suppose after adding Gaussian white noise with variance of $\sigma^2 = 0.01$ to a HSI, we get PSNR $= 20.00$ and SSIM $= 0.15$. If the normal K-SVD denoising algorithm [4] would produce a image with PSNR $= 29.97$ and SSIM $= 0.62$, then our target PSNR is $20.00 \times 10^{\frac{5}{20}} = 35.57$ and our target SSIM is $(1 + 0.62) \times 0.5 = 0.81$.

- More specifically, we took one clean HSI and corrupted it with three levels of Gaussian noise: $\sigma = 0.1$, $\sigma = 0.2$, and $\sigma = 0.3$. For each HSI, we plan on using the tensor learning dictionary method, our refined algorithm, and a few other state of the art denoising methods to denoise each of these noisy versions of the HSI. Our method is expected to achieve at least 35dB PSNR so that the image quality will be acceptable, as well as at least 0.9 SSIM.

# 3 Plan

## 3.1 Summary of the methods to be investigated

We have one primary paper [1] whose HSI denoising methods, Decomposable Nonlocal Tensor Dictionary Learning, is what we will investigate.

### 3.1.1 HSI Noise model

Consider a noisy HSI $z(\boldsymbol{x}) \in \mathbb{R}^{W \times H \times S}$, where $W$ represents the width of the HSI data, $H$ is the height of the image, and $S$ is the number of spectral bands. The HSI can be written as:

$$z(\boldsymbol{x}) = y(\boldsymbol{x}) + n(\boldsymbol{x}), \tag{6}$$

where $y$ is the latent image and $n$ is zero-mean Gaussian noise with variance $\sigma^2$.

### 3.1.2 Traditional Dictionary Learning (DL)

In the traditional Dictionary Learning (DL) model, for a set of image patches (ordered lexicographically as column vectors) $\{\boldsymbol{T}_i\}_{i=1}^n \subset \mathbb{R}^d$, where $d$ is the dimensionality and $n$ is the number of image patches, DL aims to calculate the dictionary $\boldsymbol{D} = [d_1, \ldots, d_m] \in \mathbb{R}^{d \times m}$, composed by a collection of atoms $d_i$ and the coefficient matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n] \in \mathbb{R}^{m \times n}$, composed by the representation coefficients $\boldsymbol{z}_i$, by the optimization model

$$\min_{\boldsymbol{D}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_n} \sum_{i=1}^{n} \|\boldsymbol{T}_i - \boldsymbol{D}\boldsymbol{z}_i\|_2, \ s.t. \mathcal{P}(\boldsymbol{z}_i) \leqslant k, \tag{7}$$

where $\mathcal{P}(\ )$ denotes a sparsity controlling operator such as the $l_0$ norm.

By learning from the training data, usually the corrupted HSI itself, we can find the basic atoms to form a dictionary. Using the learned dictionary, the clean image patches can be represented with sparse coefficients.

The special properties of HSI need to be considered. There are two main characteristics of HSIs: nonlocal similarity in space, and global correlation in spectrum. Nonlocal similarity means there are many similar patches in an image that can be clustered together for processing. Global correlation means that images in different bands are always highly correlated. The tensor dictionary learning method keeps the high dimensional structure and performs better than dimensional reduction methods in HSI denoising.

### 3.1.3 Decomposable Nonlocal Tensor Dictionary Learning

Considering a HSI $z(\boldsymbol{x}) \in \mathbb{R}^{W \times H \times S}$, the steps of HSI denoising method are as follows.

Firstly, construct HSI patches with spatial width $d_w$, height $d_h$, and spectral bands of size $B$ to get a $3^{rd}$ order tensor $\mathbf{T} \in \mathbb{R}^{d_w \times d_h \times S}$. By sweeping across the whole image with overlaps, we get 3D full-band patches (FBP) and reformulate all the FBPs as a group of 3D patches $\{\mathbf{T}_i\}_{i=1}^n$, where $n = (W - d_w + 1)(H - d_h + 1)$ denotes the total number of patches. Each FBP contains local spatial patterns while preserving global spectral dimensionality.

Secondly, due to non-local self similarity, we adapt the k-means++ [5] to group the similar FBPs as one cluster and form a total of $K$ clusters. The k-means++ [5] method is a clustering technique that seeks to minimize the average squared distance between points (patches) in the same cluster. Specifically, this method works as follows:

---

Algorithm: k-means++ method

**INPUT**: A group of 3D overlapped patches $\{\mathbf{T}_i\}_{i=1}^n$, where $n = (W - d_w + 1)(H - d_h + 1)$ denotes the total number of patches, and the total numbers of clusters $K$.
**OUTPUT**: An $n$-element output column vector that contains the cluster label of each patch.

1. Choose an initial center $\mathbf{c}_1$ uniformly at random from $\{\mathbf{T}_i\}_{i=1}^n$ .
2. Choose the next center $\mathbf{c}_i$, selecting $\mathbf{c}_i = \mathbf{T}' \in \{\mathbf{T}_i\}_{i=1}^n$ with probability $\frac{\text{Dis}(\mathbf{T}')^2}{\sum_{\mathbf{T}\in\{\mathbf{T}_i\}_{i=1}^n}\text{Dis}(\mathbf{T})^2}$. $\text{Dis}(\mathbf{T}) = \min \|\mathbf{T} - \mathbf{c}_{\text{chosen}}\|_F$ denotes the shortest distance from the data point to the closest center we have already chosen.
3. Repeat Step 2 until we have chosen $K$ initial centers $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$.
4. For each $i \in \{1, \ldots, K\}$, set the cluster $\mathbf{C}_i$ to be the set of points in $\{\mathbf{T}_i\}_{i=1}^n$ that are closer to $\mathbf{c}_i$ than they are to $\mathbf{c}_j$ for all $j$.
5. For each $i \in \{1, \ldots, K\}$, set $\mathbf{c}_i$ to be the center of mass of all points in $\mathbf{C}_i$: $\mathbf{c}_i = \frac{1}{|\mathbf{C}_i|}\sum_{\mathbf{T}\in\mathbf{C}_i}\mathbf{T}$.
6. Repeat Steps 4 and 5 until $\mathcal{C}$ no longer changes.

After this process, we get $K$ groups of overlapped patches. The $k$-th group can be written as a tensor $\mathcal{X}^{(k)} \in \mathbb{R}^{d_w\times d_h\times S\times n^k}$, where $n^k$ is the number of patches in the $k$-th group.

Thirdly, for each group $k$, we learn the corresponding spatial dictionaries $\boldsymbol{D}_W^k \in \mathbb{R}^{d_w\times m_w}$ and $\boldsymbol{D}_H^k \in \mathbb{R}^{d_h\times m_h}$, spectral dictionary $\boldsymbol{D}_{\text{spectral}}^k \in \mathbb{R}^{S\times m_{\text{spectral}}}$, and sparse coefficient tensor $\mathcal{Z}^k \in \mathbb{R}^{m_w\times m_h\times m_{\text{spectral}}\times n^k}$ for each patch in the group, where $m_w, m_h$ and $m_{\text{spectral}}$ are the number of atoms in the width, height, and spectral dictionary respectively.

Fourthly, using the sparse coefficients and the dictionaries, we reconstruct each patch and average them to form the target HSI image.

