Summary of Rethinking the CSC Model for Natural Images

Chenxi Liu , Adi Weinberger

1 Abstract

In this paper, the author provide some new insights regarding the CSC model and its capability to represent natural images.

They not only suggest a Bayesian connection between this model and its patch-based ancestor, but also a novel feed-forward network that follows an MMSE approximation process to the CSC model, using strided convolutions as well. The performance of their proposed supervised architecture is shown to be on par with the state of the art methods while using much fewer parameters.

2 Introduction

The field of image restoration mainly deals with the recovery of degraded images. The popular forms of image degradation include:an additive noise, blurring kernel, missing pixels, etc. In this work, they mainly focus on using sparse representation to retrieve an image from its degraded version.

Then, the authors introduced that the sparse representation model assumes that a signal $X \in \mathbb{R}^N$ composed by only a few atoms which are combined linearly. The basic atoms are:
(1) dictionary: $D \in \mathbb{R}^{N \times M}$
(2) sparse representation: $\Gamma \in \mathbb{R}^M$, where $X = D\Gamma$
(3) noisy signal: $Y = X + V \in \mathbb{R}^N$, where V is a bounded energy noise $||V||_2 \leq \epsilon$.
The goal for this problem is to seek the sparse representation $\hat{\Gamma}$ in order to estimate the origin signal via $\hat{X} = D\hat{\Gamma}$ by minimizing $\hat{\Gamma}$ according to

$$\min_{\Gamma} ||\Gamma||_0 \quad \text{s.t.} ||D\Gamma - Y||_2 \leq \epsilon \tag{1}$$

However, the sparse coding is NP-hard in general so some approximation methods are used, where a common approach is to use l_1 norm which leads to a convex problem termed Basis-Pursuit (BP). BP method has shown to be a successful way to recover a solution close to the original sparse representation, the success depends much on an important prior "the dictionary".

As a matter of fact, learning from the dictionary has been shown to be quite effective, which leads to the development of various dictionary learning algorithms. However, due

to the curse of dimensionality, those algorithms are only applicable for a few sized signals. Many algorithms try to overcome this limitation by dividing the complete signal into fully overlapping small patches and treating each individually using a local dictionary $D_L \in \mathbb{R}^{n\times m}$,

$$\forall i : \min_{\alpha_i} ||\alpha_i||_0 \quad s.t. \quad ||D_L\alpha_i - P_iY||_2 \leq \epsilon \tag{2}$$

where $P_i \in \mathbb{R}^{n\times N}$, which extracts the i-th patch from $Y (n \ll N)$, and its representation α_i is assumed to be sparse. - And once the clean patches are found, they proceed Patch Averaging (PA) to merge all the refined patches and form a final global estimate of the clean image by using:

$$\hat{X} = \frac{1}{n}\sum_i P_i^T D_L \alpha_i \tag{3}$$

Operating independently on patches neglects the dependencies between the patches. Recently, the Convolutional Sparse Coding (CSC) global model may overcome this local-global dichotomy by replacing the traditional patch-based model with a global shift-invariant one that efficiently handle the global pursuit.

In this paper, the author provided a novel insight regarding the CSC for modeling the natural images, together with an explanation for the incompetence of this model in representing the natural images reliably, and they showed that PA can be perceived as an Minimum Mean Square Error approximation to the CSC. Then they suggest suggest to obtain a CSC estimation that operates directly on an image without any preprocessing steps to improve this approximation. And finally leveraged the observations and implemented a feed-forward CNN. The results of which are on par with the state of the art supervised methods with much fewer parameters.

3 Background: Convolutional Sparse Coding (CSC)

3.1 The CSC Model

The CSC model assumes that $X \in \mathbb{R}^N$ is constructed by a sum of m convolutions of sparse feature maps $\{Z_i\}_{i=1}^m \in \mathbb{R}^N$ by filters $\{d_i\}_{i=1}^m$ of length n which is a shift invariant property in the signal. Then CSC is refered to solve the following problem:

$$\min_{\{Z_i\}_{i=1}^m} \sum_{i=1}^m ||Z_i||_0 \quad s.t. \quad X = \sum_{i=1}^m d_i * Z_i \tag{4}$$

