Sparse Representations of Hyperspectral Images

Thesis by
Robin Joseph Swanson

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology, Thuwal,
Kingdom of Saudi Arabia

December, 2015
The thesis of Robin Joseph Swanson is approved by the examination committee

Committee Chairperson: Wolfgang Heidrich
Committee Member: Markus Hadwiger
Committee Member: Bernard Ghanem
ABSTRACT

Sparse Representations of Hyperspectral Images

Robin Joseph Swanson

Hyperspectral image data has long been an important tool for many areas of science. The addition of spectral data yields significant improvements in areas such as object and image classification, chemical and mineral composition detection, and astronomy. Traditional capture methods for hyperspectral data often require each wavelength to be captured individually, or by sacrificing spatial resolution. Recently there have been significant improvements in snapshot hyperspectral captures using, in particular, compressed sensing methods. As we move to a compressed sensing image formation model the need for strong image priors to shape our reconstruction, as well as sparse basis become more important. Here we compare several methods for representing hyperspectral images including learned three dimensional dictionaries, sparse convolutional coding, and decomposable nonlocal tensor dictionaries. Additionally, we further explore their parameter space to identify which parameters provide the most faithful and sparse representations.
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Wolfgang Heidrich, for his support and guidance of this research. I would also like to thank Dr. Biswarup Choudry and Dr. Qiang Fu for their proofreading and critiques, Anna Frühstück and Tobias Klein for their suggestions and comments, and the members of the Visual Computing Center and KAUST and the Imager Lab at the University of British Columbia for their support.

Finally, I would like to extend my thanks to my family, Katie Black and the Black family, and my friends in Canada and around the world for their patience and support.
# TABLE OF CONTENTS

Examination Committee Approval .......................... 2  
Copyright ................................................. 3  
Abstract .................................................. 4  
Acknowledgements ........................................ 5  

1 Introduction ............................................. 8  
   1.1 Motivation .......................................... 8  
   1.2 Previous Work ...................................... 9  
   1.3 Organization ...................................... 11  

2 Characteristics of Hyperspectral Images ............... 12  
   2.1 Hyperspectral Images .............................. 12  
   2.2 Characteristics of Hyperspectral Images ........... 13  
   2.3 Data Sets and Comparison Methods ................... 17  

3 Three Dimensional Dictionary Learning ................ 21  
   3.1 Dictionary Learning ............................... 21  
   3.2 Parameters of Importance .......................... 24  
   3.3 Methods ......................................... 25  
   3.4 Results ........................................... 26  
      3.4.1 Effect of Patch-Size on Reconstruction Quality ...... 26  
      3.4.2 Effect of Overcompleteness on Sparsity ........... 29  
      3.4.3 Effect of Step-Size .............................. 31  
      3.4.4 Sample Output and Error Analysis ................... 32  
      3.4.5 Robustness to Noise ............................. 37  
   3.5 Discussion ......................................... 37  
   3.6 Conclusion ......................................... 41
Chapter 1

Introduction

In this thesis we compare three methods of representing hyperspectral data: 3D Patch based Dictionary Learning, Tensor Dictionary Learning, and 3D convolutional sparse coding by investigating their ability to solve the denoising problem, that is, their ability to faithfully reconstruct corrupted data. To illustrate their differences we evaluate the quality of their reconstruction, the sparsity of the found basis, and the time taken to compute it. Furthermore, we introduce techniques to better contrast the differences in their reconstruction.

Specifically, our contributions are:

• A thorough investigation of sparse representations of hyperspectral images,

• Apply extended metrics to evaluate the performance of these algorithms,

• Extend convolutional sparse coding to three dimensional data, and

• Identify a common error in hyperspectral dictionary methods and a simple solution to correct it

1.1 Motivation

Hyperspectral image data has long been used for both scientific and industrial uses. The additional spectral information can be used for everything from identifying veg-
etation on distant planets [1], to classifying ripe fruit [2], to detecting counterfeit medication [3]. To capture this additional spectral information traditional hyperspectral cameras have often relied on multiple exposures, filtering specific wavelengths for each capture. Recently, however, great efforts have been made to create a compressive hyperspectral camera capable of single capture hyperspectral images. These compressive sensing methods rely on their ability to sparsely represent hyperspectral images such that the full hyperspectral image can be faithfully recovered from the compressed data. This presents a real and current need for an efficient manner to represent hyperspectral data sparsely. We therefore hope to compare current methods and explore future avenues to meet this growing need.

1.2 Previous Work

While traditional hyperspectral images have existed for some time, there have been multiple works depicting compressive hyperspectral cameras. The most widely known example is the Coded Aperture Snapshot Spectral Imaging (CASSI)[4] system which employed a wavelet basis for denoising and reconstruction. More recently the Spatial-spectral Encoded Compressive Hyperspectral Imaging (SSCSI)[5] project created a similar snapshot hyperspectral camera which was encoded using a 3D patch based dictionary. Finally, many non-compressive hyperspectral imaging systems have also been produced such as HyperCam[6]. However, these methods are non-compressive and do not require a sparse basis for reconstruction and are therefore of little relevance to our goals.

Dictionary based representations of hyperspectral data has also been well explored for various purposes. Akhtar et. al[7] used Bayesian learning and an upscaling hyperspectral dictionary to learn a high resolution hyperspectral image from a low resolution hyperspectral and high resolution RGB input. Wang et. al[8] modified
the CASSI system to simultaneously capture video via a beamsplitter. They accomplished this by exploiting an overcomplete 3D dictionary learned on video data to reconstruct high-speed hyperspectral video. While these methods all exploit a 3D dictionary in their reconstruction, they often fail to motivate their parameter choices, or explain their training data and learning process in depth.

Convolutional Sparse Coding (CSC) has also made steady advances in recent years. Recent work by Bristow et. al [9], and Heide et. al [10], has shown that CSC will soon be a viable alternative to competing image representations. However, to date these works have focused on greyscale images.

A number of other image representation methods exist that have already been well explored and shown to be insufficient for hyperspectral reconstruction or denoising. Chakrabarti et. al [11] first evaluated spatio-spectral bases, but limited their research to characteristics of the images and basis without evaluating their reconstruction in depth. More recently, Lin et. al[5] showed that learned 3D patch based dictionaries were superior to representations based on 3D Fourier transform (FFT), discrete cosine transform (DCT), and principal component analysis (PCA), derived bases. Since then Peng et. al[12] further showed their tensor dictionary learning method to be superior to an extensive library of alternatives. These included K-SVD[13] using a 3D patch dictionary; BM3D[14] and BM4d[15], based on shape derived PCA; ANLM3D[16], a non-local means filter derived for medical imaging; low rank tensor approximation [17]; and hyperspectral adapted PARAFAC methods[18]. These comparisons, however, also lacked details or insight into their parameter choices. On occasion they also chose parameters which are known to be inferior, e.g., online learning of K-SVD dictionaries as opposed to pre-trained, to make their comparisons.
1.3 Organization

This thesis is organized as follows:

Chapter 2 gives a brief introduction into hyperspectral images, their characteristics, and describes our testing methods.

Chapter 3 describes the three dimensional dictionary method of representing hyperspectral images, its parameters, and how they affect the representation.

Chapter 4 describes tensor dictionary learning and its use in restoring and representing hyperspectral images.

Chapter 5 describes convolutional sparse coding of images, and introduces its extension into three dimensional data.

Chapter 6 introduces our novel hyperspectral imaging system and future use of this research.

