# Image Denoising Using Overcomplete Dictionary

Lei Lyu
*Electrical Engineering*
UNI: ll3433

***Abstract*—This article is about research on using dictionary learning to solve the image denoising problem. The main approach used here is K-SVD algrorithm, which will alternatively update coding and dictionary in the process. For the dictionary, this article used overcomplete dictionary, which starts from well-defined overcomplete DCT dictionary. As the dictionary is overcomplete, to make the problem well-posed the coding is assumed to be sparse, which is the sparse representation of image on given dictionary. Especially, for the coding calculation, proximal gradient descent method is adopted. Upon the above algorithm, this research did several experiments on setting of uniform Gaussian noise, non-uniform Gaussian noise and globally learned dictionary to find interesting facts and results on controlling the sparsity, changing the size of patch, changing learning strategy, etc in K-SVD algorithm.**

***Index Terms*—Image Denoising, Sparse Representation, Dictionary Learning, K-SVD, Overcomplete Dictionary, Proximal Gradient Descent**

## I. Introduction

Image Denoising is a very important task. Noise always exists in an image, but sometimes the magnitude of the noise is very small, or the signal-to-noise ratio is very high, so that it can't be observed by humans' eyes. However, if we want to do deeper analysis on images, especially in this era of using computer to solve the problem, the quality of each pixel in the image is essential to the final success in the result. Therefore, we'd like to remove the noise as much as possible without hurting the original structure in the image.

Sparse model and low-rank model are very interesting and popular methods to solve many image processing tasks in recent years [1]. Image denoising is one of those tasks that can be solved by sparse model. More specifically, dictionary learning is very popular in the image denoising task and many researches haven't been conducted in this field [2] [3].

Aharon and Elad [4] implemented K-SVD algorithm to address the image denoisng problem on image corrupted by white Gaussain noise, designed strategy to train the dictionary either on the noised image itself or other high-quality image and came up with ideas to force sparsity when coding each image patches. Dong etal. [5] deisgned a clustering-based sparse representation model to remove the noise from image and applied re-weighting and regularizing techniques in the process. Beckouche etal. [6] applied centered dictionary learning method to solve image denoising problem in the astronomical images and gained better results than traditional methods in the field of image denoising in astronomy.

Besides denoising task, other two similar and equally important tasks, image deblurring and image super-resolution, also deserve attentions. There are also many developed sparse models [7] and dictionary learning methods [8] [9] on solving these two tasks.

## II. Technical Approach

### A. K-Means and Expectation Maximization Algorithm

Before introducing K-SVD algorithm which is the main technique used in this research, it is proper to first introduce a other two similar techniques, K-Means algorithm, which is commonly used in machine learning on clustering given several vectors, and Expectation Maximization, which is the support in a probabilistic view on the convergence of such an alternative learning strategy.

*1) K-Means:* K-Means algorithm [10] is generally aiming at solving the clustering problem given vectors $\{v_1, v_2, \cdot, v_n\}$ in $\mathbb{R}^d$.

The first step is to set the number of the clusters $k$ to find, and then randomly initialize $k$ centroids $c_i \in \mathbb{R}^d$, which can be regarded as the center of vectors to the cluster. Then assign vector $d_j$ to the nearest centroid $c_i$ in terms of Euclidean distance $\operatorname{argmin}_{c_i} \|c_i - d_j\|_2$. After assigning all the vectors to one specific cluster, recalculate the centroid of cluster $c_i$ using the mean of all vectors belonging to that cluster. Do the above reassigning and re-centering steps alternatively for many rounds until convergence, and the vectors can be efficiently clustered into $k$ ones.

*2) Expectation Maximization:* Another algorithm Expectation Maximization [11] can give some intuition in a probabilistic view on how the above alternative algorithm as well as K-SVD algorithm converges to a correct result.

The given condition of Expectation Maximization is that we want to estimate some target $p_1$ with known parameters $p$ and unknown parameters $p_2$. To estimate the value of $p_1$, we'd like to maximize the likelihood function $f(p|p_1)$, which is the probability of $p$ given $p_1$. Often, we will maximize $\ln f(p|p_1)$. As sometimes it is hard to directly model the probability of $p$ from $p_1$. For example, the above problem, estimating the centroids of clusters using only vectors is not possible and we introduce some unknown parameters, the possible cluster of each vector to help with finding the specific location of each cluster. The unknown assignment to each cluster is just one specific example of the unknown parameters $p_2$ that we'd like to use in Expectation Maximization algorithm. With the introduction of unknown parameter $p_2$, we can rewrite $\ln f(p_1)$ as:

$$\int f(p_2) \ln \frac{f(p, p_2|p_1)}{f(p_2)} dp_2 + \int f(p_2) \ln \frac{f(p_2)}{f(p_2|p, p_1)} dp_2 \quad (1)$$

, which can be verified by adding the two terms up and eliminating $p_2$. Then the alternative steps come as the expectation step to first set $f(p_2) = f(p_2|p, p_1)$ to estimate $p_2$ or its distribution with given $p$ and $p_1$ and reduce the Equation 1 to:

$$\int f(p_2)[\ln f(p, p_2|p_1) - \ln f(p_2)]dp_2 \tag{2}$$

, and the maximization step to maximize $p_1$ over Equation 2 to update new estimated values for $p_1$.

With the above alternative steps, it can be proved that the value of $\ln(p_1)$ keeps increasing.

First, in the expectation step, no matter what $f(p_2)$ we choose, the value of $\ln f(p|p_1)$ stays unchanged, since the relation $\ln(f(p|p_1) =$ Equation 1 always hold for any choice of $f(p_2)$. Then in the maximization step, we choose a new value for $p_1$ so that it increases $\ln(p|p_1)$. Therefore, at the end of each iteration, $\ln(p|p_1)$ is larger than the previous iteration.

## *B. K-SVD Algorithm*

K-SVD algorithm [10] is very similar to previously described K-Means and EM Algorithm, in terms that they all solve problem with two unknown variables in an alternative way.

The first step is to initialize an overcomplete dictionary, which can be generated randomly. However, usually, to achieve faster computation speed and better performance, it is better to start from those well-defined dictionary, like DCT dictionary, wavelet transform dictionary, etc [4] [12].

Before the second step, an additional step is to crop the noised images into patches. The detail of this part will be shown in the next section.

The second step is conducting sparse coding and dictionary updating steps alternatively until the algorithm converge.