Then they define a single sparse representation vector $\Gamma \in \mathbb{R}^{Nm}$ constructing by interlacing the sparse feature maps $\{Z_i\}_{i=1}^m$. And a global dictionary D contains N shifts of local dictionary $D_L \in \mathbb{R}^{n\times m}$, then the problem becomes

$$\min_{\Gamma} ||\Gamma||_0 \quad s.t. \quad X = D\Gamma \tag{5}$$

Besides this, they introduced some additional definitions: the sparse representation Γ can be thought of N concatenated vectors $\alpha_i \in \mathbb{R}^m$ which can termed needles and describes

the contribution of the m filters when aligned to the i-th element in X, e.g.

$$X = D\Gamma = \sum_{i=1}^{N} P_i^T D_L \alpha_i = \sum_{i=1}^{N} P_i^T s_i \tag{6}$$

This may suggest that the CSC is in fact a global model extending Patch Averaging. The CSC model and its components is displayed in the figure below:

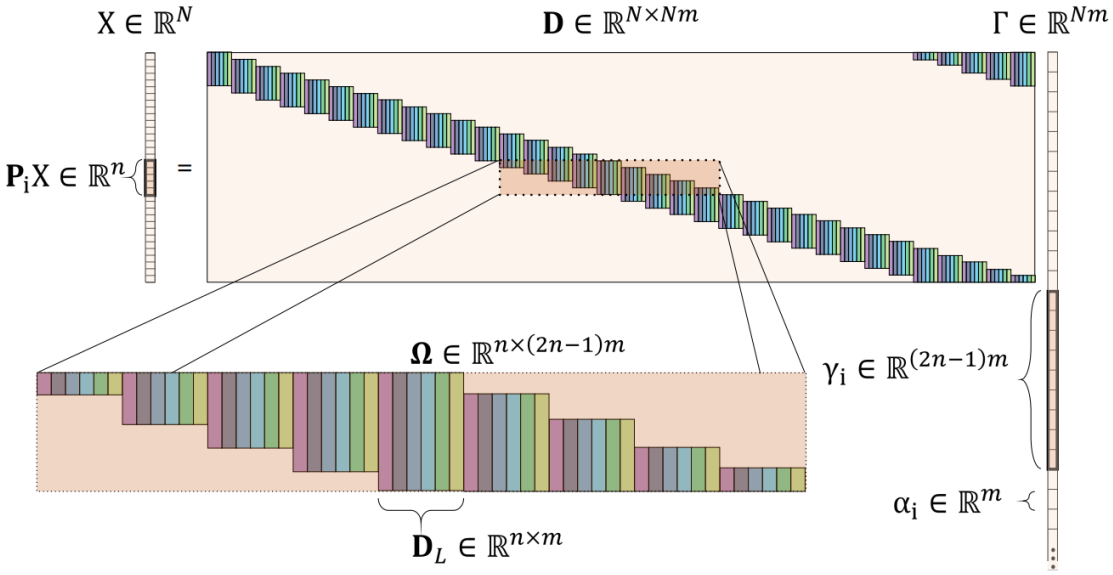


Figure 1: The CSC model and its components.

3.2 CSC in Practice

There are several applications that have achieved successful results by incorporating the CSC model, which includes:

(1) **Cartoon-texture separation**: where the goal is to decompose an image into its cartoon parts and texture blindly.

(2) **Image fusion:** In the algorithm of this application, in order to integrate complementary information from multiple source images of the same scene, each image is first decomposed into smooth and detailed layers. Then the fusion is obtained by computing the convolutional sparse representation of the detailed layer and following with a merging step which is achieved by a pixel-wise max-pooling strategy.

(3) **Single image super-resolution:** The goal is to obtain a high-resolution (HR) image from a low-resolution (LR) image by separating the LR image into a smooth and a residual image. Where the smooth part is interpolated and the residual part is coded using CSC. Then a set of filters are applied to the obtained sparse representations to recover the final detailed image.

These successful applications mainly have 2 characteristics in common and lead to some questions which will be answered in the following sections :

(1) The input image is separated into smooth and non-smooth images and the CSC only models on the detail-rich non-smooth part of the image.The first question is why CSC only perform well on non-smooth part of the image?

(2) The data is assumed to be noiseless, so the second question is can CSC benefit from noisy natural images?