We can construct the following nonlocal HSI DL model:

$$\text{argmin}_{\boldsymbol{D}_W^k,\boldsymbol{D}_H^k,\boldsymbol{D}_{\text{spectral}}^k,\mathcal{Y}^k}\sum_{k=1}^{K}\|\mathcal{X}^{(k)} - \mathcal{Y}^k \times_1 \boldsymbol{D}_W^k \times_2 \boldsymbol{D}_H^k \times_3 \boldsymbol{D}_{\text{spectral}}^k\|_F, \tag{8}$$

where the $\times$ notation represents the product of a tensor and a matrix, and $\mathcal{Y}^k = \text{Sub}(\mathcal{Z}^{(k)})$ is the sub-tensor of $\mathcal{Z}^{(k)}$ obtained by keeping the index set of the nonzero entries. The mode-$n$ product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\ldots\times I_n\times\ldots I_N}$ by a matrix $\boldsymbol{B} \in \mathbb{R}^{J_n\times I_n}$, denoted by $\mathcal{A} \times_n \boldsymbol{B}$, is also an $N$-th order tensor $\boldsymbol{C} \in \mathbb{R}^{I_1\times\ldots\times I_n\times\ldots I_N}$ , whose entries are computed by

$$c_{i_1\ldots i_{n-1}j_n i_{n+1}\ldots i_N} = \sum_{i_n} a_{i_1\ldots i_N} b_{j_n i_n}. \tag{9}$$

To solve the optimization problem (8), we first note that we can approximate the $k$-th group tensor $\mathcal{X}^{(k)}$ by a low-rank tensor using the dimensionality redundancy in its 3rd spectral mode (global correlation across spectrum):

$$\mathcal{X}^{(k)} \approx \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3 \times_4 \boldsymbol{U}_4, \tag{10}$$

where $\boldsymbol{U}_1 \in \mathbb{R}^{d_w\times r_k^w}$, $\boldsymbol{U}_2 \in \mathbb{R}^{d_h\times r_k^h}$, $\boldsymbol{U}_3 \in \mathbb{R}^{S\times r_k^{\text{spectral}}}$ , $\boldsymbol{U}_4 \in \mathbb{R}^{n^k\times r_k^n}$ and $\mathcal{G} \in \mathbb{R}^{r_k^w\times r_k^h\times r_k^{\text{spectral}}\times r_k^n}$ is the core tensor [6].

The Tucker [6] decomposition technique, a higher order SVD, finds the tensors in (5) which solve

$$\text{argmin}_{\boldsymbol{U}_1,\boldsymbol{U}_2,\boldsymbol{U}_3,\boldsymbol{U}_4,\mathcal{G}} \|\mathcal{X}^{(k)} - \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3 \times_4 \boldsymbol{U}_4\|. \tag{11}$$

Upon obtaining these optimal $\mathcal{G}, \boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \boldsymbol{U}_4$, we solve (8) by letting $\boldsymbol{D}_W^k = \boldsymbol{U}_1$, $\boldsymbol{D}_H^k = \boldsymbol{U}_2$, $\boldsymbol{D}_{\text{spectral}}^k = \boldsymbol{U}_3$, and $\mathcal{Y}^k = \mathcal{G} \times_4 \boldsymbol{U}_4$.

Finally we adopt the well known AIC/MDL [7] method to estimate the rank parameter $r_k^w$, $r_k^h$, $r_k^{\text{spectral}}$, and $r_k^n$. The original method examines the model

$$\mathbf{x}(t) = \sum_{i=1}^{q} \mathbf{A}(\phi_i) s_i(t) + \mathbf{n}(t), \tag{12}$$

where $\mathbf{x}(t)$ is the output $P$ dimensional vector, $\mathbf{A}(\phi_i)$ is a $P$ dimensional vector parametrized by an unknown vector $\phi_i$, $s_i(t)$ are scalar signals assumed to be stationary Gaussian random processes with zero mean, and $\mathbf{n}(t)$ is a $P$ dimensional vector of Gaussian noise. Let $q$ denote the number of signals. We assume that $q < P$ and that $N$ observations of the output $\mathbf{x}(t_1)$, ..., $\mathbf{x}(t_N)$ are known. The problem is then to find the number of signals $q$.

To solve this problem, two criterion for quantitatively determining the goodness of fit of a statistical model are established. More precisely, given a set of $N$ observations, $\boldsymbol{X} = [\mathbf{x}(t_1) \dots \mathbf{x}(t_N)]$, and a parametrized family of probability distributions $f(\boldsymbol{X}|\Theta_p)$, we wish to select the model that best fits the data. The Akaike Information Criterion, defined by

$$\textbf{AIC}_p = -2 \log f(\boldsymbol{X}|\Theta_p') + 2p, \tag{13}$$

where $\Theta_p'$ is the maximum likelihood estimation of the parameter vector $\Theta_p$ and $p$ is the number of free parameters in $\Theta_p$, evaluates the goodness of fit of the model. Similarly, the MDL criterion is defined as

$$\textbf{MDL}_p = -\log f(\boldsymbol{X}|\Theta_p') + \frac{1}{2} p \log N. \tag{14}$$

In both cases, the goal is to find the model that minimizes the chosen criterion.

The idea of AIC/MDL [7] is then the following. Let the sample covariance matrix $\boldsymbol{R}$ be defined by

$$\boldsymbol{R} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}(t_i) \mathbf{x}(t_i)^T. \tag{15}$$

Let $l_1 > l_2 > \ldots > l_P$ be the eigenvalues of $\boldsymbol{R}$, and suppose the true number of signals is $p$, for $0 \leq p \leq P - 1$. The log likelihood of the model can then be derived [6] to be

$$L(\Theta_p') = \log \left( \frac{\prod_{i=p+1}^{P} l_i^{1/(P-p)}}{\frac{1}{P-p} \sum_{i=p+1}^{P} l_i} \right)^{(P-p)N}. \tag{16}$$

We choose the value of $p$, from the $P$ possible values, that minimizes either the AIC or MDL, computed from the log likelihood.

To apply this method to our problem, for each $1 \leq i \leq 4$, let $\boldsymbol{X}_{(i)}^{(k)}$ be the flattened 2D array of the $i$-th mode of $\mathcal{X}^{(k)}$. The $i$-th mode refers to a matrix representation of a tensor. The dimension specified in $i$ map

to the rows of the matrix, and the remaining dimensions (in ascending order) map to the columns. Each $\boldsymbol{X}_{(i)}^{(k)}$ then consists of column vectors, which are to be represented by linear combinations of the atoms of the dictionaries, with some noise. Hence, the AIC/MDL [7] method can be used to estimate the number of unique atoms needed for each $\boldsymbol{X}_{(i)}^{(k)}$, without prior knowledge of the dictionaries. For $i = 1, 2, 3, 4$, we let this number for $\boldsymbol{X}_{(i)}^{(k)}$ be $r_k^w$, $r_k^h$, $r_k^{\text{spectral}}$, and $r_k^n$ respectively.

