Adaptive Representations for Image Restoration

Ruomei Yan

Department of Electronic and Electrical Engineering

The University of Sheffield

A thesis submitted for the degree of

Doctor of Philosophy (PhD)

April 2014

Abstract

In the field of image processing, building good representation models for natural images is crucial for various applications, such as image restoration, sampling, segmentation, etc. Adaptive image representation models are designed for describing the intrinsic structures of natural images. In the classical Bayesian inference, this representation is often known as the prior of the intensity distribution of the input image. Early image priors have forms such as total variation norm, Markov Random Fields (MRF), and wavelets. Recently, image priors obtained from machine learning techniques tend to be more adaptive, which aims at capturing the natural image models via learning from larger databases. In this thesis, we study adaptive representations of natural images for image restoration.

The purpose of image restoration is to remove the artifacts which degrade an image. The degradation comes in many forms such as image blurs, noises, and artifacts from the codec. Take image denoising for an example. There are several classic representation methods which can generate state-of-the-art results. The first one is the assumption of image self-similarity. However, this representation has the issue that sometimes the self-similarity assumption would fail because of high noise levels or unique image contents. The second one is the wavelet based nonlocal representation, which also has a problem in that the fixed basis function is not adaptive enough for any arbitrary type of input images. The third is the sparse coding using over-complete dictionaries, which does not have the hierarchical structure that is similar to the one in human visual system and is therefore prone to denoising artifacts.

My research started from image denoising. Through the thorough review and evaluation of state-of-the-art denoising methods, it was found that the representation of images is substantially important for the denoising technique. At the same time, an improvement on one of the nonlocal denoising method was proposed, which improves the representation of images by the integration of Gaussian blur, clustering and Rotationally Invariant Block Matching. Enlightened by the successful application of sparse coding in compressive sensing, we exploited the image self-similarity by using a sparse representation based on wavelet coefficients in a nonlocal and hierarchical way, which generates competitive results compared to the state-of-the-art denoising algorithms. Meanwhile, another adaptive local filter learned by Genetic Programming (GP) was proposed for efficient image denoising. In this work, we employed GP to find the optimal representations for local image patches through training on massive datasets, which yields competitive results compared to state-of-the-art local denoising filters. After successfully dealt with the denoising part, we moved to the parameter estimation for image degradation models. For instance, image blur identification uses deep learning, which has recently been proposed as a popular image representation approach. This work has also been extended to blur estimation based on the fact that the second step of the framework has been replaced with general regression neural network. In a word, in this thesis, spatial correlations, sparse coding, genetic programming, deep learning are explored as adaptive image representation models for both image restoration and parameter estimation.

We conclude this thesis by considering methods based on machine learning to be the best adaptive representations for natural images. We have shown that they can generate better results than conventional representation models for the tasks of image denoising and deblurring.

To my parents and my husband

Acknowledgements

Foremost, I would like to thank my supervisor Dr. Ling Shao, who has shared his expertise with me during more than three years and has been of invaluable help. On the one hand, he has given me maximum independence in pursuing my ideas and lots of encouragement. On the other hand he has provided thorough advice on research innovation and drafting papers.

Second, I would like to thank my fellow colleagues: Simon Jones, Dr. Xiantong Zhen, Di Wu, Fan Zhu, Bo Dong, Jun Tang, Feng Zheng, Li Liu, Dr. Xuezhi Wen, Dr. Peng Li, Dr. Dabo Guo, Redzuan Bin Abdul Manap, Mengyang Yu, Yang Long, Xu Dai, Peng Peng, and Ziyun Cai who have been so friendly and helpful during my PhD study. Through many group discussions and collaboration we had, I learned a lot from them. In the department of Electronic and Electrical Engineering, I specially thank the previous head of department Prof. John David and the career adviser Kevin Mahoney, who have been incredibly kind and helpful. Postgraduate Administrator, Ms. Hilary J Levesley, Technical Manager Mr. James Screaton and Safety Officer Ms. Dianne Webster are always ready to help when I need them.

Third, I would like to express my gratitude to many people who have contributed to my work in this thesis: Dr. Sascha Cvetkovic, Dr. Yan Liu, Dr. Xuelong Li, Dr. Shenghua Zhong, Mateusz Malinowski, Dr. Christopher Burger, Dr. Julien Mairal, Junyuan Xie. I would like to thank my examiners Dr. Peter H. N. de With and Dr. Wei Liu for reviewing my thesis.

In addition, there are also other researchers and friends I would like to thank from the previous VIE group of the University of Sheffield including Ji Ni, Zhizhong Yu, Yilong Cao, Bo Gao, Jing Li, Wenting Duan, Jacks Diao, and other friends: Cheng Zhou, Yang Li, Lingli Shen, Christopher Quickfall, Jing Wang, Shengheng Tan, Xiaona Xu, Ying Lan Ang, Laura Vergoz, James Godfrey, Andrew Landels, Justina Mischewski, Maryam Sadeghi, Peng Gong.

Besides, everything would not be possible without the support of the China Scholarship Council (CSC) for the financial support of my PhD study in Sheffield. So thanks to them and my great country China.

Above all, my heartfelt thanks go to my parents, my parents-in-law, and my husband Simon Jones. I know I wouldn't finish the PhD without their non-stopping love, encouragement and support.