*1) Sparse Coding:* The paper [4] mainly used OMD and FOCUSS [13] pursuit algorithm to calculate the coding. Here, I implemented proximal gradient descent method to solve the problem. For the sparse coding part, it needs to calculate coding for each patch gained in the second step. Also, we assume the dictionary to be fixed in this step. Furthermore, for the $l_0$ norm, it can be efficiently solved by $l_1$ norm. So in the end, the objective function in this part become:

$$\min_{A_i^{(t+1)}} \frac{1}{2}\|Y_i - D^{(t)}A_i^{(t+1)}\|_2^2 + \mu\|A_i^{(t+1)}\|_1 \tag{3}$$

, where $Y_i$ is the *i-th* patch in the noised image, $A_i^{(t+1)}$ is the coding for *i-th* patch in the current iteration and $D^{(t)}$ is the dictionary learned from the previous iteration.

Then the sparse coding $A_i^{(t+1)}$ can be solved by proximal gradient descent method. First, calculate the gradient of $A_i^{(t+1)}$ with respect to smooth part in the objective function as:

$$\nabla_{A_i^{(t+1)}} f_i^{(t+1)} = (D^T)^{(t)}(D^{(t)}A_i^{(t+1)} - Y_i) \tag{4}$$

. Then calculate the intermediate estimate as:

$$B_i^{(t+1)} = A_i^{(t+1)} - \frac{1}{\|D^{(t)}\|^2} \cdot \nabla_{A_i^{(t+1)}} f_i^{(t+1)} \tag{5}$$

. Finally, calculate the new estimate of $A_i^{(t+1)}$ in the current iteration as:

$$\underset{A_i^{(t+1)}}{\text{argmin}}\{\frac{\mu}{\|D^{(t)}\|^2}\|A_i^{(t+1)}\|_1 + \frac{1}{2}\|A_i^{(t+1)} - B_i^{(t+1)}\|_2^2\} \tag{6}$$

, for which the solution is to shrink all entries' magnitudes by $\frac{\mu}{\|D^{(t)}\|^2}$, and for entries less than that threshold value, set them to zero. In that way, the sparsity is enforced.

Repeat the above proximal gradient descent steps for enough iterations until convergence. We can gain a sparse enough result $A_i^{(t+1)}$ at the end of computation.

*2) Dictionary Updating:* After calculating the sparse coding on $D^{(t)}$, it comes to use that result to update a new value of dictionary $D^{(t+1)}$. This step is done iteratively from the first column to the final column in the dictionary. Let's denote the *k-th* column of the dictionary $D^{(t)}$ by $d_k^{(t)}$.

The first step is to find all the patches that have used $d_k^{(t)}$. Or in other words, the *k-th* entry in the coding of the patch is nonzero. Then we can denote the set of such patches as $\omega_k^{(t+1)} = \{i|A_i^{(t+1)}(k) \neq 0\}$.

Then $\forall j \in \omega_k^{(t+1)}$, we will calculate the error without using $d_k$, $e_j^{(t+1)} = Y_j - \sum\limits_{x<k} d_x^{(t+1)}A_j^{(t+1)}(x) - \sum\limits_{y>k} d_y^{(t)}A_j^{(t+1)}(y)$. Notice that, the reason why we only use patches that appear in $\omega_k^{(t+1)}$ is because we don't want to break the sparsity built in sparse coding steps.

After calculating all $e_j^{(t+1)}$ above, treating them as columns in a matrix, we can build the error matrix $E_k^{(t+1)}$

Next, calculate the SVD(Singular Value Decomposition) on $E_k^{(t+1)}$ as:

$$E_k^{(t+1)} = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{7}$$

Finally, update $d_k^{(t+1)} = u_1$, which is the first column of matrix $U$. Also, update nonzero entry in $A_i^{(t+1)}(k)$ for all the patches as $\sigma_1 \cdot v_i^T$, which is the largest singular value times the first row of $V^T$. After all the steps above, go to the next column until the last one.

## *C. Convergence and Correctness of K-SVD Algorithm*

In this part, I will briefly mention the convergence and correctness of K-SVD algorithm.

*1) Convergence:* : K-SVD algorithm is very similar to K-Means [10] and EM [11] algorithm that have been introduced in details in the very beginning. Then from those two convergent alternative algorithm, we can gain some intuitions how K-SVD might also converge.

First, the two unknown variables are dictionary and coding, which can correspond to centroid and cluster number in K-Means Algorithm or $p_1$ and $p_2$ in EM Algorithm.

The sparse coding part is similar to estimating the cluster number of each vector given the locations of centroids. And it is also similar to expectation step in EM algorithm. This is not straightforward, as EM algorithm is in probabilistic view. $f(p_2)$ in this deterministic setting, can be regarded as how we find the coding. Also, we can interpret $f(p_2|p, p_1)$ as given noised image patches and a fixed dictionary, find coding that reach a balance between sparsity and the error. Therefore, sparse coding can also be somehow explained as expectation step.

The dictionary updating step is like recalculating the centroids for each cluster. But there is an obvious difference that all the centroids are calculated at the same time while the columns of dictionary are updated iteratively. It will be shown in the correctness part that this iterative update still manage to reduce the error. So that it is also very similar to the maximization step in EM algorithm, in which we are saying finding $p_1$ that maximize the likelihood $f(p|p_1)$. In some sense, maximizing the likelihood is equivalent to minimizing the error. Therefore, we can see there are similarities between the maximization step in EM algorithm and dictionary updating steps in K-SVD algorithm as well.

Although the above analysis can't come as a formal proof of convergence of K-SVD algorithm. It indeed gives some intuition on the similarities of k-SVD algorithm between K-Means and EM algorithm. For the future work, one can go deeper into their relations to give a more reasonable proof on the convergence.

*2) Correctness:* The correctness of sparse coding part just follows the correctness of proximal gradient descent, which can guarantee to produce a sparse solution if the value of $\mu$ in Equation 3 is chosen appropriately.

Then for the dictionary updating step, its correctness can be proved by showing that in each step, the overall error between estimate and noised image will be reduced.