The other advantages of this paper include that the authors offer a CSC deployment with an enhancement of the denoising performance that is on par with the most recent supervised methods by adopting insights on the CSC model and its MMSE approximation,

4 Why Does CSC Model Denoise Natural Images Poorly?

4.1 Poor Coherence

Previous work showed that the theoretical uniqueness and stability guarantees for the CSC problem are conditioned on the maximum number of local non-zero elements in $||\Gamma||$,

$$||\Gamma||_{0,\infty} \ < \ \frac{1}{2}(1+\frac{1}{\mu(D)}) \tag{7}$$

where $\mu(D)$ is the mutual coherence of D. However, it requires the filters and all their shifts must have low cross-correlations which may not be satisfied on natural images. Generally a CSC representation of a natural image impose a contradiction between the cardinality of the sparse representation which requires piecewise smooth filters and the coherence of the global dictionary which demands low global mutual-coherence. The fact that the CSC model cannot satisfy these two properties simultaneously making it unsuitable for natural images.

4.2 A Bayesian Standpoint

From a Bayesian point of view, the solution to the problem in Equation (1) corresponds to the MAP estimator under a sparse prior which is inferior to the Minimum MSE estimator in terms of MSE. The author then shows that PA performs a restrained approximation to the CSC MMSE estimator. By doing so, they first formalize the PA approach, and then they present the CSC MMSE estimator and finally, they show the connections of both.

(1) PA obtains a clean estimate for each patch and then averages the overlapping estimates together. The sparse representations are calculated in its Lagrangian form as follows:

$$\{\hat{\alpha}_i = \arg\min_{\alpha_i} \lambda_i||\alpha_i||_1 + \frac{1}{2}||D_L\alpha_i - P_iY||_2^2\}_{i=1}^N \tag{8}$$

(2) The MMSE of the global convolutional sparse representation vector can be written as

$$\hat{\Gamma}_{MMSE} = \mathbb{E}_S\{\mathbb{E}\{\Gamma|Y,S\}\} = \sum_{S\in\Theta} P(S)\mathbb{E}\{\Gamma|Y,S\} = \sum_{S\in\Theta} P(S)\hat{\Gamma}_S \tag{9}$$

where S is the support of Γ, $P(S)$ is the prior probability of the support, Θ is the set of all possible supports. This equation suggests that the MMSE estimator is actually a dense vector consisting of a weighted average of all the oracle estimator.

In the case where the samples supports are $D\Gamma$ results in non-overlapping tangent slices, estimating the CSC representation $\hat{\Gamma}_S$ is equal to $\frac{N}{n}$ independent local pursuits. And repeating this estimation process n times, each time with a different shift $1 \leq k \leq n$, leading to a set of estimates $\hat{\Gamma}_k$ represent the estimate obtained using the k-th shift. Then the MMSE can be approximated as

$$\hat{\Gamma}_{MMSE} \approx \sum_{i=1}^{n} P(S)\hat{\Gamma}_i \approx \sum_{i=1}^{n} \hat{\Gamma}_i \tag{10}$$

Therefore, PA can be perceived as an MMSE approximation of the CSC model, explaining its superior MSE performance when compared to a single global CSC pursuit. Armed with this insight, they proposed a better CSC MMSE estimates in the next section.

5 The Proposed Approach

5.1 Generalizing the MMSE Approximation Using Strided Convolutions

- Preliminary evaluation: In order to evaluate their approach preliminarily, they perform a denoising experiment on the contaminated images with white Gaussian noise with standard deviation $\sigma \in \{15, 25, 50, 75\}$ from the Set12 dataset. They use both the proposed strided CSC with various strides $1 \leq q < n = 11$ and the standard PA algorithm, then followed by an averaging operation.

- The results: As expected, when using CSC with a stride of 1, the denoising performance of the standard CSC is poor and the PA method is slightly better. This result can be attributed to the high coherence of the global dictionary which makes the estimated image overfit to the noise. However, when the stride of the CSC model is large but smaller than the filter size, the best results are obtained due to the fact that the coherence is restrained which allows the filters to overlap and lead to a global consensus in each of the estimates.