PUBLICATIONS

- **R. Yan**, L. Shao, and Y. Liu, "Nonlocal hierarchical dictionary learning using wavelets for image denoising", *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4689-4698, Aug. 2013.
- L. Shao, **R. Yan,** X. Li, and Y. Liu, "From heuristic optimization to dictionary learning: a comprehensive comparison of image denoising algorithms", *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1001-1013, Jul. 2014.
- **R. Yan**, L. Shao, L. Liu, and Y. Liu, "Natural image denoising using evolved local adaptive filters", *Signal Processing*, vol. 103, pp. 36-44, Oct. 2014.
- **R. Yan**, L. Shao, "Image blur classification and parameter identification using two-stage deep belief networks", in Proc. *British Machine Vision Conference*, Bristol, UK, 2013.
- **R. Yan**, L. Shao, S. D. Cvetkovic and J. Klijn, "Improved Nonlocal Means Based on Pre-Classification and Invariant Block Matching", *IEEE/OSA Journal of Display Technology*, vol. 8, no. 4, pp. 212-218, Apr. 2012.
- **R. Yan**, L. Shao, "Blur kernel classification and estimation from a single image", submitted to *IEEE Transactions on Image Processing*, 2013.

Contents

List of Figures

List of Tables

1

Introduction and Related Work

1.1 Introduction

Image restoration is the process of recovering high quality images from degraded images using inference techniques. The field of image restoration includes a number of applications: noise reduction, image deblurring, image upscaling, image inpainting, image super-resolution, etc. The degradation model can be expressed as:

$$\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n} \tag{1.1}$$

where $\boldsymbol{k}$ is the degradation kernel matrix, $\boldsymbol{x}$ is the latent image, $\boldsymbol{y}$ is the observed image, and $\boldsymbol{n}$ is the additive noise. In different applications, $\boldsymbol{n}$ is almost always assumed to be Gaussian noise. $\boldsymbol{k}$ stands for different kernels. For instance, in noise reduction $\boldsymbol{k}$ is a identity matrix while it is a blur matrix in the blurring model.

Image restoration has been commonly explored in many electronic devices, such as cameras, mobile phones, digital TVs, or internet-based online display softwares. In order to improve image quality, there is limited amount of work we can do by modifying the hardware devices. Therefore, image restoration as a post-processing stage is useful and important.

The most frequently used inference technique for image restoration is Bayesian Inference. Assuming that the observed image is $\boldsymbol{y}$ and the latent image is $\boldsymbol{x}$, the object for us is to compute the optimal posterior probability (8):

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \tag{1.2}$$

$p(\boldsymbol{y}|\boldsymbol{x})$ is usually Gaussian distribution, and $p(\boldsymbol{x})$ denotes the image prior. According to the image prior, the Bayesian based methods could be divided into two categories: heuristic and transform-based methods.

Heuristic image priors are estimated based on certain types of images. For instance, Bilateral filters, Gaussian filters (9), etc. However, these filters are based on the assumption that natural images have prior distributions similar to a heavy tailed distribution, which is the smoothness assumption. In fact, the distribution of natural image cannot be represented by just a single type of function. Therefore, more and more transform-based methods are designed to represent images with a better linear combination of "basis functions". Assume that these "basis" functions are trained on high quality images. When they are applied to the problem of image inpainting (many pixels are missing in the corrupted images), we are still capable of using these "basis" to construct the underline structures of images.

The question for these transform-based image representation methods is: how can we make the "basis" functions adaptive to various types of natural images? Earlier research regarding this has focused on nonlocal methods (10, 11), which exploit the image self-similarity. Later, wavelet transform and its variants have been used in many state-of-the-art methods. For instance, the discrete cosine transform (12) and wavelets (13) have been used in one of the state-of-the-art algorithm BM3D (14) several years ago. Steerable wavelets (15) have been adopted in the successful denoising method BLS-GSM (16). Recently, after the successful application of sparse coding to compressive sensing, more researchers investigated this tool for image restoration problems (17) (18). Currently, the popular learning-based methods learn image priors such as Gaussian Mixture Models, sparse representation, Deep Neural Networks, etc.

The work presented in the thesis is mainly focused on how to improve current image representation models for the task of image restoration. For spatial domain methods, an improved nonlocal means based on Pre-classification and invariant block matching is proposed in chapter 3. For learning-based methods, a nonlocal hierarchical dictionary in wavelet domain is proposed in chapter 4. Also, a genetic programming based denoising filter is proposed in chapter 5 too. Later, another large part of the work (chapter 6) is based on deep learning, which is parameter estimation for noise level and image blur.

Related work regarding the research contributions are explained in the rest of this introduction.

1.2 Sparse Coding

1.2.1 The definition of sparse coding methods

The purpose of sparse coding is to represent input data with the results of the weighted averaging dictionary elements and at the same time the sparse codes (the weights) should be sparse (avoid overfitting). From the perspective of image compression, the sparse codes means that we fully exploit the redundancy (correlations) of the input data, so it means that we have a good representation and information has not been lost too much during the process of image compression. This idea can be transferred to image restoration. For instance, in the case of removing Gaussian noise, if we can represent an image with good sparse representations, it means that most of the image textures can be linearly approximated by "meaningful" dictionary elements. When we do the weighted averaging for the approximated results, the random Gaussian noise will be "averaged out". However, if the sparse representation is incorrect, the reconstructed image would have unwanted artifacts, which will take extra post-processing steps to deal with.

Assuming that a training set of m input vectors $\boldsymbol{y}_1, ..., \boldsymbol{y}_m$, their corresponding sparse coefficients $\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_m$, and the dictionary $\boldsymbol{d}^1, ..., \boldsymbol{d}^n$.

The objective function could be described as:

$$\begin{aligned} \min_{\boldsymbol{d}_j, \boldsymbol{\alpha}_i} g(\alpha) = & \sum_{i=1}^{m} \|\boldsymbol{y}_i - \sum_{j=1}^{n} \boldsymbol{d}^j \boldsymbol{\alpha}_i^j\|^2 \\ & + \lambda \sum_{i=1}^{m} \sum_{j=1}^{n} \phi(\alpha_i^j) \\ subject \quad to \quad & \|\boldsymbol{d}^j\|^2 \leq \varepsilon, \forall j = 1, ..., n \end{aligned} \tag{1.3}$$

In terms of the sparse prior, the sparse function has the following forms: L_1 penalty function $\|\alpha_i^j\|_1$, L_0 penalty function $\|\alpha_i^j\|_0$, epsilon L_1 penalty function $((\alpha_i^j)^2 + \epsilon)^{\frac{1}{2}}$, and log penalty function $\log(1 + (\alpha_i^j)^2)$. In most image restoration applications, L_1 and L_0 are often used as the regularization terms.