### 3.2 Reproduce the paper

We will implement the algorithm in MATLAB as follows:

---

Algorithm: HSI denoising via Tensor Dictionary Learning

---

**INPUT**: Noisy HSI $\mathcal{X} \in \mathbb{R}^{W \times H \times S}$,
**OUTPUT**: Spatial Dictionaries $\boldsymbol{D}_W = [\boldsymbol{D}_W^1, ..., \boldsymbol{D}_W^K]$ and $\boldsymbol{D}_H = [\boldsymbol{D}_H^1, ..., \boldsymbol{D}_H^K]$, Spectral dictionary $\boldsymbol{D}_{\text{spectral}} = [\boldsymbol{D}_{\text{spectral}}^1, ..., \boldsymbol{D}_{\text{spectral}}^K]$, and coefficient tensors $\mathcal{Z}^k, k = 1, ..., K$,where $K$ is the group number

---

1. Construct the tensor patches and group them into $K$ clusters, and find $K$ by $k$-means++ [5].
2. Calculate the rank parameters $r_k^w$, $r_k^h$, $r_k^{\text{spectral}}$ and $r_k^n$ by implementing the AIC/MDL [7] algorithm.
3. Implement the Tucker [6] decomposition technique to obtain $\mathcal{G}$, $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, $\boldsymbol{U}_3$, and $\boldsymbol{U}_4$.
4. Set $\boldsymbol{D}_W^k = \boldsymbol{U}_1$, $\boldsymbol{D}_H^k = \boldsymbol{U}_2$, $\boldsymbol{D}_{\text{spectral}}^K = \boldsymbol{U}_3$ and $\mathcal{Z}^k = \mathcal{G} \times_4 \boldsymbol{U}_4$.

---

**Noise Models**: We add zero-mean additive white Gaussian noise(AWGN) with standard deviation of $\sigma \in [0.1, 0.3]$ to each spectral channel of the image.

**Parameter Choice**: We plan to set patch size $d_w = d_h = 8$ as in [1] and sweep across the image with $\text{stepsize}_w = \text{stepsize}_h = 1$. The total number of patches from an image is $(512 - 8 + 1) \times (512 - 8 + 1)$.

**Comparison Methods**: We plan to compare our method with the state-of-the-art methods including the K-[4], ANLM3D [8], BM3D [9] and BM4D [10]. Many of these methods have code that is openly available.

---

#### 3.2.1 More details

We get our code from: `http://gr.xjtu.edu.cn/c/document_library/get_file?folderId=1766524&name=DLFE-38410.zip`.

The code for some of the comparison methods and toolboxes can be found here:

1. fkmeans: `http://www.mathworks.com/matlabcentral/fileexchange/31274-fast-k-means`
2. tensor_toolbox: `http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.5.html`

3. poblano_toolbox: https://software.sandia.gov//trac/poblano/
4. n-Mode Tensor-Matrix Product: https://https://www.mathworks.com/matlabcentral/fileexchange/24268-n-mode-tensor-matrix-product
5. invansc: http://www.cs.tut.fi/~foi/invansc/
6. SSIM_index.m: https://ece.uwaterloo.ca/~z70wang/research/\protect\protect\unhbox\voidb@x\hbox{SSIM}/
7. FeatureSIM.m: http://www4.comp.polyu.edu.hk/~cslzhang/IQA/FSIM/FSIM.htm
8. ksvdbox: http://www.cs.technion.ac.il/~ronrubin/software.html
9. ompbox: http://www.cs.technion.ac.il/~ronrubin/software.html
10. naonlm3d: http://personales.upv.es/jmanjon/denoising/arnlm.html
11. BM3D: http://www.cs.tut.fi/~foi/GCF-BM3D/
12. BM4D: http://www.cs.tut.fi/~foi/GCF-BM3D/

#### 3.2.2 Even more details

**Datasets**: We are using the Columbia Multispectral Image Database(**CAVE**) to test the proposed method. This dataset contains 32 real-world scenes, with each HSI image having $512 \times 512$ spatial resolution and 31 spectral levels. More specifically, the range of wavelength is from 400nm to 700nm in 10nm steps.

### 3.3 Division of Responsibilities

#### 3.3.1 Tasks

The following are the main tasks that we have completed.

1. Select the HSI database and establish a convenient sharing method.
2. Apply noise to the HSI images as described in section 2.
3. Calculate the number of the clusters $k$ by adapting the k-means++ [5] method.
4. Calculate the rank parameters $r_k^w$, $r_k^h$, $r_k^{\text{spectral}}$, $r_k^n$ by adapting the AIC/MDL [7] method.
5. Use Tucker [6] decompostion technique to obtain the optimal $\mathcal{G}, \boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \boldsymbol{U}_4$.
6. Compute the denoised HSI image, record, analyze and summarize its results.
7. Modify the K-SVD [4] code so it suits our purposes.
8. Modify the ANLM3D [8] code so it suits our purposes.
9. Modify the BM3D [9] code so it suits our purposes.
10. Modify the BM4D [10] code so it suits our purposes.
11. Design a table for recording experimental results.

12. Do denoising experiments using K-SVD [4] method, record, analyze and summarize its results.
13. Do denoising experiments using ANLM3D [8] method, record, analyze and summarize its results.
14. Do denoising experiments using BM3D [9] method, record, analyze and summarize its results.
15. Do denoising experiments using BM4D [10] method, record, analyze and summarize its results.
16. Compare all the above results.
17. Implement the code for our optimization problem in the extensions.
18. Test the code on the same noisy images, and record and analyze the result. Verify our predictions.
19. Read more relevant paper and elaborate on more extensions.

#### 3.3.2 Task Assignments

- Mingshuang Zhang: 1,4,5,8,9,12,13,16,17,18,19
- Jiachen Jiang: 1,2,5,6,9,10,13,14,17,18,19
- Chengtian Zhang: 2,3,6,7,10,11,14,15,17,18,19
- Jason Hu: 3,4,7,8,11,12,15,16,17,18,19

## 4 Extensions

Our extension consists of solving the unconstrained nonconvex and nonsmooth optimization problem whose objective function is

$$F(\boldsymbol{Z}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S) = \sum_{k=1}^{n} (\|\boldsymbol{Y}^{(k)} - \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S\|_F^2 + \lambda \|\boldsymbol{Z}^{(k)}\|_0), \tag{17}$$

where we now sum over each of the $n$ patches instead of clusters, and use global dictionaries. This problem can be rewritten as the constrained nonconvex and nonsmooth problem,

$$\begin{aligned} &\text{argmin}_{\boldsymbol{C}^{(k)}, \boldsymbol{Z}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S} \sum_{k=1}^{n} (\|\boldsymbol{Y}^{(k)} - \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times \boldsymbol{D}^S\|_F^2 + \lambda \|\boldsymbol{C}^{(k)}\|_0) \\ &s.t.\ \boldsymbol{C}^{(k)} = \boldsymbol{Z}^{(k)}. \end{aligned} \tag{18}$$

Therefore, we can construct augmented Lagrange Function as,

$$\begin{aligned} L_\mu(\boldsymbol{Z}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S; \mathcal{L}^{(k)}) = &\sum_{k=1}^{n} (\|\boldsymbol{Y}^{(k)} - \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S\|_F^2 + \lambda \|\boldsymbol{C}^{(k)}\|_0 \\ &+ \langle \mathcal{L}^{(k)}, \boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)} \rangle + \frac{\mu}{2} \|\boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)}\|_F^2), \end{aligned} \tag{19}$$

where $\mathcal{L}^{(k)}(k = 1, ..., n)$ are the Lagrange multipliers, and $\mu$ is a positive scalar which affects the convergence rate of the algorithm, but not the optimal point if it is unique. We instead maximize the Lagrange dual function:

$$\min F(\boldsymbol{Z}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S) = \max_{\mathcal{L}^{(k)}} \min_{\{\boldsymbol{Z}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S\}} L_\mu(\boldsymbol{Z}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S; \mathcal{L}^{(k)}). \quad (20)$$

Since the objective function is nonconvex, initial conditions for the variables must be appropriately chosen. This will be done as follows:

1. Run the algorithm in section 3.2 to determine $\boldsymbol{D}_W, \boldsymbol{D}_H, \boldsymbol{D}_S$.
2. Compute $\boldsymbol{Z}^{(k)}$ to be the pseudoinverse solution to $\boldsymbol{D} \cdot \mathrm{vec}(\boldsymbol{Z}^{(k)}) = \mathrm{vec}(\mathcal{Y}^{(k)})$ where $\boldsymbol{D} = \boldsymbol{D}^S \otimes \boldsymbol{D}^H \otimes \boldsymbol{D}^W$.
3. Set $\boldsymbol{C}^{(k)} = \boldsymbol{Z}^{(k)}$.
4. Initialize each $\mathcal{L}^{(k)}$ to be the array of 1s.