5.2 CSCNet – a Supervised Denoising Model

A popular method to solve the Basis Pursuit (BP) problem is the Iterative Soft Thresholding (ISTA) algorithm, i.e.

$$\hat{\Gamma} = \Gamma \frac{1}{2}||D\Gamma - Y||_2^2 + \lambda||\Gamma||_1 \tag{11}$$

where the operator operates iteratively as follows: $\Gamma_{k+1} = S_{\frac{\lambda}{c}}(\Gamma_k + \frac{1}{c}D^T(Y - D\Gamma_k))$ However, it requires a large number of iterations to converge which makes this process inefficient. To overcome this problem, the Learned Iterative Soft Thresholding (LISTA) algorithm has been proposed to approximate the sparse coding process, which strictly follows L iterations iterates as follows:

$$\Gamma_{k+1} = S_\tau(\Gamma_k + \frac{1}{c}A(Y - B\Gamma_k)) \tag{12}$$

where A is the convolution operator, B stands for a transposed convolution operator, Y is the noisy image, S_τ is the threshold vector learned in a supervised manner. Once Γ_L is at hand, the estimated clean image will be obtained by a linear transposed-convolutional decoder, i.e. $\hat{X} = C\Gamma_L$, and the number of parameters does not grow with the number of unrolled iterations L.

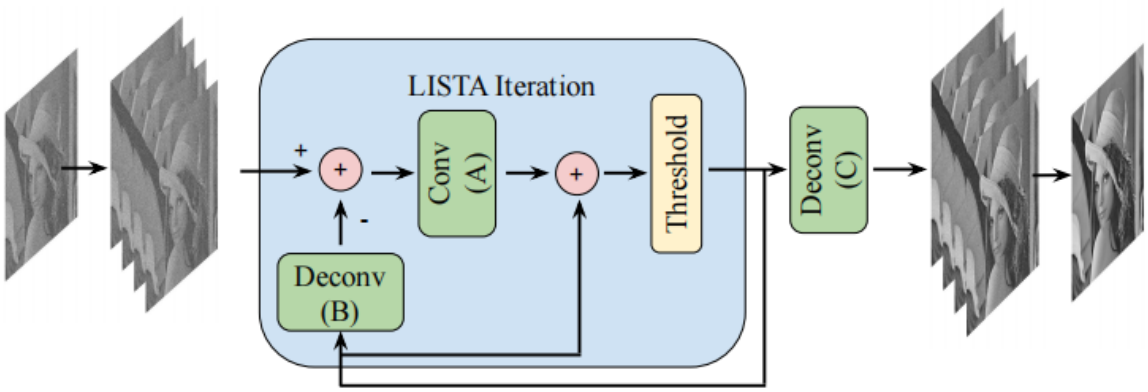


Figure 2: The CSCNET architecture.

5.3 Experiments

To train the proposed model in their experiment, both the clean and noisy input-output image pairs are prepared. Where the noisy images are obtained by adding white Gaussian noise with a constant standard deviation σ in each iteration. 4 models are trained in total and for each model, the noise level $\sigma = \{15, 25, 50, 75\}$ are used, together with the following settings:

- 175 filters of size 11×11
- stride: q = 8
- total iterations: L = 12
- ADAM optimizer
- l_2 loss: i.e. $L(X, \hat{X}) = ||X - \hat{X}||_2^2$
- learning rate 10^{-4}, which is decreased by a factor of 0.7 every 50 epochs and iterate over 250 epoch
- optimizer parameter: $\epsilon = 10^{-3}$
- dataset: BSD68

Results: Their proposed model outperforms the first 4 models shown in table2 and is on par with DnCNN and FFDNet but with much fewer parameters. The experiments are performed on both the BSD68 dataset and the 3 channels color-BSD68 dataset, both result in similar performances. The specific performance results can be shown in table 2 and table 4, the parameter results can be shown in table 3:

Table 2: Denoising performance (PSNR) on the BSD68 dataset.

σ	BM3D	WNNM	TNRD	MLP	DnCNN	FFDNet	CSCNet
15	31.07	31.37	31.42	–	31.72	31.63	31.57
25	28.57	28.83	28.92	28.96	29.22	29.19	29.11
50	25.62	25.87	25.97	26.03	26.23	26.29	26.24
75	24.21	24.40	–	24.59	24.64	24.79	24.77

Table 4: Denoising performance (PSNR) on the color-BSD68 dataset.