So why does L_1 norm introduce sparsity? L_1 norm means we use the absolute values of the sparse codes. Assume that we need to optimize Eq. (1.4), the solution to this is $\alpha^*(\lambda) = \frac{1}{\lambda^2}$, which is never zero. In general, when the λ is very large, the sparse codes are nearly zero. When the λ is very small, the sparse codes are relatively large.

The explanation in 1-D signal can be denoted as:

$$\min_{\alpha \in \mathbb{R}} [q(\alpha) = (y - D\alpha)^2 + \lambda|\alpha| \tag{1.4}$$

where y is a scalar.

In the L_2 squared regularization, we can observe from the figure that when the sparse codes are approaching zero the sparsity term is reaching zero too. However, in the case of the L_1 regularization, the sparsity term stays 1 or -1 when sparse code is close to zero. That is why L_1 is chosen for many sparse coding applications (2).

1.2.2 The application of sparse coding in Image Processing

In image reconstruction, we have the degradation model of the observed image patch as:

$$\boldsymbol{y}_i = \boldsymbol{k}_i * \boldsymbol{x}_i + \boldsymbol{n}_i, \tag{1.5}$$

where $\boldsymbol{k}_i$ is the kernel for the convolution. In image blurring process, this $\boldsymbol{k}_i$ could be blur kernels and $*$ means the convolution operation. For a noisy image, this $\boldsymbol{k}_i$ is identity matrix. The process of image inpainting is slightly different from this framework. It is the process of filling in the missing part of an image and those missing parts normally cannot be described by the convolution kernels (the results of the inpainting process have been shown in Fig. 1.3).

For any input low quality image $\boldsymbol{y}$, the goal of image restoration is to reconstruct $\boldsymbol{x}$ with very little information. Considering that sparse coding is a local image representation model, we apply it to image patches rather than whole images. For an input image, overlapping patches $\mathbf{y}_i$ are extracted and formed into a training set. The initial dictionary for this process is trained from high quality image patches, which are extracted from the training dataset using the same method. Let $D \in \mathbb{R}^{m \times K}$ be a dictionary of K atoms, $X \in \mathbb{R}^{m \times N}$ be a set of N column data vectors $\boldsymbol{x}_i \in \mathbb{R}^m$, and each of the atom can be represented as column $\boldsymbol{d}_k \in \mathbb{R}^m$. For each vector $\boldsymbol{x}_i$, the aim of sparse coding is to find the optimized sparse codes $\boldsymbol{\alpha}_i$ for representing the vector by linearly combining dictionary elements from $D \in \mathbb{R}^{m \times K}$ with the weight. For each training set X, the sparse codes are A, which can be expressed as:

$$A = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_N] \in \mathbb{R}^{K \times N} \tag{1.6}$$

The training of the dataset X is to find the best dictionary D to reconstruct X:

$$X \simeq DA \tag{1.7}$$

The optimization process can be denoted as the following equation:

$$(A^*, D^*) = \arg\min_{A,D} \|X - DA\|^2 + \lambda \|A\|_p \tag{1.8}$$

where p indicates the type of the norm ($p = 0, 1, 2$). In the problem of image inpainting, this formula needs to be changed to:

$$(A^*, D^*) = \arg\min_{A,D} \|M(X - DA)\|^2 + \lambda \|A\|_p \tag{1.9}$$

where M is the binary mask.

Usually, the dictionaries trained on high quality images are used for the initial dictionary in the image restoration process. For instance, in previous work by Mairal *et al.* 's (1), the actual dictionary used for image denoising is trained on the initial dictionary learned with the high quality images. After the dictionary and the sparse codes are trained, for every patch $\mathbf{x}_i$ the reconstructed result is obtained by averaging all results of m patches which are overlapping with $\mathbf{x}_i$:

$$\hat{\boldsymbol{x}}_i = \frac{1}{m} \sum_{j=1}^{N} \boldsymbol{R}^j DA \tag{1.10}$$

where $\mathbf{R}^j \in \mathbb{R}^{N \times m}$ is the binary matrix which selects patch j from the image. The results of the work from Mairal *et al.* (1) have been shown in Fig. 1.1. The results of another classic denoising method based on dictionary learning have also been shown in Fig. 1.2.

The dictionary learning techniques can still be used for video processing. For instance, for video denoising, the dictionary learning method could be applied to each frame. However, temporal redundancy should be exploited for improving the performance (19).