Finally, Chapter 7 concludes the work introduced here and explores future avenues of research.
Chapter 2

Characteristics of Hyperspectral Images

2.1 Hyperspectral Images

Unlike a typical image which aims to emulate human sight with only three channels (red, green, and blue), hyperspectral images refer to any image with considerably more channels. While this number is not well defined, an image on the order of 10 channels would be referred to as multispectral, while 31 channels or higher is usually referred to as hyperspectral. These channels correspond to narrow, evenly spaced, wavelengths of light. This additional spectral information can be invaluable for many computer vision tasks such as segmentation, and object classification. In part, this is due to metamerism; two objects may appear to have identical colour to the human eye while having significantly different responses at each wavelength of light. While different, when these wavelengths are combined into an RGB or grayscale image they may appear identical. Consequently, it is impossible to extract hyperspectral data from an RGB image without additional information.
2.2 Characteristics of Hyperspectral Images

For the most part the techniques that we will investigate rely on similar principles which based on the characteristics of hyperspectral images previously examined by Chakrabarti et. al[11]. The most important characteristic being spectral similarity in local, spatial patches. Put another way, although we cannot say anything about the spatial similarity in $x$ or $y$ dimension, for small regions of natural scenes the spectral, $\lambda$, dimension will exhibit a large amount of structure. This is the key to methods such as 3D dictionaries, which build dictionaries from patches with small spatial dimensions but full spectral dimensions, and tensor decomposition methods which can approximate this spectral structure with a tensor of much lower spectral rank.

That being said, this is true only for the natural scenes we are considering in this work. Hyperspectral images, for example, of a gas light or in an astronomical setting may have sharp peaks and valleys along the spectral dimensions.

This behaviour of natural images, on the other hand, can be visualized by attempting to approximate a hyperspectral image with tensor decomposition. In the following experiments we use the lraNTD tensor decomposition algorithm[19] to vary the spectral rank of our data while holding the spatial ranks constant. This method was chosen simply for ease of use although there are many alternatives including those intended specifically for use in hyperspectral data such as the work by Liu et al[20].

First, by attempting to decompose an entire hyperspectral image we can see (Figure 2.1) that there is little relationship between spectral rank and reconstruction quality. For all three scenes the reduced rank reconstruction is of poor to acceptable PSNR and inconsistent across all spatial ranks.
Second, in Figure 2.2 we see the average reconstruction quality for patches of sizes ranging from $2 \times 2$ to $64 \times 64$ spatial pixels. Here 10 patches were chosen at random from the same three images and factorized with varying spectral ranks. In this case we can clearly see the relationship between spectral rank and reconstruction quality and that we are able to achieve a better fit in smaller patches compared to those of an entire image. We can now also see a direct relationship between rank and reconstruction. As previously mentioned, the spectral rank can be considerably reduced while maintaining a high quality representation of the patch. Only at a rank 3 approximation do we begin to see a drastic fall in reconstruction quality.
Figure 2.2: Average reconstruction quality vs the reduced rank of hyperspectral image patches of varying spatial sizes decomposed by non-negative tensor factorization.

Unfortunately, this assumption does not hold under the presence of significant noise. As additional noise is added to the image overfitting can become a real concern for smaller patches. Increasing the size of the patches used for reconstruction results in more outliers being smoothed. While reconstructions performed with larger patches are less faithful to the noisy data, they better represent the intrinsic image. Again this is shown in Figures 2.3 and 2.4, where we compare average patch reconstruction of the noisy input image when compared to the intrinsic image as opposed to itself.
Figure 2.3: Average reconstruction quality for a moderately noisy image vs the reduced rank of hyperspectral image patches of varying spatial sizes decomposed by non-negative tensor factorization.

Figure 2.4: Average reconstruction quality for an extremely noisy image vs the reduced rank of hyperspectral image patches of varying spatial sizes decomposed by non-negative tensor factorization.
Whereas in the original image the smallest patches were the obvious choice, in
the noisiest cases smaller patch sizes poorly represent the underlying images until
peaking at $32 \times 32$ before falling again. The optimal patch size is then clearly highly
dependent on the amount of noise and corrupted data you expect in your images.

### 2.3 Data Sets and Comparison Methods

In this paper we focus on real world, visible spectrum hyperspectral images. Typically
these are composed of 31 evenly spaced channels from 400nm to 700nm. While there
are several data sets that match this criteria, for our purposes we have combined data
from the Harvard[11], Stanford[21], Columbia[22], and Manchester[23] sets. These
sets were chosen for their quality, variety of images, and their real world setting (i.e.,
non satellite or GIS image data). From these images we discarded those with any
masked data and randomly selected 50 images for training data and 3 quasi-randomly
for testing. The testing set was generated by randomly selecting 3 images with the
constraint that there be an outdoor scene, a natural scene, and that each image was
from a different data set. Our chosen images, and their noisy counterparts can be
seen in Figures 2.5, 2.6, and 2.7.
Figure 2.5: Testing data Scene 20 and its noisy counterpart - An outdoor scene showing a door and brick facade. Shown here is a single wavelength (channel 15), chosen due to the scene’s high response at this wavelength.

Figure 2.6: Testing data Scene 25 and its noisy counterpart - An indoor scene showing a blank wall, counter top, and garbage bins. Shown here is a single wavelength (channel 23), chosen due to the scene’s high response at this wavelength.

Figure 2.7: Testing data Scene 41 and its noisy counterpart - A closeup photo of a white flower and its leaves. Shown here is a single wavelength (channel 18), chosen due to the scene’s high response at this wavelength.
To compare results we chose to emulate the testing environment of previously published work in order to validate their results. The first method, described by Peng et. al[12], is used to compare the Tensor and 3D patch dictionary methods. In this case our input images are corrupted with significant amounts of both Gaussian ($\sigma = 0.10$) and Poisson ($\kappa = 10$) noise. Finally, once reconstruction completed, an inverse Anscombe transform is applied. The second method, described by Heide et. al[10] in their convolutional sparse coding work, is used to compare similar methods in our work. Here the images are corrupted by randomly removing 50% of pixels and attempting to inpaint the missing data.

Throughout the work we compare the quality of restored image using the PSNR metric, as well as the time required to complete the algorithm. Please note that the average intensity value of a hyperspectral image is quite low compared to that of a natural RGB scene. For most of the scenes we consider the PSNR of a completely black image would be on the order of 20dB. Therefore it should be reasonable to expect a reconstruction of no less than 20db to be considered a success.

We also inspect the computed basis to measure its sparsity. For all of our experiments we define a basis coefficient to be sparse if its value is $\leq 10^{-4}$, and the sparsity of a basis to be the ratio of sparse elements to total elements. More specifically, the sparsity of a vector $b$ is defined as,

$$sparsity(b) = \frac{1}{n_b} \sum_{i=1}^{n_b} \text{ind}_S(b_i)$$

(2.1)

Where $n_b$ is the number of elements in $b$, and $\text{ind}_C$ is the indicator function defined on the set of constraints,

$$S = \{ v \mid |v| \leq 10^{-4} \}$$

(2.2)

This sparsity value has a total range of [0, 1], where a rating of 0 would be a
completely dense vector, and 1 a completely sparse vector.

In addition to the traditional PSNR, time, and sparsity metrics for evaluation we introduce two visualizations to inspect the characteristics of these reconstructions. The first, which we identify as the per pixel error, sums the error of each pixel along each wavelength. In effect, this produces a heat map of locations where the reconstruction is poor.