First, notice that $E_k^{(t+1)}$ is the total error matrix (for patches in $\omega_k^{(t+1)}$) without using $d_k^{(t)}$. Then the square error of $e_j^{(t+1)}$, $\forall j \in \omega_k^{(t+1)}$ is just the Frobenius norm of $E_k^{(t+1)}$. Therefore, what we are going to do here is to find a new $d_k^{(t+1)}$ that can minimize the error here:

$$\min_{d_k^{(t+1)}, (\alpha_k^T)^{(t+1)}} \|E_k^{(t+1)} - d_k^{(t+1)}(\alpha_k^T)^{(t+1)}\|_F^2 \tag{8}$$

. $d_k^{(t+1)}(\alpha_k^T)^{(t+1)}$ is a rank-1 matrix and we know that the best rank-1 matrix approximation on $E_k^{(t+1)}$ is the first singular value decomposition term $\sigma_1 u_1 v_1^T$. Thus the problem becomes solving:

$$d_k^{(t+1)}(\alpha_k^T)^{(t+1)} = \sigma_1 u_1 v_1^T \tag{9}$$

. One more thing needs to be notice here is, for any given $d_k^{(t+1)}$, one can easily compute the corresponding $(\alpha_k^T)^{(t+1)}$ that minimize the Equation 8 by solving its gradient with respect to $(\alpha_k^T)^{(t+1)}$ equal to 0:

$$(d_k^T)^{(t+1)}[d_k^{(t+1)}(\alpha_k^T)^{(t+1)} - E_k^{(t+1)}] = 0 \tag{10}$$

. Thus an analytical solution can be gained here as:

$$(\alpha_k^T)^{(t+1)} = [(d_k^T)^{(t+1)} d_k^{(t+1)}]^{-1} (d_k^T)^{(t+1)} E_k^{(t+1)} \tag{11}$$

. Plug Equation 11 into Equation 9, we can find the new dictionary as:

$$d_k^{(t+1)}[(d_k^T)^{(t+1)} d_k^{(t+1)}]^{-1} (d_k^T)^{(t+1)} E_k^{(t+1)} = \sigma_1 u_1 v_1^T \tag{12}$$

. It is easy to verify that $d_k^{(t+1)} = u_1$ is a feasible solution to Equation 12 as:

$$\begin{aligned} u_1(u_1^T u_1)^{-1} u_1^T E_k^{(t+1)} &= u_1(u_1^T u_1)^{-1} u_1^T \sum_{i=1}^{r} \sigma_i u_i v_i^T \\ &= u_1(u_1^T u_1)^{-1} u_1^T \sigma_1 u_1 v_1^T \\ &= \sigma_1 u_1 v_1^T \end{aligned} \tag{13}$$

, where the second step is gained because $u_i$ is orthogonal to each other. Thus, it can be proved that the new column in the dictionary manage to make the error smaller than before, as it constructs the best rank-1 matrix approximation to $E_k^{(t+1)}$, so the following relation must hold:

$$\|E_k^{(t+1)} - d_k^{(t)}(\alpha_k^T)^{(t)}\|_F^2 \geq \|E_k^{(t+1)} - d_k^{(t+1)}(\alpha_k^T)^{(t+1)}\|_F^2 \tag{14}$$

, because $d_k^{(t)}(\alpha_k^T)^{(t)}$ is an arbitrary rank-1 matrix, where $d_k^{(t)}$ is the *k-th* column in dictionary before updating and $(\alpha_k^T)^{(t)}$ is the nonzero coding calculated for $d_k^{(t)}$ in sparse coding step.

### *D. Other Details*

*1) Overcomplete DCT Dictionary:* The way I build those overcomplete DCT (Discrete-Cosine-Transform) dictionary in 2-D is to start from 1-D DCT vectors. For example with 8 DCT-vectors of dimension 8, we can take convolution between each of them and thus gain 64 2-D DCT basis of size $8 \times 8$, which will later be flattened into one column in the dictionary. This is a complete DCT dictionary. To make the dictionary overcomplete, we need more 2-D DCT basis. Therefore, we can either modify 1-D DCT basis or 2-D DCT basis to enlarge the whole dictionary. There are so many ways to do so, in my implementation of overcomplete DCT dictionary of size $64 \times 128$, I try both enlarging ways.

*2) Cropping Image into Patches:* There are several things needed to be mentioned here on how to crop the image into patches. For each image, I will cut it into patches of the same size, which is either $8 \times 8$ or $12 \times 12$ in my implementation. Between adjacent patches, there will always be some overlapping region, or can be interpreted as stride. For this implementation, I use stride of $4$ for patch size $8 \times 8$ and use stride of 6 for patch size $12 \times 12$.

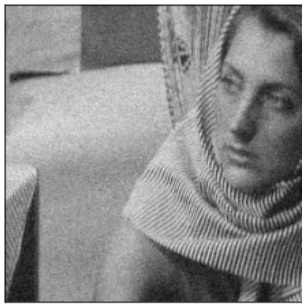
*3) Reconstructing Image from Patches:* Since the reconstruction is conducted on each patch in the implementation, then at the end I will combine these patches altogether into a whole image. There are many ways to do so. For example, we can discard the redundant edge between patches, so that the remaining part can exactly make a whole image. Or we can just put each patch on its original location in the image and count how many times each pixel has been used. At the end, take average on each pixel with respect to the counting times.

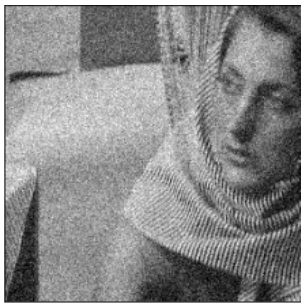
## III. Experiments and Discussion

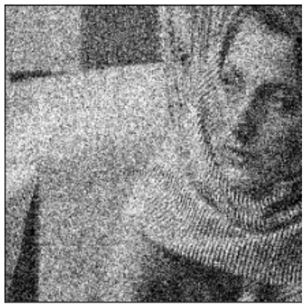
### *A. Image Denoising on Uniform Gaussian Noise*

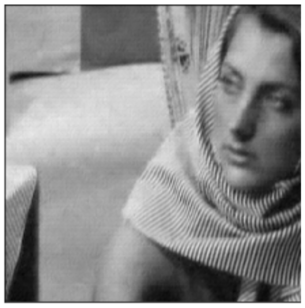
First, I evaluate how well using K-SVD algorithm performs on reconstructing image from image corrupted by uniform Gaussian noise.