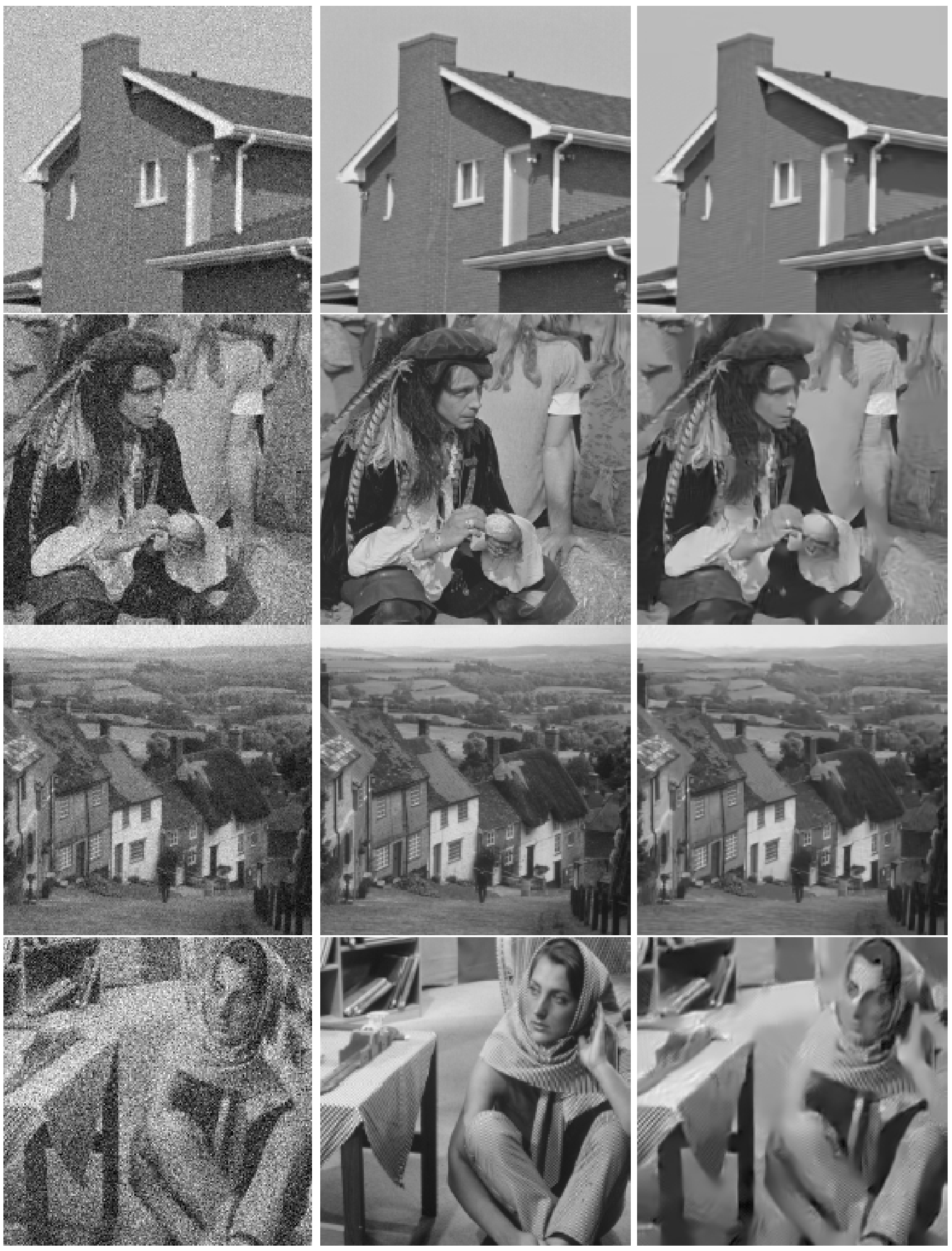


Figure 1.1: The results of image denoising in Mairal *et al.* 's work (1). Left: noisy images. Middle: original images. Right: restored images. From top to bottom: house ($\sigma = 15$), man image ($\sigma = 50$), hill image ($\sigma = 50$), barbara ($\sigma = 100$). This figure is borrowed from (2), with the courtesy of Dr. Julien Mairal.