The second visualization is referred to as the wavelength error. As a compliment to the per pixel error, this is a sum of errors of each channel in an image. When plotted, each algorithm would ideally have a flat wavelength error indicating a lack of bias in the reconstruction. If a particular pattern would form in this plot it may be due to some characteristic of the training or testing data, or the algorithm itself.

To make reasonable comparisons in running time between algorithms, we have implemented or chosen implementations based in MATLAB. Additionally, when choosing maximum number of iterations or other similar parameters we have chosen constant values for all experiments in each chapter which we feel reflect values likely to be chosen in regular use.
Chapter 3

Three Dimensional Dictionary Learning

3.1 Dictionary Learning

It has been well established in literature that natural images can be well represented by learned, overcomplete dictionaries[24]. Each dictionary contains a set of patches that, when linearly combined, can sparsely represent a patch from any natural image. These techniques take advantage of two important characteristics of hyperspectral images as outlined in Chapter 2; the spatial similarity in small patches of data, and the spectral similarity along those patches.

The dictionary learning algorithm is a straightforward technique. Any number of training images can be split into $n$ overlapping patches. Then, for each input patch $x_i$, we must simultaneously find a dictionary $D$ and sparse basis $\alpha_i$ that best approximate $x_i$. Here we employ the $\ell_1$ norm for each basis to enforce a sparse result.

$$
\min \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \| x_i - D\alpha_i \|_2^2 + \lambda \| \alpha_i \|_1
$$

(3.1)

This algorithm can be implemented using common minimization techniques such as ADMM[25], or OMP[26]. There are also a number of well known toolkits that can
be used such as KSVD[13], or SPAMS[27].

To reconstruct an image given a dictionary $D$ (and find a representative sparse basis $\alpha$), we solve what is known as the lasso problem. The image is again broken into overlapping patches, and then for each block $x$ an $\alpha$ is found such that $D\alpha$ minimizes the difference between $x$ and $D\alpha$.

This technique, while originally described for 2D images, may be used for data of any dimension. By simply vectorizing each image and dictionary patch we are able to recover our 3D basis.

An example of patches from a single wavelength can be seen in Figure 3.1. Here we can see the spatial variety of patches that can be combined to represent our data. If we look at a single patch over all wavelength channels, as seen in Figure 3.2, we can see how each patch slowly varies over each wavelength emulating the behaviour of the natural scenes found in our training data.
Figure 3.1: Dictionary kernels from a single wavelength of a 3D patch dictionary. A variety of spatial patterns are available for representing our images.
Figure 3.2: Dictionary kernels along all wavelengths. Patterns slowly change as we move along the spectral dimension, emulating how our training data sets naturally behave.

3.2 Parameters of Importance

There are two main dictionary parameters which affect the speed and quality of reconstruction. The first being the patch size of the dictionary which controls the number of pixels along $x$ and $y$, as well as the number of wavelength channels long $\lambda$. There are no restrictions on the sizes of patches, however, as discussed in Chapter 2, smaller patches will result in higher quality reconstructions while also increasing the time required to find a solution.

The second parameter is referred to as the overcompleteness factor. This value controls the number of dictionary elements learned and used during reconstruction. A complete dictionary refers to a dictionary with as many dictionary elements as elements in a single patch of the dictionary. A dictionary with an overcompleteness factor of two would therefore have twice as many patches as pixel elements per patch. Increasing this factor will also increase computation time and memory requirements.

During reconstruction it is also common to overlap and average each patch to form the final representation. Commonly a step size of two or three in both the $x$ and
Another alternative would be to randomly, or quasi-randomly, select patches and once again average the results. By averaging overlapping patches we can average out individual pixel error as well as alleviate patch boundary effects which result from solving each patch individually.

### 3.3 Methods

For these experiments we follow the general guidelines as detailed in Chapter 2. To train our dictionary we decompose the entire training data set into overlapping blocks of size defined by the dictionary of interest. From those blocks one million patches are randomly chosen and used as the training data.

To train our dictionaries we used the SPAMS [27] v2.5 library and accompanying MATLAB interface. In particular, we utilized the `mexTrainDL_Memory` function which solves the dictionary learning problem while keeping all coefficients stored in memory. This results in a more time efficient algorithm at the cost of higher memory use. Where additional, algorithm specific, parameters were available we used the recommended settings. Each dictionary was learned with a strict time limit of 16 core hours while running on the Noor2 cluster\(^1\).

Our reconstruction algorithm is a straight-forward implementation of the ADMM [25] lasso algorithm. A given image can be split into overlapping patches which can then be trivially solved in parallel. Unless otherwise stated a step-size of four was chosen, averaging between each overlapping patch in the end.

\(^1\)This research made use of the resources of IT Research Computing at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia
3.4 Results

3.4.1 Effect of Patch-Size on Reconstruction Quality

Here we compare the effect of the dictionary patch size on its reconstruction. By holding the overcompleteness factor constant at 1× overcompleteness we can directly compare the effect patch size has on the reconstruction of an image. First, we examine the time required for reconstruction in Figure 3.3, then examine the quality of the same reconstructions in Figure 3.4, and finally, the sparsity of our found basis in Figure 3.5. All of these reconstructions were performed while holding the step size constant at 4 which may advantage larger patch sizes which have more overlap to average out errors. We can therefore perform the same experiments with step sizes equal to the patch size of the dictionary. These results can be found in Figures 3.7 and 3.6.

![Reconstruction Time vs. Patch Size](image)

Figure 3.3: Effect of dictionary patch size on reconstruction time. Each dictionary was trained with varying patch sizes while holding overcompleteness constant at 1×, and a constant step size of 4.
Figure 3.4: Effect of dictionary patch size on reconstruction quality. Each dictionary was trained with varying patch sizes while holding overcompleteness constant at $1 \times$, and a constant step size of 4.

Figure 3.5: Effect of dictionary patch size on reconstructed basis’ sparsity. Each dictionary was trained with varying patch sizes while holding overcompleteness constant at $1 \times$, and a constant step size of 4.
Figure 3.6: Effect of dictionary patch size on reconstruction time. Each dictionary was trained with varying patch sizes while holding overcompleteness constant at $1 \times$, and a relative step size equal to its patch size.

Figure 3.7: Effect of dictionary patch size on reconstruction quality. Each dictionary was trained with varying patch sizes while holding overcompleteness constant at $1 \times$, and a constant step size equal to its patch size.
### 3.4.2 Effect of Overcompleteness on Sparsity

Here we compare the effect of the dictionary size on its reconstruction by holding the patch size constant at $8 \times 8$ pixels and varying the overcompleteness factor from $0.5 \times$ to $8 \times$.

First we compare the time to complete the reconstruction in Figure 3.8, its effect on reconstruction quality, as seen in Figure 3.9, and finally the sparsity of the basis found after reconstruction, as seen in Figure 3.10.

![Reconstruction Time vs. Dictionary Overcompleteness](image)

**Figure 3.8:** Effect of dictionary overcompleteness on reconstruction time. Here the overcompleteness factor was varied while holding the patch size and step size constant at 8 and 4 respectively.
Figure 3.9: Effect of dictionary overcompleteness on reconstruction quality. Here the overcompleteness factor was varied while holding the patch size and step size constant at 8 and 4 respectively.