σ	CBM3D	CDnCNN	FFDNet	CSCNet
15	33.52	33.89	33.87	33.83
25	30.71	31.23	31.21	31.18
50	27.38	27.92	27.96	28.00
75	25.74	24.47	26.24	26.32

Table 3: Comparison of number of parameters in leading denoising architectures.

Model	First layer	Last layer	Mid layers	Total
DnCNN	$3 \times 3 \times 1 \times 64$	$3 \times 3 \times 64 \times 1$	$(3 \times 3 \times 64 \times 64 + 128) \times 15$	556,032
FFDNet	$3 \times 3 \times 5 \times 64$	$3 \times 3 \times 64 \times 4$	$(3 \times 3 \times 64 \times 64 + 128) \times 13$	486,080
CSCNet	–	$11 \times 11 \times 175 \times 1$	$(11 \times 11 \times 175 \times 1) \times 2 + 175$	63,700

Differences: There are 2 main differences between the proposed model and the other two leading models:

1. Number of parameters: The proposed model use much fewer parameters compared to other modern methods since the number of parameters does not grow with the depth of the proposed model.

2. Batch Normalization: The proposed models rely only on the CSC prior and didn't employ batch normalization like what's been done in other methods to improve the performance and convergence rate of the trained model.

6 Conclusion

In this work, inspired by the patch-averaging (PA) scheme and the origin of its success, the author proposed an MMSE approximation pursuit that overcomes the limitations of the

CSC model in representing natural images in the presence of noise. The CSC feed-forward architecture is proposed and the performance is on par with the best supervised denoising algorithms in the literature and in the mean time they use much fewer parameters.

7 Research Work

As written in the article, the author proposed a CSCNet architecture to recover the images degraded by additive white Gaussian noise. However, there are all kinds of popular forms of degradation – one of them is blurring kernel, which is more common in real settings, such as the motion blur we see from previous homework. So in order to explore the performance of CSCNet with other noises, we train the CSCNet on two kinds of blurring kernels: Gaussian kernel and Sinc kernel. For all the tables bellow - Denoising performance (PSNR) are run on BSD68 dataset.

We will compare the tables we got to Table 2 in the article. Our running results on two kinds of blurring kernels, where

1. Sigma value defines the amount of blur. Bigger values = more blurring.
2. Kernel size defines how many pixels to sample during the convolution.

   We wanted to get a "feeling" and test some different kinds of kernel sizes and sigma. The results depend on kernel size and on sigma. We can see that the smaller the kernel size and sigma the better results we get.

Kernel Size	**Sigma**				
	1	**2**	**3**	**4**	**5**
4	42.62774		36.86782		38.51119
16	39.33191	30.18984	26.97941	26.87799	26.22209
32	38.52703		26.82577		24.91073
64	38.27376	29.35906	27.09874	25.75560	24.84358
128	38.29174	29.91101	27.05281	25.65735	24.85225

Table 1: Denoising performance (PSNR) on the BSD68 dataset with different Gaussian kernel size and sigma

Kernel Size	Start PSNR	Final PSNR
16	26.09930	37.47804
64	33.72630	28.56527
128	31.44485	29.17118

Table 2: Sinc kernel: start and final PSNR for different kernel sizes

We can see from the tables above that the recovery from the start PSNR to the final PSNR is greater than the recovery they got (see table below).

Kernel Size	Start PSNR	Final PSNR
4	28.78233	42.62774
16	28.92127	39.33191
64	38.27376	28.08811
128	38.29174	28.80296

Table 3: Gaussian kernel: start and final PSNR for different kernel sizes, with $\sigma = 1$

Noise + Kernel Size, $\sigma = 25$	Start PSNR	Final PSNR
Random noise, kernel size=11	**26.14776**	**28.81848**
Gaussian kernel, kernel size = 4	33.48560	37.18017
Gaussian kernel, kernel size = 11	**22.36463**	**30.08605**
Gaussian kernel, kernel size = 16	21.48514	29.78338

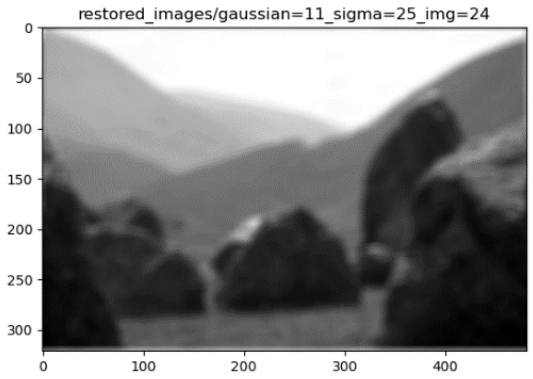
Table 4: Denoising performance (PSNR) on BSD68 dataset with different noise