Next we use the ADMM method to iteratively solve the optimization problem with four steps:

1. Fix $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S, \mathcal{L}^{(k)}, \boldsymbol{C}^{(k)}$ and find the $\boldsymbol{Z}^{(k)}$ using (23).
2. Fix $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S, \mathcal{L}^{(k)}, \boldsymbol{Z}^{(k)}$ and find the $\boldsymbol{C}^{(k)}$ using (27).
3. Gradient ascend in terms of $\mathcal{L}^{(k)}$ using (28).
4. Fix $\mathcal{L}^{(k)}, \boldsymbol{Z}^{(k)}, \boldsymbol{C}^{(k)}$ and find $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$ using (30).

For the first step, the objective function is,

$$\begin{aligned} L(\boldsymbol{Z}^{(k)}) = &\|\boldsymbol{Y}^{(k)} - \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S\|_F^2 \\ &+ \langle \mathcal{L}^{(k)}, \boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)} \rangle + \frac{\mu}{2} \|\boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)}\|_F^2 \end{aligned} \quad (21)$$

It is equivalent to

$$\begin{aligned} L(\boldsymbol{Z}^{(k)}) = &\|\,\mathrm{vec}(\boldsymbol{Y}^{(k)}) - \boldsymbol{D} \cdot \mathrm{vec}(\boldsymbol{Z}^{(k)})\|_2^2 \\ &+ \langle \mathrm{vec}(\boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)}), \mathrm{vec}(\mathcal{L}^{(k)}) \rangle + \frac{\mu}{2} \|\,\mathrm{vec}(\boldsymbol{C}^{(k)}) - \mathrm{vec}(\boldsymbol{Z}^{(k)})\|_2^2, \end{aligned} \quad (22)$$

where $\boldsymbol{D} = \boldsymbol{D}^S \otimes \boldsymbol{D}^H \otimes \boldsymbol{D}^W$ and the vec operator reshapes a matrix into a vector.
We take the gradient on $L(\boldsymbol{Z}^{(k)})$ and make $\nabla L(\boldsymbol{Z}^{(k)}) = 0$ to get the solution,

$$\mathrm{vec}(\boldsymbol{Z}^{(k)})^* = (\boldsymbol{D}^T \boldsymbol{D} + \mu I)^{-1} (\mathrm{vec}(\mathcal{L}^{(k)}) + \boldsymbol{D}^T \,\mathrm{vec}(Y^{(k)}) + \mu * \mathrm{vec}(\boldsymbol{C}^{(k)})). \quad (23)$$

Finally, we can reshape the $\mathrm{vec}(\boldsymbol{Z}^{(k)})^*$ as $(\boldsymbol{Z}^{(k)})^*$.

For the second step, the objective function is,

$$L(\boldsymbol{C}^{(k)}) = \lambda \|\boldsymbol{C}^{(k)}\|_0 + \frac{\mu}{2} \|\boldsymbol{C}^{(k)} - (\boldsymbol{Z}^{(k)} - \frac{1}{\mu} \mathcal{L}^{(k)})\|_F^2 \quad (24)$$

This is equivalent to

$$L(\text{vec}(\boldsymbol{C}^{(k)})) = \lambda \| \text{vec}(\boldsymbol{C}^{(k)}) \|_0 + \frac{\mu}{2} \| \text{vec}(\boldsymbol{C}^{(k)}) - (\text{vec}(\boldsymbol{Z}^{(k)}) - \frac{1}{\mu} \text{vec}(\mathcal{L}^{(k)})) \|_2^2 \tag{25}$$

This problem has the form of proximal operator, and we know the proximal operator of $f(\boldsymbol{x}) = \lambda \|\boldsymbol{x}\|_0$ is $\mathbf{prox}_{\mu^{-1} f}(\boldsymbol{x}) = \mathbf{hard}_{\lambda\mu^{-1}}(\boldsymbol{x})$ which is

$$\mathbf{hard}_{\lambda\mu^{-1}}(x) = \begin{cases} \{0\}, & \text{if } |x| < \sqrt{\frac{2\lambda}{\mu}} \\ \{x\}, & \text{if } |x| > \sqrt{\frac{2\lambda}{\mu}} \\ \{0, x\}, & \text{if } |x| = \sqrt{\frac{2\lambda}{\mu}} \end{cases} \tag{26}$$

Therefore, the optimal solution is

$$(\boldsymbol{C}^{(k)})^* = \mathbf{hard}_{\lambda\mu^{-1}}.(\boldsymbol{Z}^{(k)} - \frac{1}{\mu} \mathcal{L}^{(k)}). \tag{27}$$

The third step is gradient ascend of $\mathcal{L}^{(k)}$,

$$\mathcal{L}^{(k)} = \mathcal{L}^{(k)} + \mu * (\boldsymbol{C}^{(k)} - \boldsymbol{Z}^{(k)}). \tag{28}$$

For the final step, first we find $\boldsymbol{D}^W$ using the objective function

$$L(\boldsymbol{D}^W) = \sum_{k=1}^{n} \| \boldsymbol{Y}_{(1)}^{(k)} - \boldsymbol{D}^W * (\boldsymbol{Z}^{(k)} \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S)_{(1)} \|_F^2, \tag{29}$$

where $\boldsymbol{Y}_{(1)}^{(k)}$ the is mode-(1) unfolding of $\boldsymbol{Y}^{(k)}$. Using vec trick of Kronecker product to change it to a sum of least squares, and letting $\boldsymbol{A}_k = (\boldsymbol{Z}^{(k)} \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S)_{(1)}^T \otimes I$, the solution is

$$\text{vec}(\boldsymbol{D}^W) = (\sum_{k=1}^{n} \boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} (\sum_{k=1}^{n} \boldsymbol{A}_k^T \text{vec}(\boldsymbol{Y}_{(1)}^{(k)})). \tag{30}$$

Then we can reshape $\text{vec}(\boldsymbol{D}^W)$ to get $\boldsymbol{D}^W$. Similarly, we can update $\boldsymbol{D}^H$ and $\boldsymbol{D}^S$.

**Algorithm 1** Algorithm for our new method

**Input:** the HSI of $\mathcal{H} \in \mathbb{R}^{W \times H \times S}$

**Output:** denoised HSI $\mathcal{H}_{de}$, dictionaries $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$

Construct groups $Y^{(k)}(k = 1, \ldots, n)$ by the extracting and unfolding process of the $\mathcal{H}$.

Initialization of $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S, \left\{\mathcal{L}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{Z}^{(k)}\right\}_{k=1}^{n}$.