Figure 3.10: Effect of dictionary overcompleteness on reconstructed basis’ sparsity. Here the overcompleteness factor was varied while holding the patch size and step size constant at 8 and 4 respectively.
3.4.3 Effect of Step-Size

Here we compare the quality of reconstruction by varying the step size between overlapping patches in our noisiest data set. In Figure 3.11 we see the effect varying step size has on reconstruction time, and in Figure 3.12.

![Figure 3.11: Effect of overlapping patch step size on reconstruction time. Here the step size was varied while holding the patch size constant at 8 with an overcompleteness factor of $1\times$.](image-url)
3.4.4 Sample Output and Error Analysis

For the sake of comparison we show sample output from the $8 \times 8 \times 31$ spatial spectral 3D dictionary with $1 \times$ over-completeness. This dictionary was chosen due to the widespread use of $8 \times 8$ as the recommended settings for spatial dictionary size. Furthermore, as previously shown, the over-completeness factor has no additional benefit in this non-compressive setting.
Figure 3.13: Example output from the 3D Dictionary algorithm for Scene 20 with a patch size of $8 \times 8 \times 31$ and over-completeness factor of $1 \times$. Shown here is the 15th of 31 channels, chosen due to the scene's high response in this wavelength.

Figure 3.14: Example output from the 3D Dictionary algorithm for Scene 25 with a patch size of $8 \times 8 \times 31$ and over-completeness factor of $1 \times$. Shown here is the 23rd of 31 channels, chosen due to the scene's high response in this wavelength.
Figure 3.15: Example output from the 3D Dictionary algorithm for Scene 41 with a patch size of $8 \times 8 \times 31$ and over-completeness factor of $1 \times$. Shown here is the $18^{th}$ of 31 channels, chosen due to the scene's high response in this wavelength.

Finally, we show the sum of errors for the same three images. First, in Figures 3.16 3.17 3.18 the per-pixel sum of errors shows the spatial areas in which this algorithm had difficulty restoring. Second, in Figures 3.19 the normalized wavelength error shows the sum of errors in each wavelength, normalized by the number of pixels in each wavelength.
Figure 3.16: Sum of errors along the wavelength dimension from the 3D Dictionary algorithm for Scene 20 with a patchsize of 8.

Figure 3.17: Sum of errors along the wavelength dimension from the 3D Dictionary algorithm for Scene 41 with a patchsize of 8.
Figure 3.18: Sum of errors along the wavelength dimension from the 3D Dictionary algorithm for Scene 41 with a patchsize of 8.

Figure 3.19: Sum of errors for each wavelength from the $8 \times 8 \times 31$, 1× over-complete 3D Dictionary for all three scenes. Errors are summed for each wavelength and normalized by the number of spatial pixels.
3.4.5 Robustness to Noise

Our final test is to measure this algorithm's performance as noise increases. Using two patch sizes, 8 and 14, we slowly increase the amount of Gaussian noise from $\sigma = 0.0$ to $\sigma = 4.0$ examining the outputs PSNR and visual qualities. In Figure 3.20 we plot the PSNR of all three scenes with both patch sizes as the noise increases. In Figure 3.21 we can visually inspect the effect this increased noise has on the output.

![Figure 3.20: Effect of noise on the 3D dictionary algorithm for a patch size of 8 and 14.](image)

3.5 Discussion

Clearly there is a large and direct trade-off between quality and reconstruction time due to the patch size of our dictionary. Relatively large patches are able to better reconstruct our hyperspectral data; however, this comes at a considerable time cost and with no reason to expect additional sparsity. This increase holds true for both
constant and relative step sizes which follows directly from the experiments performed in Chapter 2 for noisy data. Unfortunately, due to computational restrictions for such large dictionaries, it’s unclear how higher patch sizes would behave. Intuitively there will be a peak in reconstruction quality and then a steady decline thereafter which, from Scene 20 in particular, appears to be leveling off by size 14. The difference in errors between constant and relative step size are marginal which suggests averaging overlapping patches may not be as important as emphasized by previous work, but may only be true for such noisy data.

As expected, by increasing the over-completeness factor we can see an immediate increase in reconstruction time. This is due to the larger number of coefficients that must be solved for each image patch. Additionally, after reaching completeness (i.e, a factor of unity or greater), the reconstruction quality is not greatly affected by additional dictionary elements. This is due to the fact that all additional patches will be increasingly redundant, or simple combinations of previous patches. The slight decline in reconstruction quality is likely due to the maximum number of iterations allowed during minimization. While this number may have been large enough for a 1× overcomplete dictionary, it is not surprising that the (up to 8 fold) increase in coefficients requires additional iterations. Therefore, in this non-compressive experiment, a higher sparsity reconstruction will have no effect on the result. Finally, due to the increased number of coefficients and redundant patches, we see an immediate increase of sparsity in the reconstructed coefficients.

The step size had a surprisingly marginal effect on the reconstruction quality. Considering the doubled reconstruction time and minuscule increase in PSNR (≤ 0.1dB), it would seem that a step size of 2 may not be worth the extra computation time compared to a step size of 4. As this experiment was run on a dictionary with patches of size 8 × 8, this indicates that a step size of approximately half the spatial patch dimensions is an appropriate choice.
From the sample output and corresponding sum of errors in Figures 3.16, 3.17, and 3.18 we can see that there are no specific areas that this algorithm has difficulties representing. The error reflected in these charts are consistent across the entire image and in line with the amount of noise present. In the per-wavelength sum of errors seen in Figure 3.19, a pattern emerges with higher errors in specific wavelengths in all three scenes. Due to the consistency between scenes and dictionaries, this may indicate either an inadequate amount of training data or data that may be biased in some wavelengths.

Upon further investigation this pattern follows exactly the mean of the training data, as seen in Figure 3.22. As a potential solution we trained an additional $8 \times 8 \times 31$, $1 \times$ overcomplete dictionary where the input data was normalized by wavelength. In other words, the mean of each wavelength across the entire training data set was normalized to zero with a standard deviation of one as opposed to patch normalization which was previously recommended by literature. The difference between the un-normalized and normalized wavelengths can be seen in Figure 3.22.
Figure 3.22: Per-wavelength mean of training data set before and after wavelength normalization.

Comparison between these otherwise identical dictionaries shows an immediate improvement in reconstruction quality. The per-wavelength error is both reduced and more consistent across each channel for all three test scenes. An exact comparison of PSNR values can be found in Table 3.1.

The cause of the error spike in the first channel can be found by examining the training data, learned dictionary, and input images. Examining our data channel by channel shows a significant decrease in image quality in the first channel from many of the data sets. This is not surprising, however, as the first channel corresponds to the 400nm wavelength which is at the very edge of visible violet light, and also very difficult for camera sensors to capture. This would result in considerable noise and loss of image quality which has manifested itself directly in our learned dictionary. Visually inspecting Figure 3.1 we can see that the first channel is, while similar to other kernels in its wavelength, substantially noisier. In the future, it may be worth
<table>
<thead>
<tr>
<th>Scene</th>
<th>Patch Normalized PSNR (dB)</th>
<th>Wavelength Normalized PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>27.0777</td>
<td>27.4674</td>
</tr>
<tr>
<td>25</td>
<td>27.3805</td>
<td>28.0473</td>
</tr>
<tr>
<td>41</td>
<td>24.8428</td>
<td>26.3678</td>
</tr>
</tbody>
</table>

Table 3.1: PSNR improvements after applying wavelength normalization to the dictionary training process when compared to a similarly trained $8 \times 8 \times 31$, 1× overcomplete 3D dictionary.

investigating some way of mitigating these artifacts through denoising, or by other means of image reconstruction.