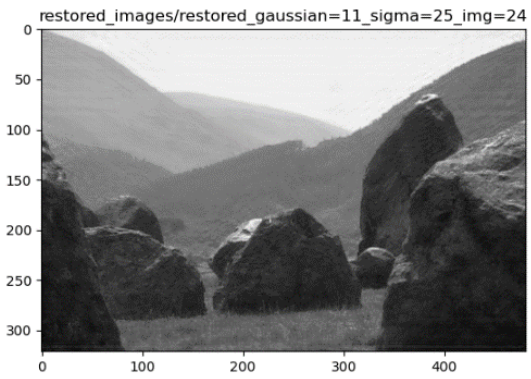
Here we compare the same sigma as they used in the article (sigma = 25) with different kernel sizes. The most interesting result we got is the one with the same sigma (25) and the same kernel size (11) as they use in the article.

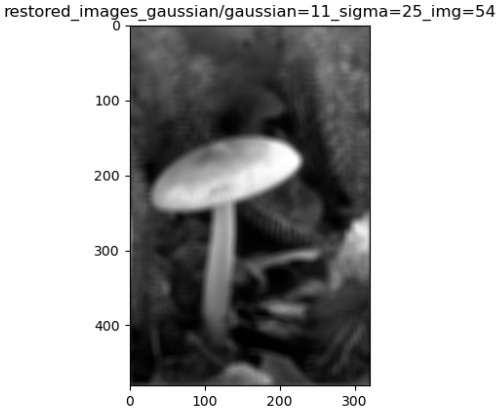
As we can see from the table above, we got a better result with the same kernel and sigma (11,25). And we also got a much better recovery result from start to final PSNR for the same kernel and sigma: from 22.36463 to 30.08605 with gaussian noise as compared to 26.14776 to 28.81848 with random white noise.

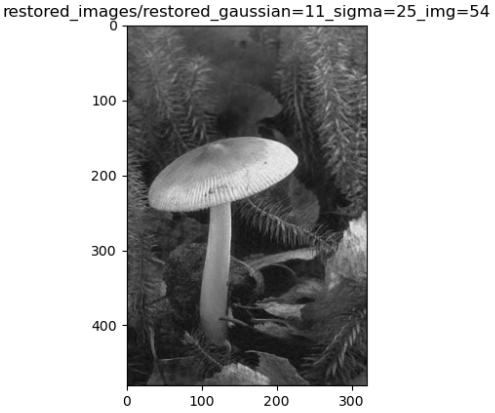
We add some noisy and recovered image pairs below for comparisons:

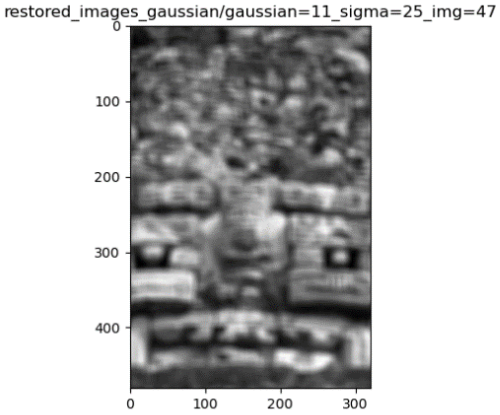
Restored blurry kernel images with Gaussian kernel: kernel size = 11, sigma = 25:

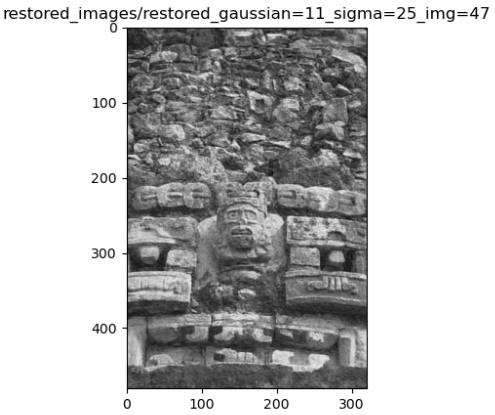




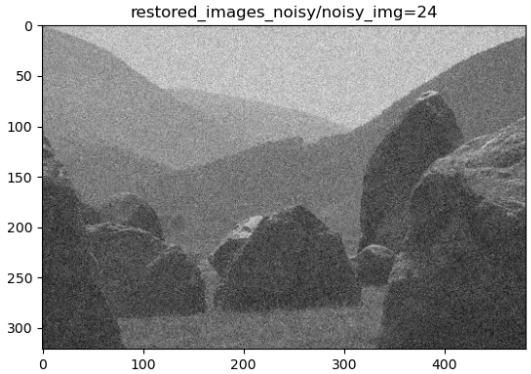


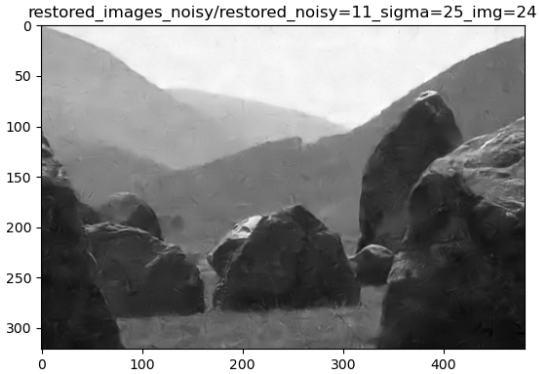


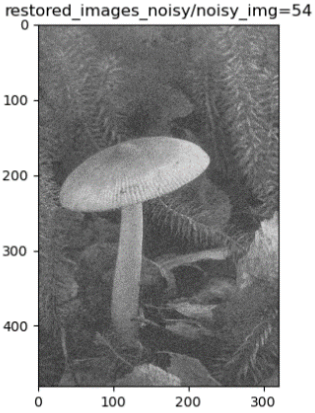


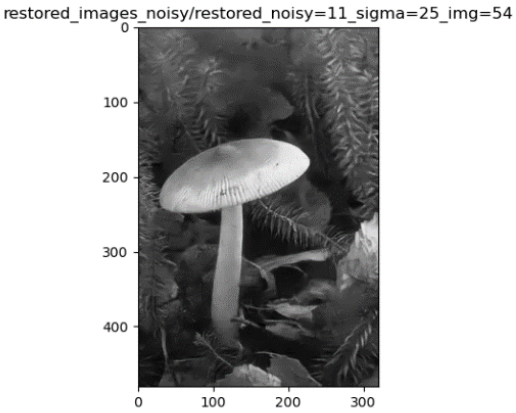


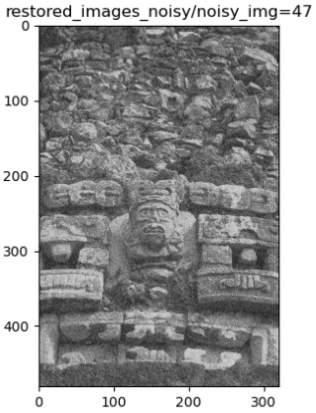
Restored additive noise images: kernel size = 11, sigma = 25

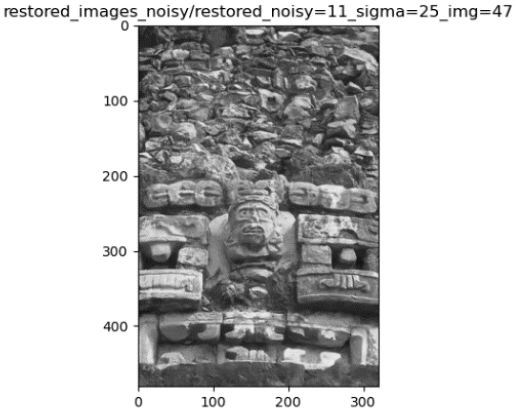





restored_images_noisy/noisy_img=54
0
100
200
300
400
0
100
200
300


restored_images_noisy/restored_noisy=11_sigma=25_img=54
0
100
200
300
400
0
100
200
300


restored_images_noisy/noisy_img=47
0
100
200
300
400
0
100
200
300


restored_images_noisy/restored_noisy=11_sigma=25_img=47
0
100
200
300
400
0
100
200
300