**while** *not converge* **do**

  **for** $k = 1 : n$ **do**

    update $\boldsymbol{Z}^{(k)}$ by (23)

    update $\boldsymbol{C}^{(k)}$ by (27)

    update $\mathcal{L}^{(k)}$ by (28)

  **end**

  update $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$ by reshaping the result of (30)

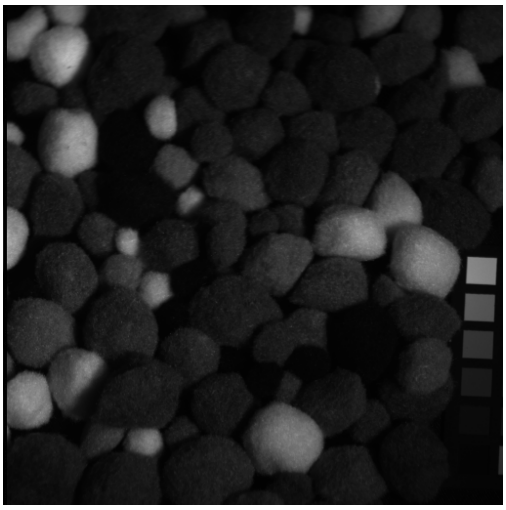
**end**

Reconstruct groups $\boldsymbol{Y}^{(k)} = \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S (k = 1, \ldots, n)$

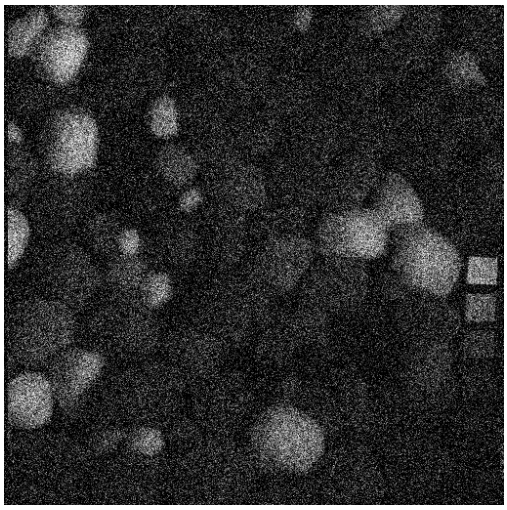
Average $\boldsymbol{Y}^{(k)}$ to form the denoised MSI $\mathcal{H}_{de}$

# 5 Comparison Methods and their Results

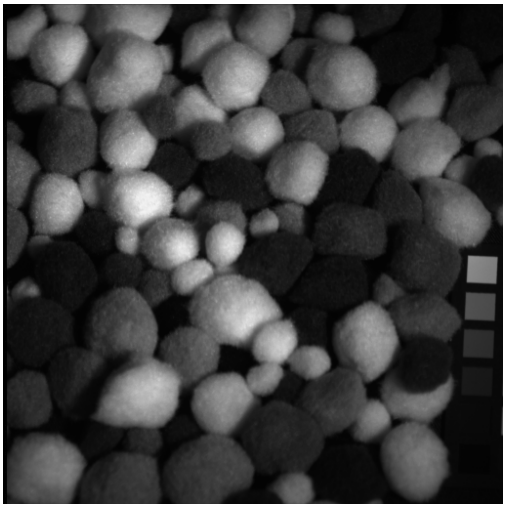
The methods we applied include K-SVD [4], BM3D [9], BM4D [10], and ANLM3D [8]. The experimental HSI image of pompom has multiple colors and different information in 31 different channels. We add zero-mean additive white Gaussian noise with standard deviation of $\sigma = 0.2$ to the HSI image (PSNR of noisy HSI is 13.98) and got the following results Fig. 1 to Fig. 5.