Finally, inspecting the results from the robustness to noise in Figure 3.20 we can see that for a patch size of 14 by $\sigma = 1.0$ we have an unacceptable PSNR for all three test scenes. In Figure 3.21, however, at $\sigma = 1.0$ or even 2.0, Scene 20 and 41 are identifiable, though once we have reached $\sigma = 4.0$ the scenes are completely unrecognizable. While there appears to be a large difference in quality between Scenes 20 and 41 compared to 25, their PSNR are nearly identical for all noise levels. This could be attributed to higher wavelength accuracy while spatial quality declined. The large amount of homogeneous regions and lack of high frequency detail may be the reason for the lack of spatial reconstruction quality in Scene 25 compared to 20 and 41.

### 3.6 Conclusion

Here we have thoroughly explored the 3D dictionary parameter space, identified a common error in hyperspectral dictionary learning, and proposed a simple modification to improve reconstruction results. For maximum reconstruction quality as small of a patch size and step size as computationally feasible are recommended. For the most sparse results it is clear that an increase in overcompleteness is crucial, but the effect of patch size is unclear. In all cases a wavelength normalized dictionary is
recommended for best results.
Figure 3.21: Visual effect of noise on the 3D dictionary algorithm for all three test scenes. From top to bottom we increase the amount of Gaussian noise where \( \sigma = 0.0, 0.2, 0.4, 0.8, 1.0, 2.0, 4.0 \).
Figure 3.23: Comparison between results from two dictionaries: one trained on patch normalized input data and the other on wavelength normalized input data. The dictionaries trained on wavelength normalized data have lower overall error and less wavelength biased error.
Chapter 4

Tensor Dictionary Learning

4.1 Introduction

Decomposable Nonlocal Tensor Dictionary Learning [12] (TDL) is a very recent and novel approach to representing and denoising multi and hyperspectral images. TDL first breaks a given hyperspectral image into overlapping patches, similar to the 3D patch based dictionary seen in Chapter 3. However, by then grouping visually similar patches together TDL is better able to exploit non-local spatial similarity and efficiently compute representations for similar patches regardless of their spatial location. Given that these groups of patches are now both similar in the spatial and wavelength dimensions, they can be well factorized by tensor approximations, the parameters of which are chosen using AIC/DIC [28].

To summarize their algorithm, the image is first broken into overlapping patches similar to the 3D patch dictionary seen earlier. Second, the patches are grouped together via k-means clustering. Each of these groups are then factorized by tensor approximation.

This final process can be represented by the following minimization problem for all $k$ groups of patches,
\[
\min_{U_1, U_2, U_3, U_4, G} \| \chi^{(k)} - G \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 U_4 \| \tag{4.1}
\]

where \( U_1 \in \mathbb{R}^{d_{Wk} \times r_{Wk}} \), \( U_2 \in \mathbb{R}^{d_{Hk} \times r_{Hk}} \), \( U_3 \in \mathbb{R}^{d_{yk} \times r_{yk}} \), \( U_4 \in \mathbb{R}^{d_{Nk} \times r_{Nk}} \), and the Tucker core \( G \in \mathbb{R}^{r_{Wk} \times r_{Hk} \times r_{yk} \times r_{Nk}} \). Here \( W, H, \lambda \) represent the width, height, and number of wavelengths of all \( N \) patches in our tensor group, \( d_{yk} \) is the actual dimension along \( y \), and \( r_{yk} \leq d_{yk} \) is the chosen reduced rank approximation along dimension \( y \). Thus by choosing an \( r_{yk} \) strictly less than \( d_{yk} \) we can enforce a rank reduction along the spectral dimension.

### 4.2 Methods

Unlike the other methods we discuss, TDL learns its dictionary online and thus requires no dictionary to be learned before use. Therefore, we may simply evaluate its performance on all three of our noisy test scenes. For best results we use the provided code and recommended parameters provided by the authors, varying only the size of the patches used during reconstruction. For the sake of comparison we chose patch sizes in the same range as those used for the 3D patch based dictionary described in Chapter 3.

### 4.3 Results

Varying only the spatial size of overlapping patches used in the algorithm we can see it’s effect on reconstruction quality 4.1, sparsity 4.2, and reconstruction time 4.3.
Figure 4.1: Example output from TDL algorithm on Scene 20 with a patch size of 8. Shown here is the 15th of 31 channels.

Figure 4.2: Example output from TDL algorithm on Scene 25 with a patch size of 8. Shown here is the 15th of 31 channels.
Figure 4.3: Example output from TDL algorithm on Scene 41 with a patch size of 8. Shown here is the 15th of 31 channels.

As an example of the resulting output we include all three images with the recommended patch size of 8.
Figure 4.4: Example output from TDL algorithm on Scene 20 with a patch size of 8. Shown here is the 15th of 31 channels, chosen due to the scenes high response in this wavelength.

Figure 4.5: Example output from TDL algorithm on Scene 25 with a patch size of 8. Shown here is the 23rd of 31 channels, chosen due to the scenes high response in this wavelength.
Figure 4.6: Example output from TDL algorithm on Scene 41 with a patch size of 8. Shown here is the 18th of 31 channels, chosen due to the scene's high response in this wavelength.

Next we can compare the sum of errors along all wavelengths for all three images as well as the normalized error per wavelength. Here we compare results for all three images computed using the recommended patch size of 8.
Figure 4.7: Sum of errors along the wavelength dimension from TDL algorithm on Scene 20 with a patch size of 8.

Figure 4.8: Sum of errors along the wavelength dimension from TDL algorithm on Scene 25 with a patch size of 8.
Figure 4.9: Sum of errors along the wavelength dimension from TDL algorithm on Scene 41 with a patch size of 8.

Figure 4.10: Sum of errors for each wavelength from TDL algorithm for all three scenes. Errors are summed for each wavelength and normalized by the number of spatial pixels.
Finally, we can investigate the TDL algorithm’s robustness to noise. Here we investigate the reconstruction quality as the noise increases. Starting from zero Gaussian noise ($\sigma = 0$) we increased the noise from moderate to extreme amounts ($\sigma = 4$). In Figure 4.11 we can see the effect of noise for a patch size of both 8 and 14 and in Figure 4.12 we can see how all three scenes are visually affected.

![TDL Robustness to Noise](image)

Figure 4.11: Effect of noise on the TDL algorithm for a patch size of 8 and 14.

### 4.4 Comparison with 3D Patch Dictionary

TDL offers a clear improvement over 3D Dictionaries in both reconstruction quality and performance. In all three test scenes TDL was able to achieve between a 5 and 10 dB improvement over the best 3D Dictionary results while having a more sparse basis, and in significantly less time as well.

Comparing cropped insets of the two results in Figures 4.13, 4.14, and 4.15, we can see that there is a significant difference in the appearance of the two results.
While the TDL results appear more blurred, particularly in high frequency areas, the 3D Dictionary output preserves many of these sharp details such as the door number in Scene 20. This assessment is also corroborated by comparing per-pixel sum of errors between algorithms. Whereas the 3D Dictionary errors appear to be largely due to Poisson noise peaks, the TDL algorithm was able to remove this noise at the expense of fine detail. Finally, we can also see the effect of tensor decomposition in the resulting image by way of banding artifacts in individual patches which haven’t been compensated for by averaging overlapping pixels.

When observing the effect of increasing noise on both algorithms it is evident that TDL is much more robust to noise. Even in our most extreme test case with Gaussian noise $\sigma = 4.0$ the TDL output was recognizable whereas 3D Dictionary is unrecognizable at $\sigma = 2.0$.