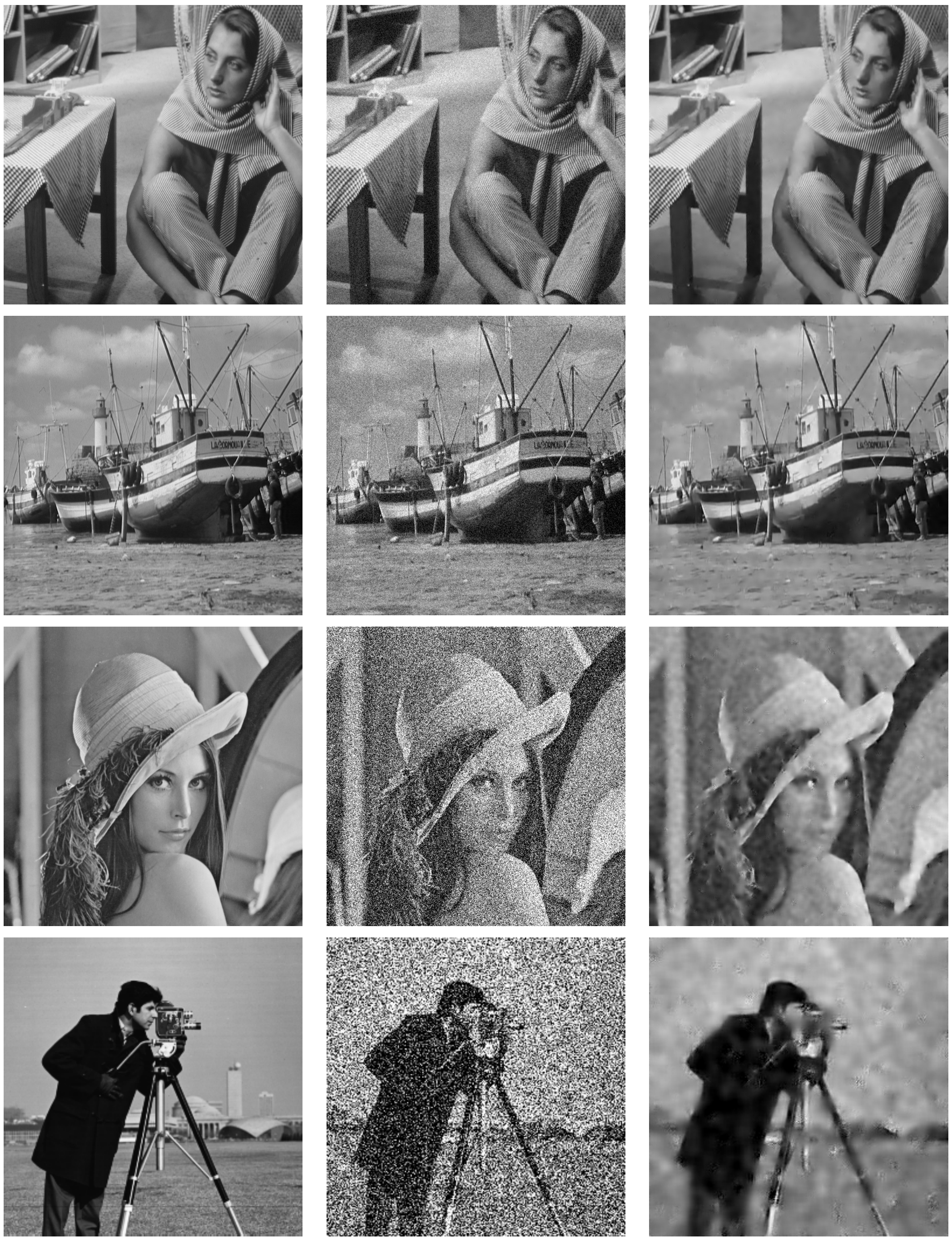


Figure 1.2: The denoising results of KSVD. From top to bottom, the noise levels are: $\sigma = 15$, $\sigma = 20$, $\sigma = 75$, $\sigma = 100$.

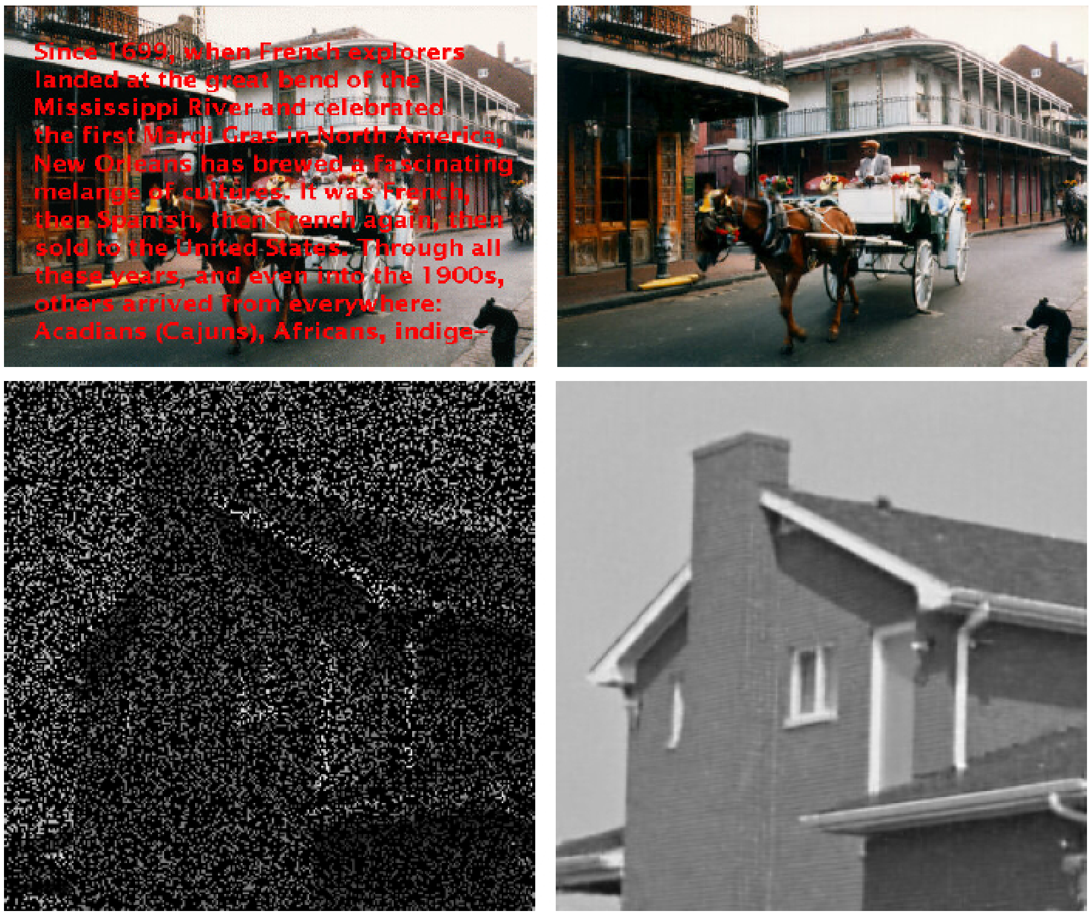

Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

Figure 1.3: The inpainting results. This figure is borrowed from (2), with the courtesy of Dr. Julien Mairal.

1.3 Deep Learning

1.3.1 The deep architectures

Deep learning is a recent popular research topic in the machine learning research community. It has been applied to many computer vision research areas (20). For instance, speech recognition, object recognition, and image retrieval, etc. The successful applications in these research directions are all to do with the advantage of deep learning acting as a good representation method. Most of the conventional data representations are all in the form of handcrafted features, which are too heuristic to be adaptive to the various input data. Many feature learning methods in the area of computer vision try to use human prior knowledge to improve the performance. However, these features have limited performance in terms of obtaining discriminative information from the input data. In deep architectures, according to the target in the output layer, this deep learning process can help to learn a more representative model. For instance, the model could have edge detectors in the first layer, more abstract feature in the second layer, and then continued layers with more abstract features. In this section, some specific applications in computer vision are listed below.