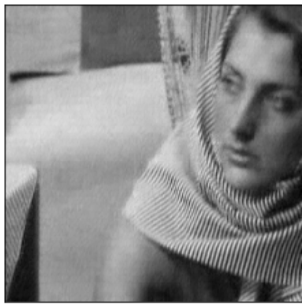
(a) Original Image

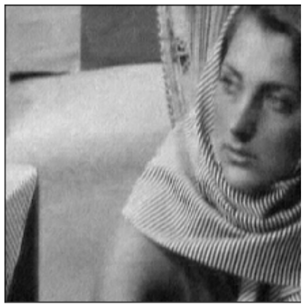

(b) Gaussian noise $\sigma = 0.05$

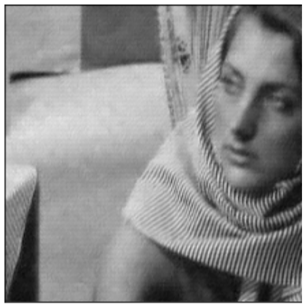

(c) Gaussian noise $\sigma = 0.1$

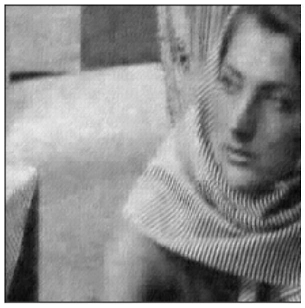

(d) Gaussian noise $\sigma = 0.2$

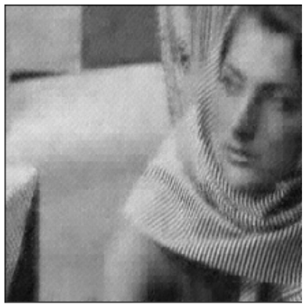

(e) $\sigma = 0.05$, $8 \times 8$ patch

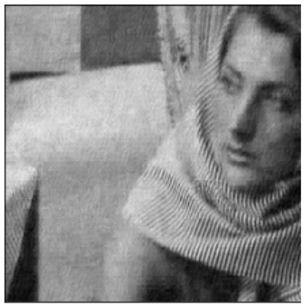

(f) $\sigma = 0.05$, $12 \times 12$ patch

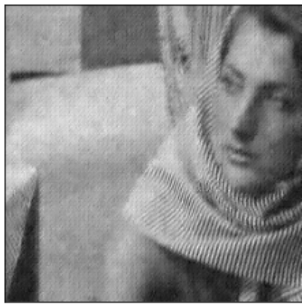

(g) $\sigma = 0.05$, $8 \times 8$ patch with sparsity control

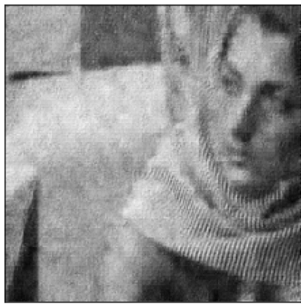

(h) $\sigma = 0.05$, $8 \times 8$ patch with edge-discarding reconstruction

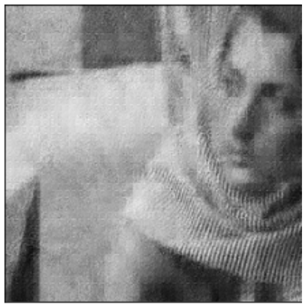

(i) $\sigma = 0.1$, $8 \times 8$ patch

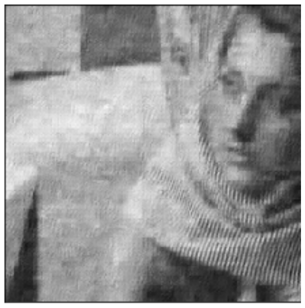

(j) $\sigma = 0.1$, $12 \times 12$ patch


(k) $\sigma = 0.1$, $8 \times 8$ patch with sparsity control


(l) $\sigma = 0.1$, $8 \times 8$ patch with edge-discarding reconstruction


(m) $\sigma = 0.2$, $8 \times 8$ patch


(n) $\sigma = 0.2$, $12 \times 12$ patch


(o) $\sigma = 0.2$, $8 \times 8$ patch with sparsity control

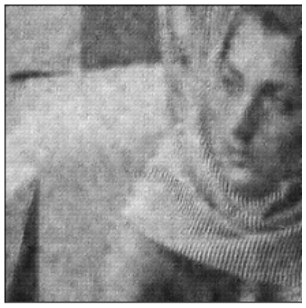

(p) $\sigma = 0.2$, $8 \times 8$ patch with edge-discarding reconstruction

Fig. 1. Experiment Results on Uniform Gaussian Noise

Generally, Gaussian noise on each pixel is assumed to be independent and identically distributed Gaussian variable with zero mean and variation of $\sigma$, where $\sigma$ is the magnitude that controls how serious the noise is. The visualization of the noised image under different choices of $\sigma$ is shown from Fig. 1(b) to 1(d). The *PSNR* and *SSIM* values are listed in Table I.

TABLE I
PSNR AND SSIM WITH UNIFORM AND NOUNIFORM GAUSSIAN NOISE

| | $\sigma = 0.05$ | $\sigma = 0.1$ | $\sigma = 0.2$ |
|---|---|---|---|
| PSNR | 26.03 | 20.03 | 14.49 |
| SSIM | 0.891 | 0.712 | 0.443 |
| | random $\sigma$ | larger $\sigma$ on left | larger $\sigma$ on right |
| PSNR | 19.02 | 19.84 | 20.35 |
| SSIM | 0.666 | 0.713 | 0.767 |

To make the computation and experiments more efficient, I only choose a small part of the original image. However, the portion I cropped indeed contains important regions that is interesting to concentrate on, like the ground that is smooth over a large area, the scarf that has rapidly changed stripes and face that is combining smooth area on the cheek and sharp edge near the nose.

Then, starting from the whole image, the first step is to choose the size of patches and stride between each patches. To explore on whether the size of patches have significant influence on the result, I choose two different cropping schemes. The first one is using patch of size $8 \times 8$ with stride of 4 pixels between patches and the other is using patch of size $12 \times 12$ with stride of 6 pixels between each patch.

TABLE II
PSNR AND SSIM RESULTS AFTER REMOVING UNIFORM GAUSSIAN NOISE

| $\sigma = 0.05$ | $8 \times 8$ | $12 \times 12$ | $8 \times 8$ + control | $8 \times 8$ + edge discard |
|---|---|---|---|---|
| PSNR | 30.80 | 31.13 | 31.22 | 30.18 |
| SSIM | 0.870 | 0.881 | 0.863 | 0.846 |
| $\sigma = 0.1$ | $8 \times 8$ | $12 \times 12$ | $8 \times 8$ + control | $8 \times 8$ + edge discard |
| PSNR | 26.36 | 26.37 | 26.64 | 24.58 |
| SSIM | 0.737 | 0.737 | 0.711 | 0.600 |
| $\sigma = 0.2$ | $8 \times 8$ | $12 \times 12$ | $8 \times 8$ + control | $8 \times 8$ + edge discard |
| PSNR | 22.19 | 22.63 | 23.65 | 21.46 |
| SSIM | 0.526 | 0.514 | 0.560 | 0.417 |