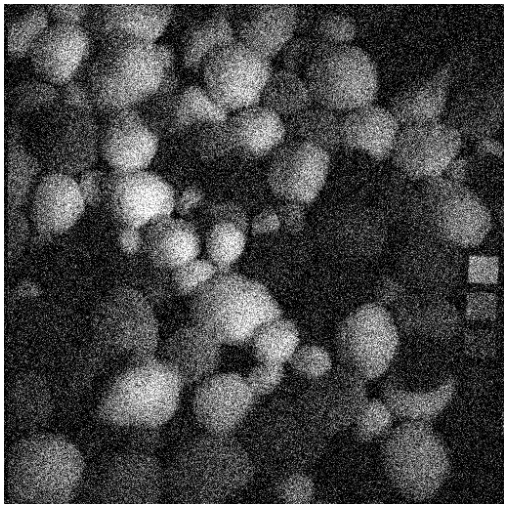
**(a)** Channel 10 No Noise



**(b)** Channel 10 Gaussian $\sigma = 0.2$

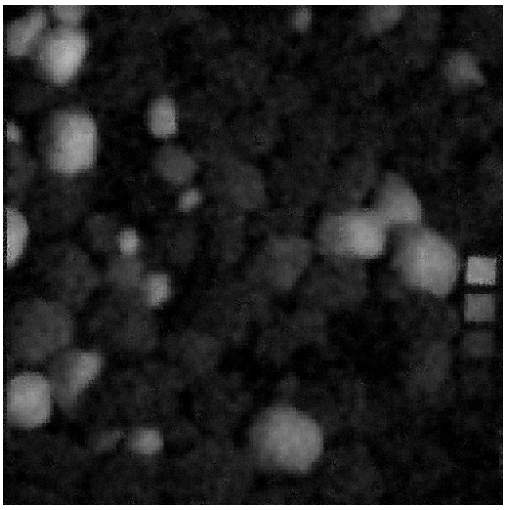


**(c)** Channel 30 No Noise

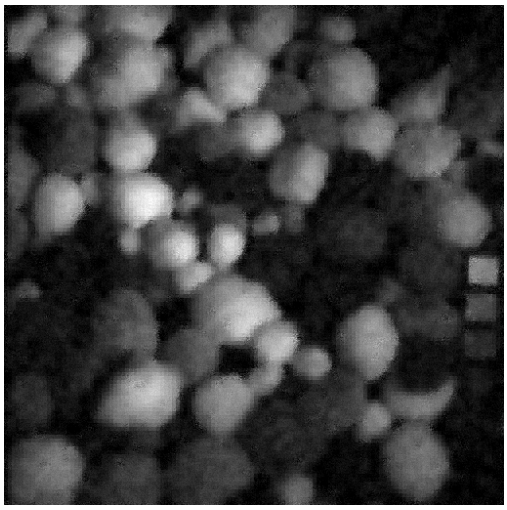


**(d)** Channel 30 Gaussian $\sigma = 0.2$

**Figure 1:** Original and noisy HSI, channel 10 and channel 30 with Gaussian noise $\sigma = 0.2$.

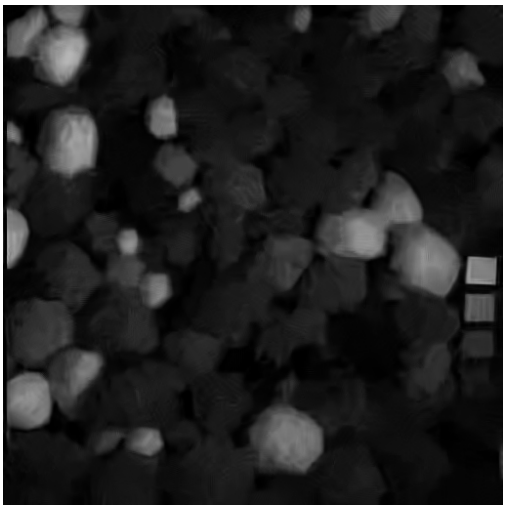


**(a)** Channel 10 K-SVD Denoised

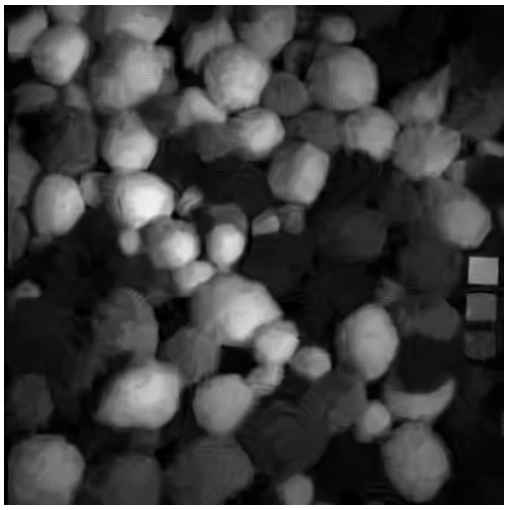


**(b)** Channel 30 K-SVD Denoised

**Figure 2:** K-SVD [4] Denoising Results for channel 10 and channel 30 with Gaussian noise $\sigma = 0.2$.

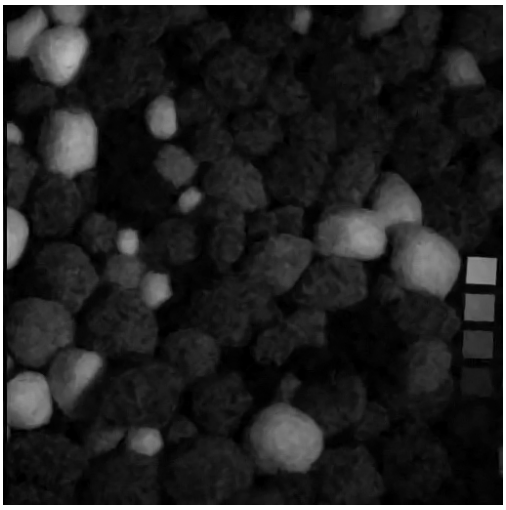


**(a)** Channel 10 BM3D Denoised

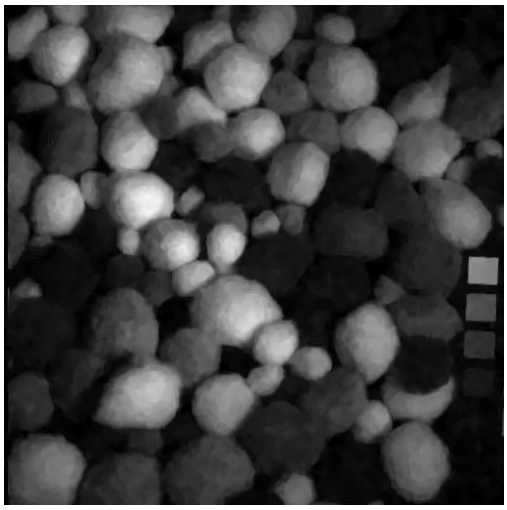


**(b)** Channel 30 BM3D Denoised

**Figure 3:** BM3D [9] Denoising Results for channel 10 and channel 30 with Gaussian noise $\sigma = 0.2$.

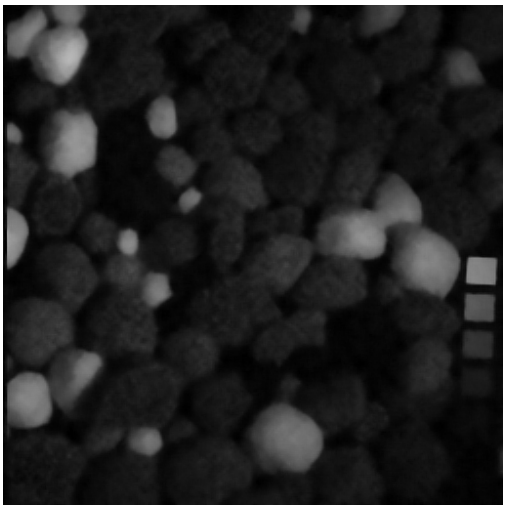


**(a)** Channel 10 BM4D Denoised

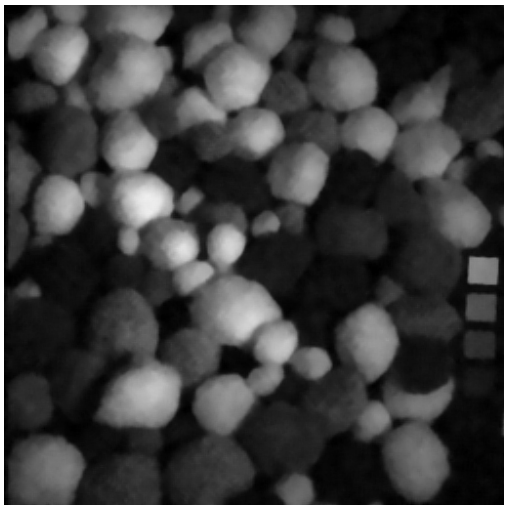


**(b)** Channel 30 BM4D Denoised

**Figure 4:** BM4D [10] Denoising Results for channel 10 and channel 30 with Gaussian noise $\sigma = 0.2$.



**(a)** Channel 10 ANLM3D Denoised



**(b)** Channel 30 ANLM3D Denoised

**Figure 5:** ANLM3D [8] Denoising Results for channel 10 and channel 30 with Gaussian noise $\sigma = 0.2$.

| Method | $\sigma = 0.1$ | | | $\sigma = 0.2$ | | | $\sigma = 0.3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR | SSIM | time(s) | PSNR | SSIM | time(s) | PSNR | SSIM | time(s) |
| Noisy HSI | 20.00 | 0.1628 | 0.00 | 13.98 | 0.0498 | 0.00 | 10.46 | 0.0225 | 0.00 |
| K-SVD | 30.27 | 0.6691 | 8.01 | 27.67 | 0.5610 | 6.30 | 26.12 | 0.4895 | 6.92 |
| BM3D | 35.45 | 0.8976 | 1.85 | 32.36 | 0.8514 | 2.43 | 30.19 | 0.8075 | 2.51 |
| BM4D | 37.44 | 0.9260 | 284.91 | 34.13 | 0.8741 | 344.70 | 32.03 | 0.8275 | 337.92 |
| ANLM3D | 35.20 | 0.9142 | 2655.70 | 32.93 | 0.8797 | 2056.98 | 31.37 | 0.8378 | 2233.65 |
| Tensor DL | 36.90 | 0.9210 | 52.59 | 34.22 | 0.8890 | 49.10 | 32.91 | 0.8646 | 53.78 |
| Our method | 30.01 | 0.8244 | 473 | 29.45 | 0.8048 | 631 | 28.68 | 0.7572 | 703 |

**Table 1:** Denoising results of different methods with Gaussian noise $\sigma = 0.1, 0.2, 0.3$.

Table 1 shows the denoising results of different methods with Gaussian noise $\sigma = 0.1, 0.2, 0.3$. As the value of $\sigma$ increases, the PSNR and SSIM of the noisy HSI decreases. As for the PSNR and the SSIM results, the Tensor DL [1] method generally outperforms all the other methods, except for the $\sigma = 0.1$ case, where its PSNR is 0.54 dB lower than the BM4D [10] method and its SSIM is 0.005 lower than the BM4D [10] method. The difference in image quality is acceptable, as the running time of the Tensor DL [1] method is much smaller than the BM4D [10] method, which suggests that it can achieve a good denoising result in a fairly short time period.