1.3.2 The overview of deep learning in computer vision

1.3.2.1 Deep belief nets

Deep belief nets, as a generative model, are multiple cascaded Restricted Boltzmann Machines as shown in Fig. 1.4. The reason why we need multiple layers is that it could provide a more meaningful learning system to capture the structure in the input data, which is usually too complicated for a single layer learning model. The goal of learning this generative model is to tune the weights between layers and biases within each layer to better represent the input observed data. The learning process is: the current layer is treated as input layer for the next layer as the architecture in restricted Boltzmann machine. The whole DBN is trained as a cascade of many RBMs. Once all the hidden layers are trained, the DBN training is finished accordingly. For lower layers of the network, very preliminary features are extracted, i.e., edges. When it gets to higher levels, more semantic features are extracted. For instance, primitive shape detectors, high level visual abstractions. In deep belief nets, nodes in hidden layers are

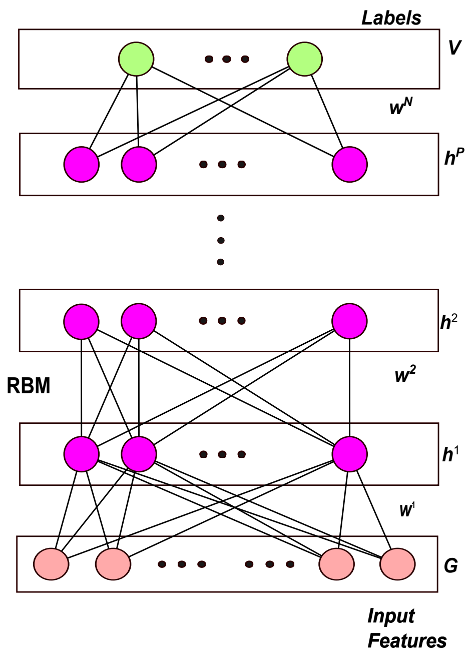


Figure 1.4: The hierarchical structure of deep belief networks.

supposed to be conditionally independent. However, given one node from the previous layer (either hidden or visible layer), the nodes in current layer are still dependent on each other, which has been addressed as "explaining away" (21). This explains why when we use DBN for reconstructing input data it can improve the actual restored results. From this perspective, deep belief nets are very different from Neural Network, which is almost just a regressor and there is very few semantic meaning involved.

Though deep belief networks are designed as generative models, they can be used for discriminative learning (22). It has been proved in Erhan *et al.* 's work (23) that pre-trained deep belief networks are very good initialization for the deep architectures, which can be used in a discriminative way by the process of backpropagation.

An example of using deep belief networks for face recognition was proposed by Huang *et al.* in (24). In this work, the convolutional deep belief networks are applied to local binary patterns (25) rather than image pixels. This method proves that by using deep learning and the basic handcrafted features the results of face verification can be improved because DBN can learn a better higher level representation (descriptor) from the local binary pattern.

Another example of using deep belief nets for image classification was proposed by Sheng *et al.* in (26). The deep belief network in this work is learned in a semi-supervised way with novel initialization for each hidden layer. This method outperforms most deep learning methods in terms of classification rates. At the same time, this algorithm is invariant to noise, which demonstrates the advantages of the deep belief network as a generative model.

1.3.2.2 Stacked denoising auto-encoders

The idea of denoising autoencoder was proposed (27) to learn the existing patterns within input layer under the circumstances of partial corruption. It has similarity with the process "denoising", which is to recover the high quality image from a noisy input. However, here, the goal of the learning is not just reconstructing good input but learning high level representations of the input data.

The process of training denoising auto-encoders is different from training restricted boltzmann machines in the way that the object function for this training is to minimize the reconstruction error between the raw input and the restored vector while in RBM the goal is to sample the distributions for constructing an informative model according to the input layer. The first problem is a very straight forward minimization problem by stochastic gradient descent. However, for RBM, the optimization is applied to the log likelihood, which also involves gradients. These gradients are used for simulating the model regarding the distribution of the input data (21). As is described in Fig. 1.5, assume that the raw input is $\mathbf{r}$, the corrupted input is $\mathbf{c}$, the hidden layer is $\mathbf{h}$, and the reconstruction layer is $\mathbf{z}$, then the training process is to minimize:

$$\arg\min_{\Theta} D(\mathbf{r}, \mathbf{z}) \tag{1.11}$$

where D is the loss function deciding the distance between $\mathbf{r}$ and $\mathbf{z}$, Θ is the weights between layers and the bias within each layer.

Since the stacked denoising autoencoder is resilient to noise in the input layer, this deep architecture has been applied to many vision tasks, for instance, image classification. It is very common to see SDA act as the pre-training stage for deep neural networks. One of the examples of a successful application is using SDA for image denoising and blind image inpainting (3). For instance, in the image denoising task, the SDA is trained on the input noisy image and after reconstructing from the hidden layer, the result of which is compared with the high quality ground truth to optimize the SDA. The performance of this algorithm is comparable to one of the popular learning-based denoising method KSVD. In the task of blind image inpainting, this SDA model works very well because when the missing pixels are completely random SDA can still recover the high quality image. This is due to the fact that the nature of the denoising autoencoder is not related to image topology. Some of the visual results of this work can be found in Fig. 1.6 and Fig. 1.7.

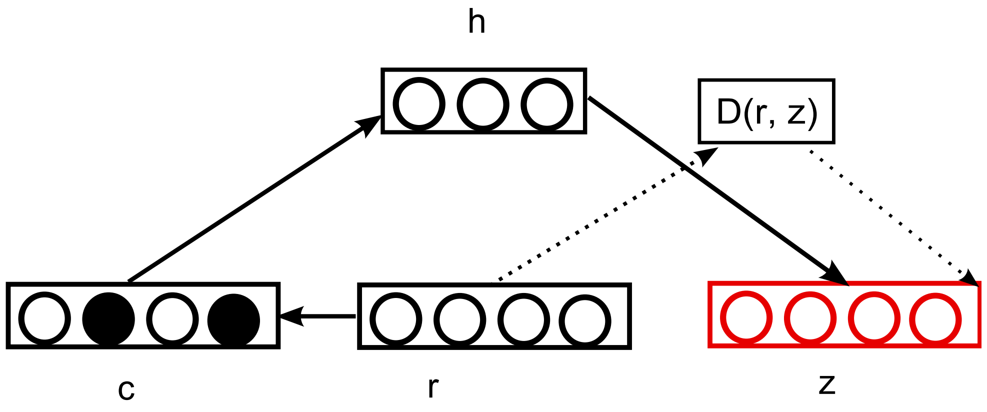


Figure 1.5: The denoising autoencoder.