The image results are shown in the first two columns in Fig. 1, while the *PSNR* and *SSIM* reuslts are shown in Table II. Those results show that using a larger patch size, the noise is better removed, but the details of the image are reconstructed not as well as using a smaller patch size. In terms of *PSNR* and *SSIM*, there is no significant advantage of using larger patch over smaller patch. The reason I guess for this to happen is because as shown in Equation 7, the singular value decomposition on the error matrix has $r$ terms, which is mainly constrained by the size of patch as there are far more patches here(for example when the size of patch is $8 \times 8 = 64$, there are in total 3864 patches gained). Therefore, when there are more terms, the error reduced by approximating error matrix by the first singular value decomposition will decrease. However, as there are more columns of using larger size of patch, then there will be more error eliminating times for larger patch size. Under this trade-off, there is no significant improvement on *PSNR* and *SSIM* under two sizes of patch. Another interesting observation made in this part is for patch size of $8 \times 8$ and $12 \times 12$, the good $\mu$ value is very closed.

Next, as in the implementation, I used proximal gradient descent to calculate the sparse coding of dictionaries, which is very sensitive to a good choice on the value of $\mu$. So, beyond the ordinary strategy that fix value of $\mu$ in the whole process, I also come up with another strategy to control the value of $\mu$ in the procedure to ensure the sparsity is neither too small or too large. So, there are two schemes I design for this part. The first is to learn a good value of $\mu$ from the pre-computed DCT dictionary and use that value in the whole process. Another scheme is to use adaptive values of $\mu$, which can be changed at the sparse coding step until reaching a reasonable range of sparsity on coding. The range can also be roughly estimated on overcomplete DCT dictionary. This sparsity controlling scheme is inspired from the EM algorithm, in which the value of $\ln f(p|p_1)$ doesn't change in expectation step. Corresponds to the expectation step in EM algorithm is the sparse coding part. Notice that in the dictionary updating step, the sparsity of the coding is kept, as only nonzero entries get updated. Therefore, in the next sparse coding part, if we keep the sparsity in a reasonable range, to decrease the loss function, the only way is to make the square error $\|Y_i - D^{(t)} A_i^{(t+1)}\|_2^2$ small. However, in practice, it is still possible for error to increase a little bit, as I make constraint on a range on the values of sparsity instead of a deterministic value and $l_1$ norm is used to approximate results of $l_0$ norm, for which there is no guarantee that the same sparsity provides the same value of $l_1$ norm on coding. Therefore, the error of estimate still fluctuate in the sparse coding steps, instead of keeping decreasing, but the extent is smaller than without controlling the sparsity.

The results of image reconstruction quality and *PSNR* and *SSIM* values are shown in Fig. 1 and Table II(column $8 \times 8$ + control stands for controlling $\mu$ stategy mentioned above). In terms of image quality, with control on the sparsity, the reconstruction on details of image is further improve while removing noise has no improved performance. With controlling sparsity, the values of *PSNR* increase while the *SSIM* values drop a little bit, except for very large noise *SSIM* is also higher with control. One interesting thing found during the experiments is with controlling on the sparsity, the algorithm tends to take more iterations to converge but the values of *PSNR* in the whole process is more steady than using a fixed value of $\mu$ in the whole process. The reason for this I think is because as the dictionary changes in the updating, the same value of $\mu$ will produce different sparsity on the coding. Either too large or too small sparsity will hurt the result, so that the performance is not steady if without controlling the sparsity during the process. Also, the good choice of sparsity tends to have no significant deviation on the change of dictionary.

However, one more thing needs notice is that the magnitude of Gaussian noise indeed has influence on choice of $\mu$ and sparsity. When the value of $\sigma$ in Gaussian noise becomes larger, the larger $\mu$ or sparser results are expected to get better performance on the reconstruction. So before implementing K-SVD algorithm, it is better to first test on overcomplete DCT dictionary or other available well-built dictionaries to estimate what values of $\mu$ and sparsity provide good performance.

Finally, after finding the dictionary from corrupted patches and calculating the sparse coding of that dictionary for each patches, we need to combine these patches together to form a whole image. Still, I have two schemes on this combination task. The first trial is to directly put patches into their original places in the original image and later on take average on how many each pixel has been used. The second method is discarding the overlapping regions of each patch, so that the remaining part can exactly form a whole image.

Again, the results are shown in Fig. 1 and Table II(column $8 \times 8$ + edge discard stands for the second patch combination strategy mentioned above). The result is very obvious in terms of image quality and *PSNR* and *SSIM* values, the first strategy of averaging over how many pixels are used in all patches does a better job here.

## *B. Image Denoising on Nonuniform Gaussian Noise*

Upon experiments in the first part, a good choice of experiment setting can be decided on using patch size $8 \times 8$ and patch combination strategy of putting patches into original places and averaging each pixel on how many times it has been used in patches. Then we can use this setting on trying to solve image with nonuniform Gaussian noise. There are generally two types of the nonuniform noise. The first is shown in Fig. 2(b), in which the value of $\sigma$ for noise on each pixel is uniformly distributed on the range $(0, 0.2]$. The second is shown in Fig. 3(b) and 4(b), where the value of $\sigma$ increases or decreases linearly between the range [0.25, 1.75] from left to right on the image. The values of *PSNR* and *SSIM* after adding Gaussian noise in those ways are listed in Table I.

TABLE III
PSNR AND SSIM RESULTS ON NONUNIFORM GAUSSIAN NOISE

| random $\sigma$ | with noise | $8 \times 8$ | $8 \times 8$ + sparsity control |
|---|---|---|---|
| PSNR | 19.02 | 25.86 | 26.21 |
| SSIM | 0.666 | 0.710 | 0.696 |
| Larger $\sigma$ on left | with noise | $8 \times 8$ | $8 \times 8$ + sparsity control |
| PSNR | 19.84 | 25.66 | 25.94 |
| SSIM | 0.713 | 0.675 | 0.678 |
| Larger $\sigma$ on right | with noise | $8 \times 8$ | $8 \times 8$ + sparsity control |
| PSNR | 20.35 | 25.98 | 26.05 |
| SSIM | 0.767 | 0.753 | 0.762 |

First, for random magnitude of Gaussian noise, as shown in Table III, *PSNR* is higher with sparsity control, while *SSIM* drops under sparsity control. More important thing than the comparison above is the image reconstruction quality shown in Fig. 2. Those reconstruction image still looks good, but the quality is not as good as removing uniform Gaussian noise. Some parts on the face are more blurred and noise is still obvious in some background areas. The reason for this to happen I think is because different magnitude of noise requires different sparsity to get a good balance between removing the noise and reconstructing the details to produce an image of high visual quality. However, as the noise magnitude differs in each patch, it is hard to determine what sparsity is good for each patch. Then using an average sparsity is just an alternative to make the image good in general, but it can't make everywhere perfect. This reason can also be used to explain the failure in the following two experiments.