# 6 Results of the Innovation Method

## 6.1 New Method Implementation

For the initialization of the dictionaries $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$, we selected the dictionaries from the result of Tensor DL [1] method. When implementing this method, the 3D patches we used were $\boldsymbol{T} \in \mathbb{R}^{8\times8\times31}$. To decrease computation time and ensure necessary overlap of pixels, the patches were picked every 4 rows and every 4 columns, so that the overlap area of 2 adjacent patches is of size $4 \times 4$. In total, there are $n = ((W - d_w)/4+1)((H - d_h)/4+1)$ patches and the number of clusters is 162. The $\boldsymbol{D}^W \in \mathbb{R}^{8\times8}$, $\boldsymbol{D}^H \in \mathbb{R}^{8\times8}$ and $\boldsymbol{D}^S \in \mathbb{R}^{31\times3}$ we picked were the dictionaries of the clusters that contains the largest numbers of patches in width dimension, height dimension and spectral dimension, respectively. The idea of setting $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$ to be concatenated dictionaries (as opposed to a single cluster dictionary) was discarded due to runtime, as even using a concatenation of two dictionaries for each resulted in a runtime of greater than 10 minutes for a $128 \times 128 \times 31$ HSI.

During the implementation experiments of our Innovation method, we found that updating the three dictionaries $\boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S$ together in the loop results in the algorithm not converging for hundreds of iterations for an image of width and height 128 pixels. Next we tried only updating one of the dictionaries, and keeping the other two dictionaries as in the initialization. The result was that when updating only $\boldsymbol{D}^W$ or $\boldsymbol{D}^H$, the algorithm would converge, but yield a PSNR that was within 0.5 of the PSNR obtained when not updating any of the dictionaries. However, when updating only $\boldsymbol{D}^S$, the PSNR would be significantly higher as seen in the following table. Furthermore, tests with updating two of the dictionaries yielded inconsistent convergence. Thus, we chose to only update $\boldsymbol{D}^S$ in all subsequent executions of the algorithm. Table 1 shows the results of our method.

### 6.2 Code Explanation

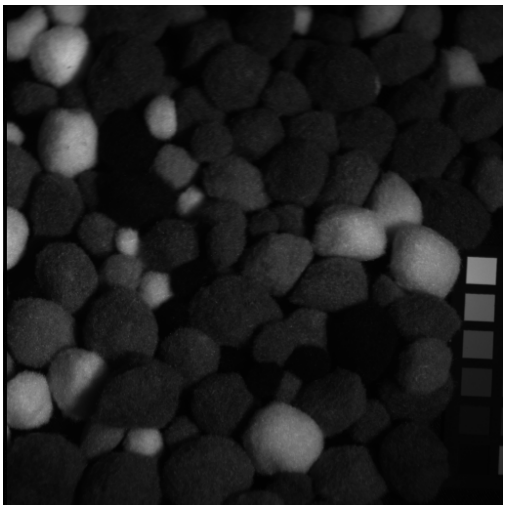
We used separate functions for each of the updating steps for the variables. To isolate the correctness of the code from the efficacy of the optimization problem, tests were conducted on each of the individual variable optimization functions to ensure that the output result was indeed optimal. This was done by first randomly initializing all the other variables. Next, the remaining variable optimal value was found using our function, with the resulting objective function being evaluated for these variables. For comparison, the variable being optimized was then randomly adjusted a number of times from its optimal value, with the objective function reevaluated each time. It was found that after every adjustment, the objective function had a higher value than the original value, demonstrating the correctness of the algorithms.

Table 1 shows the runtimes of the algorithm for each noise level. For each case, the time spent updating $\boldsymbol{Z}$ took up about half the total runtime, by far the most expensive function. This was likely because the updating function for $\boldsymbol{Z}$ involved taking a matrix inverse.

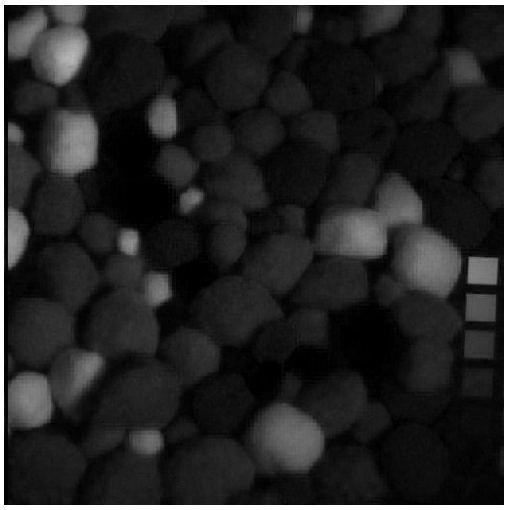
The tensor algebra functions used were tensor multiplication and taking the mode $i$ unfolding of a tensor. For the former, we used [11] as an existing implementation, and for the latter, we implemented it manually. All other functions were implemented ourselves as well.

### 6.3 New Method Results

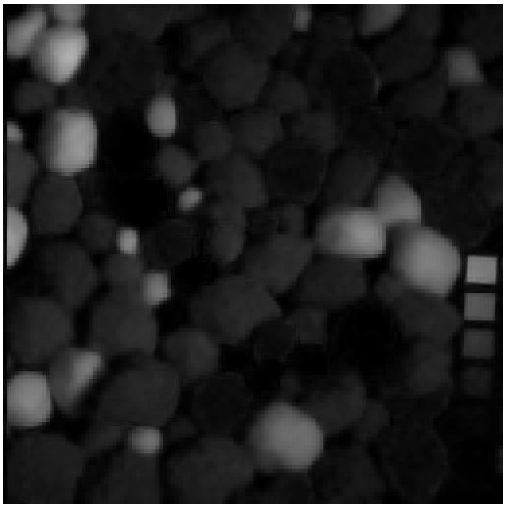
Fig. 6 shows the denoised images for $\sigma = 0.1, 0.2, 0.3$.



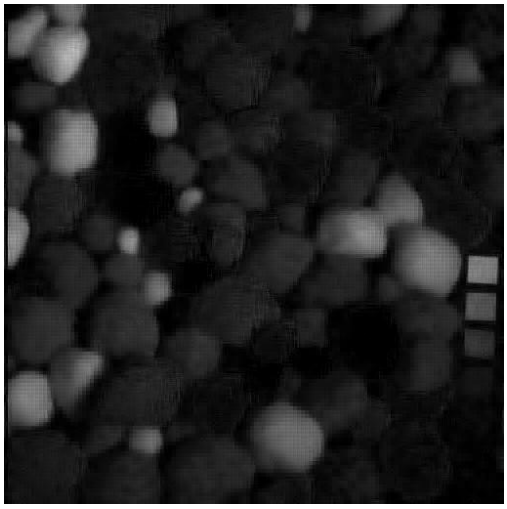
**(a)** Channel 10 Original



**(b)** Channel 10, $\sigma = 0.1$ Denoised



**(c)** Channel 10, $\sigma = 0.2$ Denoised



**(d)** Channel 10, $\sigma = 0.3$ Denoised

**Figure 6:** Denoising results for channel 10 using our innovation method.

# 7 Conclusions and Future Work

## 7.1 Evaluation of Predictions

The predictions were to obtain a PSNR 5dB higher than that of the K-SVD [4] method and a value of SSIM which was the average of 1 and that of the K-SVD [4] method. For $\sigma = 0.1, 0.2, 0.3$, this would mean getting PSNR values of 35.27dB, 32.67dB, and 31.12dB respectively, and SSIM values of 0.8345, 0.7805, and 0.7447 respectively. From the table, it can be seen that we met the prediction for SSIM with $\sigma = 0.2$ and $\sigma = 0.3$, but did not meet any of the other predictions.