\section{4.5 Discussion}

With TDL there is a clear and simple relationship between patch size, sparsity, and time. While the reconstruction quality varied very little for each image across all patch sizes ($\leq 1$ dB), sparsity was consistently reduced by increasing the patch size and in one case (Scene 20), drastically so. Due to the fact that the patch overlap size is determined by the patch size (defaulting to half the patch size), the reconstruction time also drastically decreases as the patch size is increased. Our findings show good agreement that the recommended patch size of 8 be used due to marginal changes in PSNR and sparsity, while halving the reconstruction time compared to a patch size of 6.

Examining the output images, it is evident that we are using a patch based algorithm. Scene 25, shown in Figure 4.5, in particular has a surprising amount of blocking artifacts considering its high PSNR. Furthermore Scene 20, shown in Figure
4.4, which appears considerably less blocky, has a considerably lower PSNR. However, by inspecting the sum of errors chart for Scene 20 in Figure 4.7 we can see their are significant errors in high frequency areas such as the brick details.

Unlike the 3D Dictionary, the per-wavelength error (as shown in Figure 4.10) is significantly different for each input scene. Much like the 3D Dictionary however, this error is caused by the mean of the training data which, as this is an online dictionary learning tool, is the input test scene itself. Nevertheless, as we can see in Figure 4.17, all three scenes’ errors correspond directly to their mean.

![Scene Wavelength Mean](image)

**Figure 4.17:** Mean value of each wavelength for all three scenes.

To correct for this error we can pre-process the input data as described for 3D Dictionaries in Chapter 3. Unfortunately, as our input image is corrupted by noise, adding a wavelength correction pre-process only has a marginal effect on reconstruction. As we can see in Figure 4.18, the wavelength dependent bias is improved in all three test scenes. Exact PSNR changes can be found in Table 4.1.

Examining the effect of noise on TDL we can see that even with extreme amounts
Table 4.1: PSNR improvements after applying wavelength normalization to the TDL algorithm with default parameters.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Default TDL PSNR (dB)</th>
<th>Wavelength Norm TDL PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>32.3638</td>
<td>33.7945</td>
</tr>
<tr>
<td>25</td>
<td>40.4183</td>
<td>41.9813</td>
</tr>
<tr>
<td>41</td>
<td>38.0818</td>
<td>38.8458</td>
</tr>
</tbody>
</table>

of noise many visual characteristics of the underlying image are visible. In the case of Scene 20 even at the highest amount of noise the scene is completely identifiable. Furthermore, as predicted in Chapter 2, the increased patch size has a noticeable effect for noisier images. While the PSNR might be considered very high for such a noisy scene, as we mentioned in Chapter 2, a completely black image will have a PSNR of approximately 20 dB. Therefore, only once we have reached $\sigma = 4.0$ do any of the scenes have unreasonable reconstructions.

### 4.6 Conclusion

Based on the results from our experiments we can see that TDL is able to well represent all three of our hyperspectral test images. The TDL algorithm was also very fast, extremely robust to noise, and was able to provide a sparse basis for all test images when used with a small patch size. However, when scenes contain large areas of high frequency information, such as Scene 20, it does develop noticeable patch and tensor artifacts.
Figure 4.12: Visual effect of noise on the TDL algorithm for all three test scenes. From top to bottom we increase the amount of Gaussian noise where $\sigma = 0.0, 0.2, 0.4, 0.8, 1.0, 2.0, 4.0$. 
Figure 4.13: Comparison between TDL and 3D Dictionary output on Scene 20. Five cropped sections from Channel 15 of the image are shown top to bottom. From left to right we show (a) the input image, (b) the input image corrupted by noise, and the image reconstructed by (c) Tensor Dictionary Learning and (d) 3D Dictionary Learning. Please note that the image has been scaled for clarity.
Figure 4.14: Comparison between TDL and 3D Dictionary output on Scene 25. Five cropped sections from Channel 23 of the image are shown top to bottom. From left to right we show (a) the input image, (b) the input image corrupted by noise, and the image reconstructed by (c) Tensor Dictionary Learning and (d) 3D Dictionary Learning. Please note that the image has been scaled for clarity.
Figure 4.15: Comparison between TDL and 3D Dictionary output on Scene 41. Five cropped sections from Channel 18 of the image are shown top to bottom. From left to right we show (a) the input image, (b) the input image corrupted by noise, and the image reconstructed by (c) Tensor Dictionary Learning and (d) 3D Dictionary Learning. Please note that the image has been scaled for clarity.
Figure 4.16: Visual comparison due to the effect of noise on the TDL and 3D dictionary algorithms. From top to bottom we increase the amount of Gaussian noise where $\sigma = 0.0, 0.2, 0.4, 0.8, 1.0, 2.0, 4.0$. 

(a) Noisy Input  (b) TDL  (c) 3D Dictionary
Figure 4.18: Comparison between results obtained with default input and a wavelength normalization pre-processing step. Sum of errors for each wavelength from the TDL algorithm are shown for all three scenes. Errors are summed for each wavelength and normalized by the number of spatial pixels.
Chapter 5

Hyperspectral Convolutional Sparse Coding

5.1 Convolutional Sparse Coding

Recent advances, such as the work by Bristow et. al [9], and Heide et. al [10], have moved convolutional sparse coding (CSC) into the spotlight for a number of machine learning and computer vision tasks. Similar to spatial image dictionaries explored in Chapter 3, a dictionary of patches is learned that can be used to represent any image. In this case, however, the patches are convoluted with a sparse image scale basis. In addition to being faster, these creates image scale features eliminate patch-sized artifacts that often appear in spatial patch dictionaries.

More formally, the traditional 2D CSC problem is expressed by,

$$\min_{d,z} \frac{1}{2} \|x - \sum_{k=1}^{K} d_k * z_k\|_2^2 + \beta \sum_{k=1}^{K} \|z_k\|_1 \tag{5.1}$$

subject to \( \|d_k\|_2 \leq 1 \quad \forall \ k \in \{1, \ldots, K\} \)

Where \( x \in \mathbb{R}^D \) is the data, or image, we wish to represent, and \( z_k \in \mathbb{R}^D \) are the sparse feature maps for each \( d_k \in \mathbb{R}^M \) dictionary patches. Here \( M \) is the size of the
image we wish to represent, $D$ the size of all $K$ dictionary patches, and $*$ is the 2D convolution operator.

For our purposes we will base our work on the most recent efforts of Heide et. al [10] who reformulate the problem as such,

$$
\min_{d,z} \frac{1}{2}||x - M \sum_{k=1}^{K} d_k \ast z_k||_2^2 + \beta \sum_{k=1}^{K} ||z_k||_1 + \sum_{k=1}^{K} ind_C(d_k)
$$

(5.2)

Here $M$ is a diagonal or block-diagonal matrix which decouples linear systems of the form $(M^T M + I)x = b$ into smaller, easier to solve independent subproblems, and $ind_C$ is the indicator function defined on the convex set of the constraints $C = \{ v \mid ||Sv||_2 \leq 1 \}$. Further details and motivations for these changes can be found in their paper.

While these methods have proved quite effective for greyscale images, currently there has been no work investigating their application to hyperspectral data sets. In this section we provide some preliminary investigations and results of our work extending CSC to hyperspectral images and its current limitations.