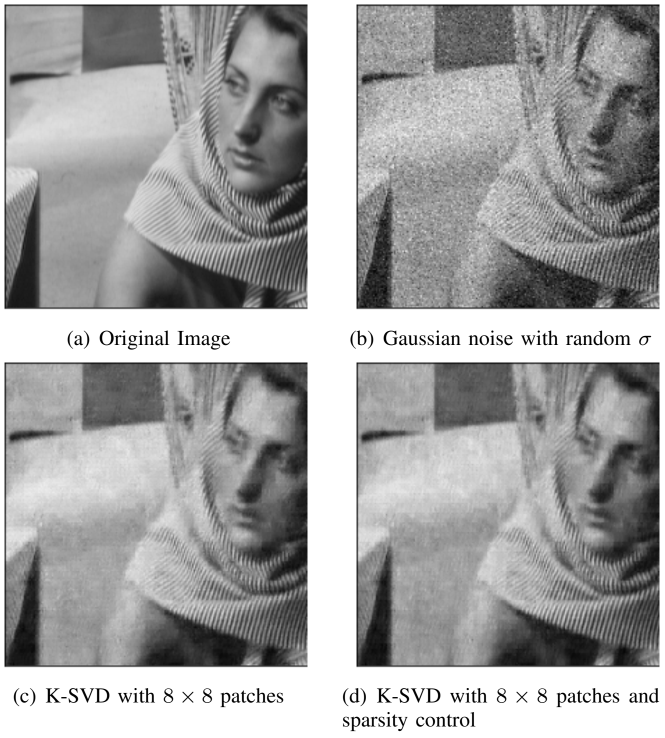

(a) Original Image
(b) Gaussian noise with random $\sigma$
(c) K-SVD with $8 \times 8$ patches
(d) K-SVD with $8 \times 8$ patches and sparsity control

Fig. 2. Image Reconstruction on image with random magnitude of Gaussian noise

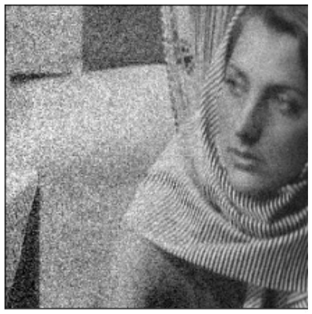
In the setting where larger noise is imposed on the left-hand side of the image, again the index *PSNR* and *SSIM* in Table III shows an acceptable reconstructing quality, with a slightly better performance with sparsity control. When we look at the reconstructed images in Fig. 3, the failure is more obvious than results in random magnitude part. The reconstruction on face is very blurred, the left side still has significant noise remained and only the middle part has good visual quality. Again, the reason is because removing different magnitude noise needs different sparsity to perform well.

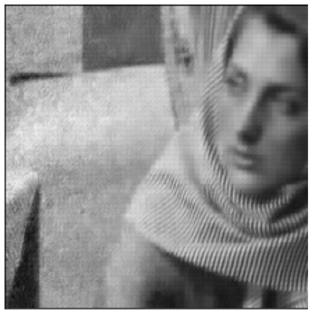
The same thing happens when the face is imposed a larger noise. From Fig. 4, we can observe that the details on the face are reconstructed well but the noise is still significant. For the performance on *PSNR* and *SSIM* index, the performance of the algorithm is still very good in Table III.

## *C. Image Denoising with Globally Learned Dictionary*

One issue that learning dictionary on given corrupted image has is that, for each new image, we might want to learn a new

(a) Original Image


(b) Gaussian noise, larger $\sigma$ on left


(c) K-SVD with $8 \times 8$ patches

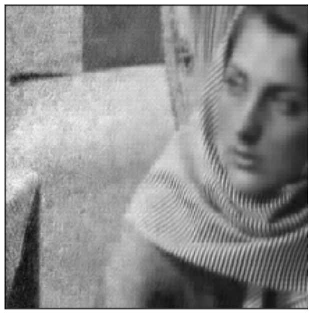
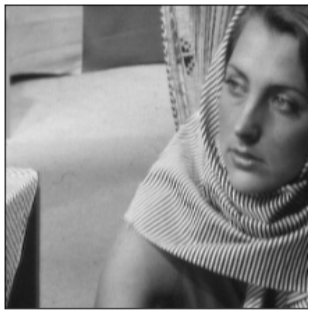
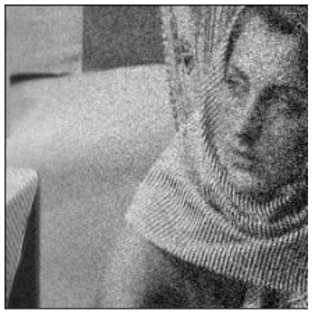
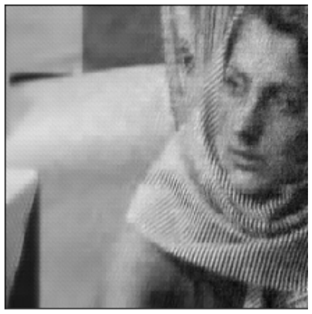

(d) K-SVD with $8 \times 8$ patches and sparsity control

Fig. 3. Image Reconstruction on Image with larger magnitude of Gaussian noise on the left


(a) Original Image


(b) Gaussian noise, larger $\sigma$ on right


(c) K-SVD with $8 \times 8$ patches

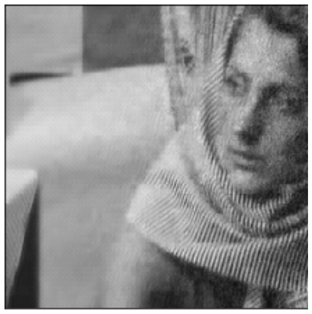
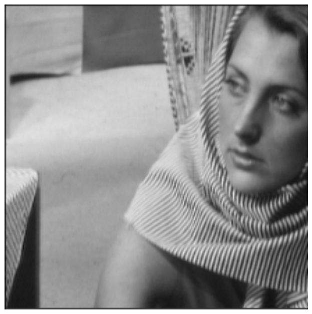
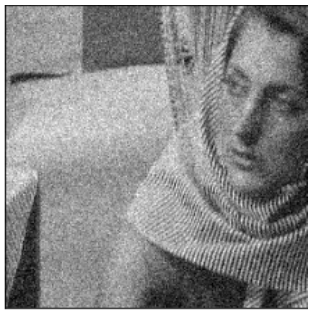
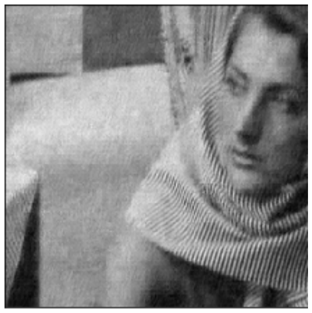