Another example is the recent work from Agostinelli *et al.* (28), which has improved Xie's work by training multiple stacked sparse denoising autoencoders and concatenating the feature vectors by weighted averaging using radial basis function network. This method has overcome the problem that exists in most state-of-the-art denoising algorithms that they cannot handle various noise models rather than Gaussian. In the testing stage, the input noisy image can contain the noise which does not exist in the training set. This framework has also been applied to the problem of classifying noisy images in MNIST dataset. Some of the visual results of this work can be found in (3).

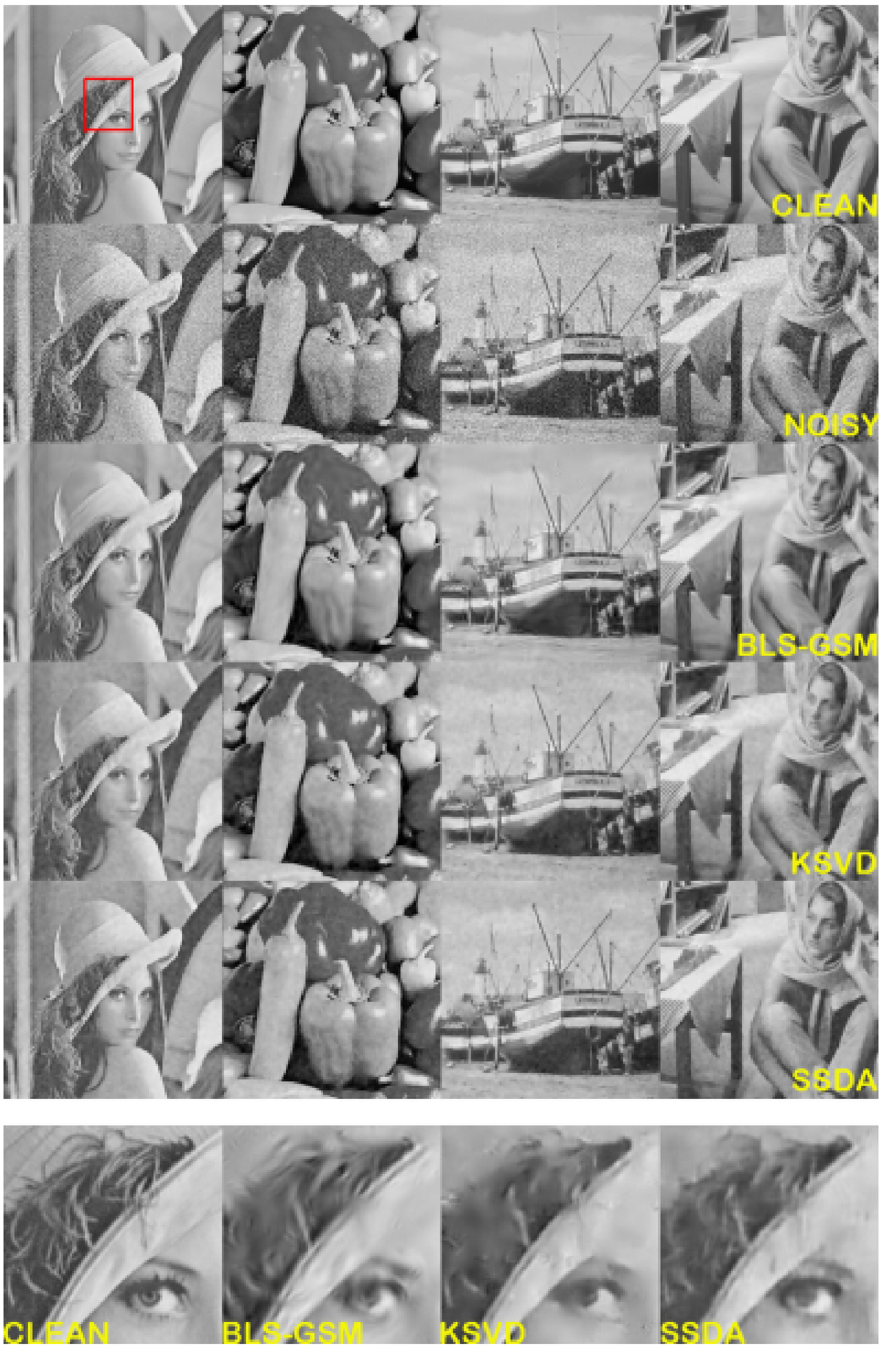


Figure 1.6: The denoising results of the denoising autoencoder. This figure is borrowed from (3), with the courtesy of Junyuan Xie.