The main reason for the quality being lower than expected is that we were unable to update each of the dictionaries simultaneously and still maintain consistent convergence of the algorithm. One reason for this

is that the problem is nonconvex, so a poor choice of initialization could be responsible. However, as the dictionaries are chosen using the results of the reliable and accurate Tensor DL method, this is unlikely to be the true reason why. The initializations for $\mathcal{L}$ and $\boldsymbol{C}$ were chosen somewhat arbitrarily, so this may be the reason why, although their minimization algorithms are simple and fast.

In regards to the Tensor DL method, which quantitatively performed higher in the two described metrics as well as using less time, the fundamental difference is that the Tensor DL method uses clustering and an individual dictionary for each cluster of similar patches for denoising. The conclusion is that although this method minimized the objective function for each individual cluster, as opposed to a global summed objective function for our optimization problem (a more realistic measure of how well an image is denoised), the clustering method had advantages which proved to be too difficult to overcome. Furthermore, the large number of iterations of variables needed limited the size of the dictionaries we could use. This limited size likely led to poor representation of the patches.

## 7.2 Future Improvements

The main issue of the current method was convergence issues, which greatly slowed down the algorithm, as well as prevented the iterated update of all the dictionaries at once. The next improvement would be to fix these issues, likely via a better choice of initial conditions, or other choices of parameters.

Other possibilities include changing the optimization problem in question. For instance, another possibility is to consider the objective function

$$F(\boldsymbol{Z}^{(k)}, \boldsymbol{D}^W, \boldsymbol{D}^H, \boldsymbol{D}^S) = \sum_{k=1}^{n} (\|\boldsymbol{Y}^{(k)} - \boldsymbol{Z}^{(k)} \times_1 \boldsymbol{D}^W \times_2 \boldsymbol{D}^H \times_3 \boldsymbol{D}^S\|_F^2 + \lambda \|\boldsymbol{Z}^{(k)}\|_1), \tag{31}$$

which is convex in $\boldsymbol{Z}^{(k)}$, allowing for a faster step of updating $\boldsymbol{Z}^{(k)}$. In this case it would also be necessary to enforce unit norm columns of $\boldsymbol{Z}^{(k)}$ to avoid trivial rescalings of $\boldsymbol{Z}^{(k)}$ and the dictionaries.

It would have also been possible to consider equation (17) without the Lagrange dual, avoiding the issue of the duality gap due to nonconvexity. By setting $\boldsymbol{D} = \boldsymbol{D}^S \otimes \boldsymbol{D}^H \otimes \boldsymbol{D}^W$, the objective function reduces to

$$F(\boldsymbol{Z}^{(k)}, \boldsymbol{D}) = \sum_{k=1}^{n} (\| \operatorname{vec}(\boldsymbol{Y}^{(k)}) - \boldsymbol{D} \cdot \operatorname{vec}(\boldsymbol{Z}^{(k)})\|_2^2 + \lambda \|\boldsymbol{Z}^{(k)}\|_0). \tag{32}$$

This is the traditional sparse dictionary coding problem which can be solved with orthogonal matching pursuit. Alternatively, since the solution does not have to be perfect for each iteration, the computationally faster weak matching pursuit can be used for finding $\boldsymbol{Z}^{(k)}$.

## 7.3 Code Repository

All code for the project can be found at
`https://drive.google.com/drive/folders/1mIf8AajxtiiwLXrcDW3RUDGUw2VEq70L?usp=sharing`.

# 8 Group Effort Table

Group member names (alphabetically by last name):

1. Jason Hu
2. Jiachen Jiang
3. Chengtian Zhang
4. Mingshuang Zhang

| Jason Hu<br>jashu | Jiachen Jiang<br>jiachenj | Chengtian Zhang<br>zctchn | Mingshuang Zhang<br>mingshzh | Task |
|---|---|---|---|---|
| 7 | 22 | 19 | 52 | leadership |
| 12 | 31 | 26 | 31 | planning/design |
| 50 | 14 | 14 | 22 | programming |
| 32 | 16 | 24 | 28 | testing |
| 26 | 26 | 22 | 26 | paper reading and analysis |
| 10 | 70 | 10 | 10 | innovation theory |
| 29 | 23 | 23 | 25 | communication with professor |
| 14 | 14 | 58 | 14 | code searching |
| 22 | 21 | 26 | 31 | 1st proposal |
| 25 | 25 | 25 | 25 | 2nd proposal |
| 25 | 25 | 25 | 25 | progress report |
| 31 | 15 | 26 | 28 | report |
| 100 | 0 | 0 | 0 | code repository |
| 25 | 25 | 25 | 25 | presentation |

**Table 2:** Group Project Effort

# 9 Bibliography


[1] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, and B. Zhang. "Decomposable Nonlocal Tensor Dictionary Learning for Multispectral Image Denoising". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2949–2956. DOI: 10.1109/CVPR.2014.377 (cit. on pp. 1, 2, 16).

[2] W. He, H. Zhang, L. Zhang, and H. Shen. "Hyperspectral Image Denoising via Noise-Adjusted Iterative Low-Rank Matrix Approximation". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.6 (2015), pp. 3050–3061. DOI: 10.1109/JSTARS.2015.2398433 (cit. on p. 1).

[3] X. Gong, W. Chen, and J. Chen. "A Low-Rank Tensor Dictionary Learning Method for Hyperspectral Image Denoising". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 1168–1180. DOI: 10.1109/TSP.2020.2971441 (cit. on p. 3).

[4] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation". In: *IEEE Transactions on Signal Processing* 54.11 (2006), pp. 4311–4322. DOI: 10.1109/TSP.2006.881199 (cit. on pp. 3, 7–9, 12, 14, 18).

[5] David Arthur and Sergei Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245 (cit. on pp. 4, 7, 8).

[6] S. Zubair and Wenwu Wang. "Tensor dictionary learning with sparse TUCKER decomposition". In: *2013 18th International Conference on Digital Signal Processing (DSP)*. 2013, pp. 1–6. DOI: 10.1109/ICDSP.2013.6622725 (cit. on pp. 5–8).

[7] M. Wax and T. Kailath. "Detection of signals by information theoretic criteria". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 33.2 (1985), pp. 387–392. DOI: 10.1109/TASSP.1985.1164557 (cit. on pp. 6–8).

[8] José V. Manjón, Pierrick Coupé, Luis Martí-Bonmatí, D. Louis Collins, and Montserrat Robles. "Adaptive non-local means denoising of MR images with spatially varying noise levels." In: *Journal of Magnetic Resonance Imaging Jmri* 31.1 (2010), pp. 192–203. DOI: 10.1002/jmri.22003 (cit. on pp. 7–9, 12, 15).

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering". In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095. DOI: 10.1109/TIP.2007.901238 (cit. on pp. 7–9, 12, 14).

[10] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. "Nonlocal Transform-Domain Filter for Volumetric Data Denoising and Reconstruction". In: *IEEE Transactions on Image Processing* 22.1 (2013), pp. 119–133. DOI: 10.1109/TIP.2012.2210725 (cit. on pp. 7–9, 12, 15, 16).

[11] Fabian Schneiter. "n-Mode Tensor - Matrix Product, MATLAB Central File Exchange". In: 2009. DOI: https://www.mathworks.com/matlabcentral/fileexchange/24268-n-mode-tensor-matrix-product (cit. on p. 17).