(d) K-SVD with $8 \times 8$ patches and sparsity control

Fig. 4. Image Reconstruction on Image with larger magnitude of Gaussian noise on the right


(a) Original Image


(b) Uniform Gaussian noise $\sigma = 0.1$


(c) K-SVD with dictionary learned from one image

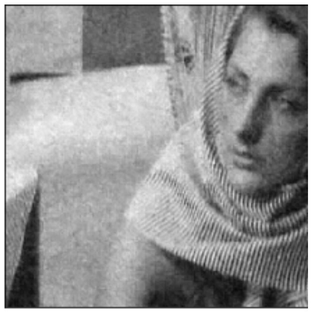

(d) K-SVD with dictionary learned globally

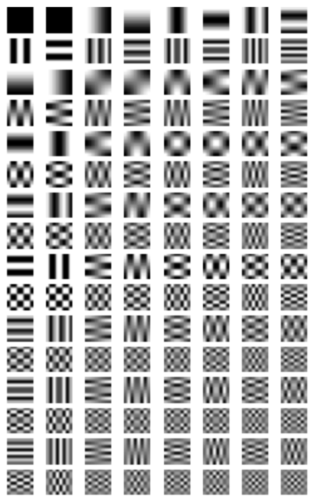
Fig. 5. Image Reconstruction on Image with dictionary learned from noised image versus dictionary learned from other high-quality images

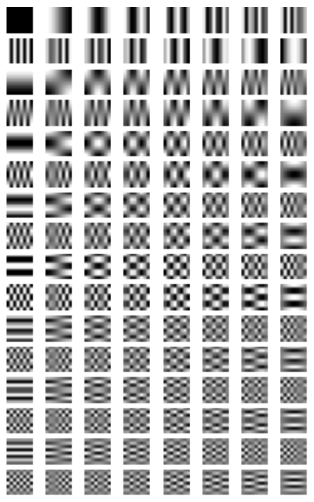
dictionary, which is extremely time consuming if we have a lot of images to deal with.

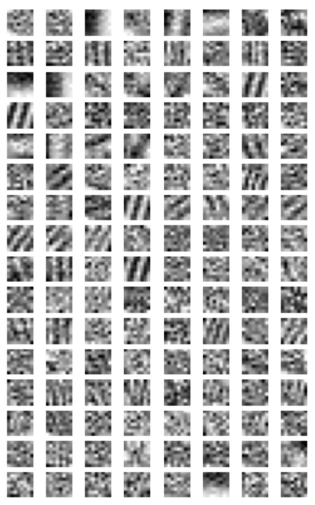
To solve this issue, one may learn a dictionary from a bunch of images to build a globally learned dictionary. In the former part, the dictionary sizes used are $64 \times 128$ and $144 \times 244$ respectively. Then in this globally learned dictionary, I build a dictionary of size $64 \times 128$ to make comparisons between the dictionary of the same size but built with different methods before. To be more specific, I choose 91 high quality images and crop them into patches. Because of the limited computation power, for each of them, I only choose 80 samples. Then with those patches, the training procedure is the same as training on the noised image.

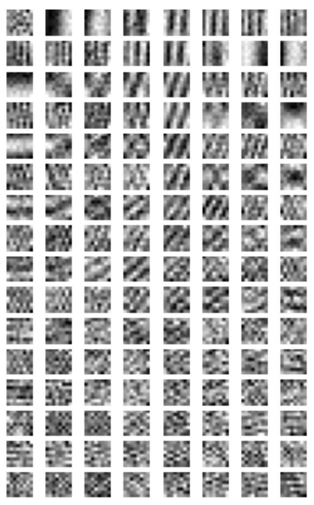
The reconstruction results using globally trained dictionary are show in Fig. 5(d). Compared with reconstructed image from dictionary learned from the noised image itself in 5(c), the details reproduction is better but the noising removing is worse. The reason for this, as will be shown in Fig. 6 in later section, the dictionary learned from noised image itself learns very specific edge patterns, while the globally built dictionary learned more general patterns. So that the noise can also somehow be estimated by those general patterns, which leads to more noise remained in the image. So it remains uncertain whether increase the size of globally learned dictionary can actually gain better performance, as can be seen in the reconstructed image that the learned pattern is already enough to represent good details.

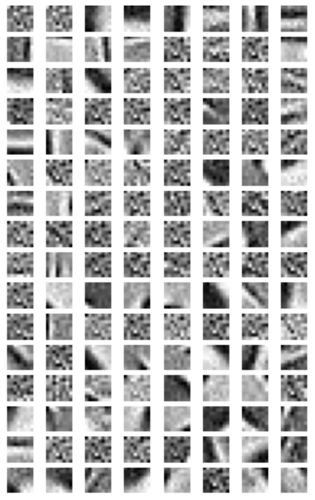
For the index performance, globally learned dictionary gains


(a) Overcomplete DCT dictionary 1


(b) Overcomplete DCT dictionary 2


(c) Specific dictionary learned overcomplete DCT dictionary 1


(d) Specific dictionary learned overcomplete DCT dictionary 2


(e) Global dictionary learned overcomplete DCT dictionary 1

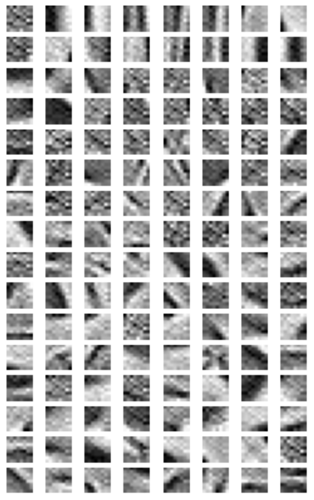

(f) Global dictionary learned overcomplete DCT dictionary 2

Fig. 6. Dictionary Visualization on different starting overcomplete dictionaries and different learning strategies

$PSNR = 26.36$ and $SSIM = 0.737$, while specifically learned dictionary gains $PSNR = 26.32$ and $SSIM = 0.702$. Their performance is competitive in terms of those values.