### 5.2 Extension for Hyperspectral Images

There are several possible ways to extend CSC along a third dimensions. First, it would be possible to learn 3D feature patches and to treat $x$, $d$, and $z$ as rank three tensors, applying $*$ as a 3D convolution. However, due to the structure of hyperspectral data, and the considerably reduced rank along the wavelength dimension, there is little reason to believe image scale structure, similar to those found along the spatial dimensions, would be exploitable in as few as 31 bands. In practice, we also found it infeasible to train dictionaries of 3D patches with more than 7 channels due to memory constraints. Furthermore, likely due to the limited number of bands, our dictionaries failed to converge to sensible solutions, quickly diverging before enough
ADMM iterations had been taken to create sharp patch features.

Second, and true for our implementation, we instead simultaneously learn a set of 2D dictionaries, one for each wavelength. Consequently, during reconstruction we simultaneously learn a sparse support basis for each wavelength. This method results in high quality dictionaries in every wavelength with sharp Gabor like features expected from 2D methods.

Third, a 3D dictionary could be learned while restricting the basis to two dimensions, convolving the same coefficients along all wavelengths. Intuitively this approach would be most capable of encoding 3D information in dictionary kernels while also alleviating memory and computation requirements by reducing the number of coefficients stored and updated. Currently however, this approach is not yet fully realized and as such is left for future work.

5.3 Memory Analysis

Unlike more mature representation techniques, such as 3D Dictionary Learning, initial implementations of CSC algorithms are less optimized for memory and CPU consumption. It should come as no surprise then that the methods for training our patch based dictionaries may not be applicable here. Here we show a simplified analysis of memory consumption in the previously described CSC dictionary learning algorithm.

Let $\mathbf{b}$ be the input vector of images, $\mathbf{d}$ the dictionary to be learned, and $\mathbf{z}$ the coefficients simultaneously learned for each dictionary element. If $M$ is the number of input images, each of size $N$ pixels along $\lambda$ wavelengths, and as before $K$ is the
number of dictionary elements it can be show that,

\[ b \in \mathcal{O}(N \times \lambda \times m) \]
\[ d \in \mathcal{O}(N \times \lambda \times K) \]
\[ z \in \mathcal{O}(N \times \lambda \times K \times m) \]

Additionally, while learning a dictionary we must compute the Fourier transform of both \( \mathbf{d} \) and \( \mathbf{z} \), \( \hat{\mathbf{d}} \) and \( \hat{\mathbf{z}} \) respectively, which must be stored as double precision complex values. Note that as \( \hat{\mathbf{z}} \) is both dense and must be stored for computation an implementation based on sparse matrices is not feasible.

A low estimate of the total memory used would then be approximately \( \mathcal{O}(b+3d+3z) = \mathcal{O}(N\lambda(m+3K(1+m))) \). If we were to learn a dictionary with 100 elements, as recommended in [10], using our previous training data of 50 hyperspectral images with 31 wavelengths, each approximately 1.5 megapixels in size and, assuming 8 bits per pixel, we would need approximately,

\[ \text{Total Memory} \in \Omega((N\lambda(m+3K(1+m)))) \]
\[ = 5.7102 \times 10^{12} \text{ bits} \]
\[ \approx 714 \text{ GB} \]

Which is, despite being a conservative estimate, unfortunately an infeasible amount of RAM for our purposes. Therefore our methods and data sets for training the CSC dictionaries are different than those used in previous sections and are described in further detail below.
5.4 Methods

The algorithms used in this chapter are largely based off the work and code provided by [10]. In addition to extending their algorithm to allow for multiple channels (the algorithm was designed for grayscale images only), we have also made considerable changes to favour a reduced memory footprint over CPU performance. This includes, for example, removing pre-computations and other caching in favour of recomputing the values at each iteration.

Due to the memory constraints previously described we have reduced the set of input training data to six images with 31 channels and further reduced their size to $100 \times 100$ pixels. This spatial reduction was made so that more input images could be used, allowing for a wider range of sample images to learn a more robust dictionary.

Consistent with previous CSC papers and implementations, we pre-process each image individually via contrast normalization before both learning and reconstruction. Unfortunately this puts us in a situation where we cannot apply what we have learned in previous sections about wavelength normalization. Since the completion of this work, however, more recent advances have been made which are capable of avoiding this contrast normalization step. In the future, it will be worth exploring the effect that this change, and the addition of wavelength normalization, could have on our results.

In addition to our 3D CSC methods we compare our results with a 2D dictionary where each wavelength is solved independently. Due to the inherent independence of each of these subproblems, we solve them in parallel on the same number of cores provided to the 3D case.
5.5 Dictionary Initialization

Although our method solves for all dictionary kernels and coefficients simultaneously, it lacks any constraint requiring each wavelength from a dictionary kernel to be used at a given coefficient position. In this chapter we show how we can exploit the construction of the dictionary to impose this type of solution automatically.

While there are many ways to initialize our dictionary and coefficients during the dictionary learning process, here we emphasize two specific scenarios. The first, referred to from now on as \textit{all random} initialization, describes a dictionary which was initialized with random variables for all coefficient and kernel values across all wavelengths. The second, referred to from now on as \textit{repeated random} initialization, describes a dictionary for which one wavelength’s coefficients and kernels were randomly initialized and then repeated for each additional channel.

5.5.1 Effect on Dictionary Kernels

The manner in which the dictionary and its coefficients are initialized has an obvious effect on the resulting dictionary. A single wavelength from either dictionary, shown in Figures 5.1 and 5.2, makes no clear distinction between the two.
Figure 5.1: A single wavelength (400nm) of an all random initialized CSC dictionary.

Figure 5.2: A single wavelength (400nm) of a repeated random initialized CSC dictionary.
Furthermore, the kernels in an all random dictionary, as seen in Figure 5.3, have no clear ordering or similarity across channels. The repeated random dictionary on the other hand, seen in Figure 5.4, is more similar to the 3D patch dictionary seen in Chapter 3. The same kernel along each wavelength appears to contain a high degree of similarity.

Figure 5.3: 10 kernels across all 31 wavelengths of an all random initialized CSC dictionary.

Figure 5.4: 10 kernels across all 31 wavelengths of a repeated random initialized CSC dictionary.
5.5.2 Effect on Reconstruction Coefficients

The type of dictionary initialization also has a distinct effect on the coefficients found when reconstructing the image. Here we compare the magnitude and location of coefficients from all three dictionary types: repeated 2D (Figure 5.5), repeated random (Figure 5.7), and all random (Figure 5.6). For clarity we restrict the number of wavelengths to seven evenly spaced channels, and draw only the 1000 most significant coefficients, i.e., the 1000 coefficients with largest magnitude across all dictionary kernels. Additionally, we have included vertical dashed bars to discriminate between adjacent kernels.

![Coefficients of 2D CSC](image)

Figure 5.5: 1000 most significant coefficients of reconstructed basis using a single 2D CSC dictionary for all wavelengths.
Figure 5.6: 1000 most significant coefficients of reconstructed basis using an all random CSC dictionary.
Figure 5.7: 1000 most significant coefficients of reconstructed basis using a repeated random CSC dictionary.

5.6 Results

To compare the three CSC dictionaries we apply the same metrics described earlier. For all three scenes we can compare their reconstruction quality, time, and sparsity, which can be found in Figures 5.8, 5.9, and 5.10 respectively.
Figure 5.8: Image reconstruction quality for all scenes and dictionaries.

Figure 5.9: Image reconstruction time for all scenes and dictionaries.
Figure 5.10: Image reconstruction sparsity for all scenes and dictionaries.