### *D. Visualization of Learned Dictionary*

To better understand how initial overcomplete dictionary and the learned dictionary differs and also what the dictionary actually learns, it is intuitive to directly visualize those dictionaries.

As shown in Fig. 6, the dictionaries learned from noised image itself and learned globally have obvious difference in the patterns. In the specifically learned dictionary, there are many high-frequency edge-like patterns in different orientation. However, in globally learned dictionary, the learned patterns are much simpler, like those patches that are half white and half black, and they can be interpreted as more generally learned property. Under those two different patterns inside dictionaries, those two dictionaries tend to be good at conducting different jobs. As has been shown in the former section, for dictionary learned from noised image itself, the noise is removed better, as the dictionary is good at reconstructing particular structures in that specific image. At the same time, since the learned patterns in globally learned dictionary are simpler and more general, so it can model the noise in some extent as well. Therefore, more details are reproduced from the noised image, regardless of whether it is noise or not.

Another interesting things can be observed here is no matter which initialized overcomplete DCT dictionary we start from here, the learned patterns dictionary are very consistent under particular learning schemes. This somehow shows that the algorithm is not so dependent on a very good initialization on the dictionary to gain an acceptable performance. The good initialization might matter more on saving time and computational power in the whole process to reduce the exploration time using randomly generated dictionary.

## IV. Conclusion

This part acts as a summary to the research conducted above, as the more detailed analysis and discussion has been made besides the experimental results in the last section.

This article starts with introduction on K-SVD algorithm, as well as some qualitative and quantitative analysis on K-SVD algorithm to verify its correctness and convergence to be used to solve the overcomplete dictionary learning problem. Beyond those theoretical results, a bunch of experiments are conducted to analyze the practical power of this algorithm to remove noise in the image. First, the experiments are conducted on image corrupted with uniform Gaussian noise. Experiments are designed to investigate on the influence of patch size, sparsity controlling scheme and patches recombination methods on the final quality of the noise-removed image. It is found that the patch size has no significant effect on the final results, for which the reason might be the trade-off during dictionary updating steps between amount of error eliminated each time and number of error eliminating times. Also, the experiments show that with controlling on the sparsity, the image reconstruction tends to be better in details reproducing and a little bit worse at noise removing. Also, in those experiments, it is found that larger magnitude of noise requires smaller sparsity in the coding to gain best results. Finally, the experiments on different ways to combine patches show that it is better to use the strategy of putting patches into original locations and averaging each pixel over the times it has been used in patches. The second experiment is on nonuniform Gaussian noise applied on the image. There are three settings, random magnitude, larger magnitude on the left and larger magnitude on the right. The results in terms of *PSNR* and *SSIM* index are as good as removing uniform Gaussian noise. However, in terms of image quality, K-SVD algorithm implemented here is somehow fooled by nonuniform patterns of noise, with part of image not being denoised enough and other part getting blurred. The reason for this to happen is the finding in the former experiments that different magnitudes of noise prefer different sparsity on the coding. The final experiment is about learning dictionary also from many high quality images. The result in this part shows that *PSNR* and *SSIM* results tend to have no significant difference between dictionary learned from noised image itself, while the globally learned dictionary is better at reproducing details and poor at removing noise. The reason for the difference can come from the visualization on dictionaries learned in each scheme, which show obviously different learned patterns in the dictionary. Dictionary learned from noised image itself learns some edge pattern, while the pattern in globally learned dictionary is more like area pattern thing.

For the future explorations on this topic, one thing deserves researching on is to address the problem of different magnitudes of noise require different values sparsity in the model. It can either improve on the model itself or adapt other image processing techniques to deal with the magnitude of the noise before using K-SVD algorithm. Another issue is I use $l_1$ norm to impose the sparsity on the coding, which results in fluctuation on approximation to noised patches and hardness to converge in some cases(when the uniform Gaussian noise magnitude is $\sigma = 0.2$ for example), which might be solved if we use other pursuit algorithms in the sparse coding step. Also for the globally learned dictionary, in order to make comparison here, I use the same size of dictionary as that learned from noised image itself. However, it remains unknown whether increasing the size of dictionary can overcome the disadvantage arising from using globally built dictionary. Also, other types of noise like Poisson noise is also an interesting research topic.

## References

[1] X. Zhou, C. Yang, H. Zhao, and W. Yu, “Low-rank modeling and its applications in image analysis,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1–33, 2014.

[2] J. Xiao, R. Zhao, and K.-M. Lam, “Bayesian sparse hierarchical model for image denoising,” *Signal Processing: Image Communication*, vol. 96, p. 116299, 2021.

[3] L. Shao, R. Yan, X. Li, and Y. Liu, “From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithms,” *IEEE transactions on cybernetics*, vol. 44, no. 7, pp. 1001–1013, 2013.

[4] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[5] W. Dong, X. Li, L. Zhang, and G. Shi, “Sparsity-based image denoising via dictionary learning and structural clustering,” in *CVPR 2011*. IEEE, 2011, pp. 457–464.

[6] S. Beckouche, J.-L. Starck, and J. Fadili, “Astronomical image denoising using dictionary learning,” *Astronomy & Astrophysics*, vol. 556, p. A132, 2013.

[7] L. Chen, F. Fang, S. Lei, F. Li, and G. Zhang, “Enhanced sparse model for blind deblurring,” in *European Conference on Computer Vision*. Springer, 2020, pp. 631–646.

[8] W. Dong, L. Zhang, G. Shi, and X. Wu, “Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization,” *IEEE Transactions on image processing*, vol. 20, no. 7, pp. 1838–1857, 2011.

[9] S. Ayas and M. Ekinci, “Single image super resolution using dictionary learning and sparse coding with multi-scale and multi-directional gabor feature representation,” *Information Sciences*, vol. 512, pp. 1264–1278, 2020.

[10] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[11] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[12] A. Qayyum, A. S. Malik, M. Naufal, M. Saad, M. Mazher, F. Abdullah, and T. A. R. B. T. Abdullah, “Designing of overcomplete dictionaries based on dct and dwt,” in *2015 IEEE Student Symposium in Biomedical Engineering & Sciences (ISSBES)*. IEEE, 2015, pp. 134–139.

[13] I. F. Gorodnitsky and B. D. Rao, “Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm,” *IEEE Transactions on signal processing*, vol. 45, no. 3, pp. 600–616, 1997.