For a given scene we can also compare their per-pixel and wavelength dependent errors in Figures 5.11, 5.12, and 5.13; and Figure 5.14 respectively.
Figure 5.11: Per-pixel errors from Scene 20 with the 2d dictionary.

Figure 5.12: Per-pixel errors from Scene 20 with the all random dictionary.
Figure 5.13: Per-pixel errors from Scene 20 with the repeated random dictionary.

Figure 5.14: Wavelength Error across all 31 channels for all three dictionaries on Scene 20.
Finally, given the nature of the convolutional sparse coding coefficients, i.e., their image scale, we can examine their placement and magnitude. In the Figures we sum the absolute value of coefficients from all dictionary kernels to examine where they are placed and what they emphasize.

Figure 5.15: Sum of basis coefficients from Scene 20 with the 2d CSC dictionary.
Figure 5.16: Sum of basis coefficients from Scene 20 with the all random CSC dictionary.

Figure 5.17: Sum of basis coefficients from Scene 20 with the repeated random CSC dictionary.
5.7 Discussion

From the coefficient maps found in Section 5.5.2 we can clearly see that common coefficients can be exploited by creating dictionaries in a specific way. While the 2D dictionary used an even spread of coefficients throughout the dictionary, the repeated random and all random dictionaries clearly emphasized certain kernels across all wavelengths as you would expect from a 3D dictionary.

Exploring the per wavelength error for all three dictionaries, we can see that the 2D implementation has a flat error across all channels. This is not surprising as each channel is independently solved and, as all channels are usually similar, should have roughly equivalent error. The two 3D dictionaries, on the other hand, roughly follow a similar spectral error pattern. Unlike the 3D patch and tensor dictionaries, however, their error pattern does not appear to be a direct result of the input data or follow an obvious trend. One explanation may be that the contrast normalization performed prior to processing is affecting these curves. Based on our investigations, however, contrast normalization performed by wavelength as opposed to by image has little effect on the quality of 3D reconstruction.

The sum of coefficient diagrams, while an interesting byproduct of the algorithm, seem to offer little insight into the performance of our algorithms. For all three dictionaries the diagrams have only minute differences in image space coefficient placement and, as you might expect, they collect near the gradients and other prominent image features. Therefore they are unlikely to help diagnose systematic faults in our dictionaries.

While stricter coefficient placement enforcement from the repeated random dictionary caused a small increase in sparsity, we also see that it causes a reduction in PSNR in addition to increasing reconstruction time. Due to its high degree of parallelism, the 2D dictionary was by far the fastest implementation and in all three cases while also producing the highest quality reconstructions. There are many reasons why 3D
dictionaries may not yet be able to achieve higher reconstruction rates. Unlike the 2D dictionary, which can be easily trained on a large number of images, 3D dictionary learning is still heavily restricted by available memory. Furthermore, by having independent coefficients for each wavelength we are not truly exploiting the 3D structure of our data. Finally, by restricting all experiments to a MATLAB implementation we may be artificially restricting ourselves. In future implementations these are the key issues which must be addressed.

5.8 Conclusion

While 3D convolutional sparse coding is still in its infancy, it does show some promise. To achieve better results than 2D solutions, which will likely always be faster, the issues of memory consumption and dimensionality must be addressed.
Chapter 6

Future Applications

6.1 Introduction

The original intent of this work was to incorporate our results into upcoming hyperspectral projects. Although their progress has not yet reached the stage where image reconstruction results can be compared, here we will provide a brief overview of our first project, its image formation model, and how our results will be utilized.

This current project aims to reconstruct hyperspectral data from the single capture of a grayscale camera sensor. To alleviate the metamerism problem described in Chapter 2, we incorporate a simple diffractive element to the front of the camera lens system. In effect, this addition spreads incoming light according to its wavelength, allowing us to gather more information about its colour components. Although early designs aimed to reconstruct the hyperspectral data from this addition alone, simulations and preliminary experiments found there to be inadequate information to properly solve for high quality results. As such, the current design includes a binary mask placed near the sensor in addition to the diffractive element. Generally speaking, this mask should alleviate ambiguities in wavelength components, particularly in homogeneous image areas where the diffractive element alone would also result in homogeneous spread of light. Through the addition of the mask this spread of light can be found in homogeneous areas where the mask blocks this light as well.
6.2 Image Formation Model

Writing these factors into an image formation model we arrive at,

\[ y = PMBx \quad (6.1) \]

where \( x \) is the intrinsic image, \( y \) is our captured data, \( P \) is the projection matrix from the hyperspectral image space with 31 channels to our grayscale image space, \( M \) is the sheared binary mask pattern which partially attenuates incoming light, and \( B \) is a blur operator consisting of per-wavelength point spread functions due both the lens and transport from mask to sensor.

These factors lead us to a very difficult minimization problem so to further aid in reconstruction we can replace our captured data with a sparse representation of \( x \) using the techniques described in previous chapters. Using dictionary learning, for example, we could replace \( x \) with a previously learned dictionary \( \Psi \) and find updated coefficients \( \alpha \) during each iteration of our minimization.

This would lead to a minimization problem similar to,

\[ \min \|y - PMB\Psi\alpha\|_2^2 + \lambda\|\alpha\|_1 \quad (6.2) \]

where we again enforce sparse coefficients \( \alpha \) using the \( \ell_1 \) norm. This same model would similarly hold for other representations including TDL and CSC with minor changes to the formation model.

6.3 Image Reconstruction

Current reconstruction efforts are based on ADMM implementations of the image formation model described above. By exploiting a sparse basis we hope to validate results found by [5] wherein their reconstruction quality was directly correlated to the
sparsity of their dictionary representation.
Chapter 7

Concluding Remarks

7.1 Summary

We have thoroughly explored three different sparse representations of hyperspectral images, showing the parameters which are best able to exploit sparsity, and the computational trade-offs required for the highest quality reconstructions.

The 3D dictionary approach continues to be a viable choice for sparse reconstruction. When used with appropriate parameters it is a very straightforward method to find high quality reconstructions with a very sparse basis, although the time required to do is extreme in some cases.

Tensor Dictionary Learning had the highest reconstruction quality and, in most cases, was by far the most time efficient algorithm. Although its sparsity was inconsistent at higher patch sizes, when used with default parameters its basis sparsity was comparable to those of the 3D dictionary and CSC implementations.

Convolutional Sparse Coding for hyperspectral images shows great promise for the future. While our early implementations are restricted due to considerable memory requirements, we believe future work result in high quality reconstructions comparable to the patch based 3D dictionary approach, with less time complexity.

Finally, we have shown a common error in the way state of the art hyperspectral dictionaries are trained, introduce comparison methods to evaluate those errors, and
provide a simple solution improve reconstruction. When a wavelength correction is applied to the data we see a PSNR increase of, on average, 1.5dB with no other modifications to the code required.

7.2 Future Work

While 3D patch and tensor dictionaries have been thoroughly explored, there are many avenues for improvement in the domain of hyperspectral convolutional sparse coding that we hope to investigate. First and foremost, to explore the possibility of a 3D dictionary convolving with a 2D basis. Alleviating memory requirements, allowing for high quality dictionaries trained on higher resolution images to be trained, is another high priority problem that must be resolved.

The project in progress (discussed in Chapter 6), when fully formed, may also result in new found insight to this ongoing area of research. We look forward to continued progress of this project and its outcomes.
REFERENCES