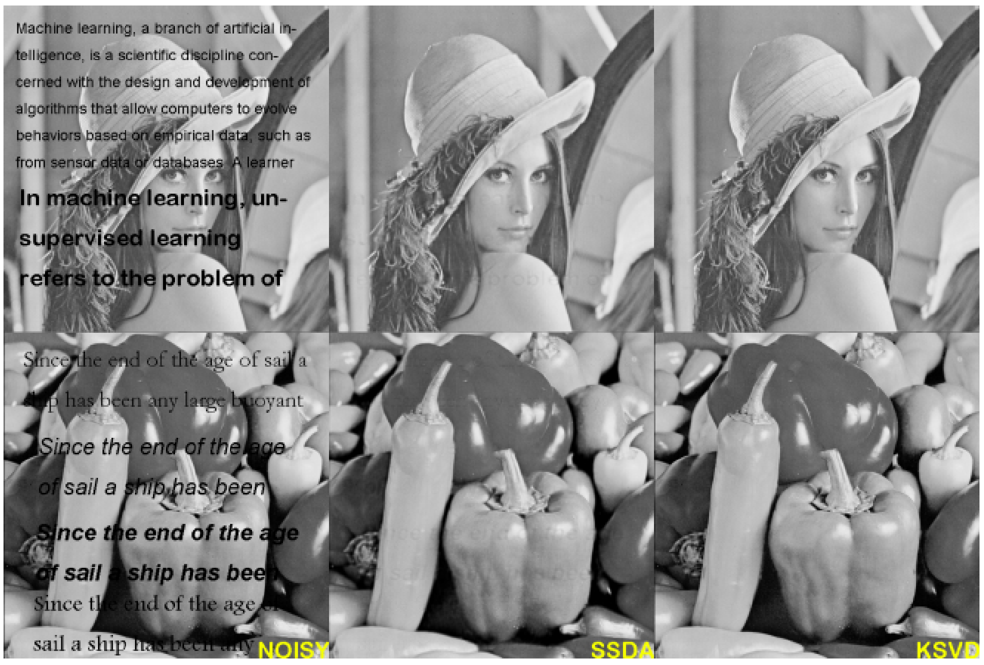


Figure 1.7: The inpainting results of the denoising autoencoder. This figure is borrowed from (3), with the courtesy of Junyuan Xie

2

From Heuristic Optimization to Dictionary Learning: A Comprehensive Review and Comparison of Image Denoising Algorithms

2.1 Introduction and Taxonomy of state-of-the-art denoising methods

Though the topic [1] has been well exploited for many years, image denoising is a critical research area for many vision related applications such as visual recognition, image entertainment, and medical imaging. Due to the increase of image sensors per unit area, camera devices can be interrupted by more noise. Therefore, denoising techniques have become an important step for improving the final visual quality of images (29, 30, 31).

Denoising is the process of reconstructing the original image by removing unwanted noise from a degraded image. It is designed to suppress the noise while preserving as many image structures and details as possible. The main challenge is to design noise

[1]This chapter is based on the following work: L. Shao, **R. Yan,** X. Li, and Y. Liu, "From heuristic optimization to dictionary learning: a comprehensive comparison of image denoising algorithms", *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1001-1013, Jul. 2014.

reduction filters that provide a compromise between these two. Suppose we have such an image formation model as:

$$\mathbf{v}((x)) = \mathbf{u}((x)) + \mathbf{n}((x)), \quad x \in X, X \subset \mathbb{Z}^2 \tag{2.1}$$

where (x) denotes the 2-D spatial coordinates of pixels in an image, (u) is the ground truth, and n indicates the independent additive noise, which we assume is normally distributed with a standard-deviation σ and a zero mean.

Since the Gaussian noise is the most common one encountered in real applications, the additive Gaussian noise is used as the noise model in this chapter.

Generally, image denoising approaches can be categorized as *spatial domain*, *transform domain*, and *dictionary learning based* according to the image representation. Spatial domain methods include local and nonlocal filters, which exploit the similarities between either pixels or patches in an image. Both transform domain and dictionary learning based methods consider transforming images into other domains, in which similarities of transformed coefficients are employed. The difference between them is that transform domain approaches usually represent images with fixed basis functions, but learning-based methods use sparse representations based on a redundant dictionary.

Spatial domain methods attempt to utilize the spatial correlations, which exist in most natural images (32). For a given patch (pixel), a series of candidates will be used in the filtering process. According to the selection of candidates, spatial filters can be categorized as: local and nonlocal filters. A filter is local if the filter support is a spatial neighborhood of the candidate pixels and the filter coefficients are restricted by the spatial distance. A large number of local filtering algorithms have been designed for noise reduction such as Gaussian filter (33), Wiener filter (34), Least Mean Squares filter (35), Trained Filter (TF) (36), bilateral filter (37), anisotropic filtering (38), Steering Kernel Regression (SKR) (39), Metric $\boldsymbol{Q}$ for tuning parameters in SKR (MSKR) (40), and Kernel-based image denoising employing Semi-Parametric Regularization (KSPR) (41). In general, local methods are effective in terms of time complexity. However, when the noise level is high, these methods cannot perform very well because the correlations between neighboring pixels have been corrupted by the severe noise. On the contrary, the nonlocal filters make use of the self-similarity of natural images in a nonlocal manner. Nonlocal Means (NLM) (42) obtains a denoised patch by weighted averaging all other patches in the same image. Since Nonlocal Means (NLM) was proposed, many improvements on it have been developed. Some of them improved the acceleration of NLM (43, 44, 45, 46, 47, 48). Others are for improving the denoising results (4, 44, 49). Recently, a considerable amount of research in image denoising has been shifted from local to nonlocal filters (50, 51, 52, 53, 54). The idea of "nonlocal" has been expanded to transform domain (14) and learning-based methods (1). Even though they are better than local filters for dealing with high noise levels. The major drawback of nonlocal spatial filters is that they still tend to bring artifacts such as over-smoothing the necessary image structures (55).

The second category is *transform domain* methods. The image patches are represented by the orthonormal basis (e.g., wavelets (56), curvelets (57), contourlets (58), and bandelets (59)) with a series of coefficients. The smaller coefficients are the high frequency part of the input image which are related to image details and noise. After the smaller coefficients are adjusted, the reconstructed image could have less noise. In this chapter, wavelet-based denoising methods are discussed as an instance due to its popularity (14, 16, 60, 61). Wavelet-based methods achieve better performance (62) compared to spatial domain methods, because they have superior properties such as sparsity and multiresolution (63). However, the intra-scale and inter-scale correlations of the wavelet coefficients have not been fully explored.

As an emerging machine learning technique, sparse representations have become a trend and have been used for image restoration (39, 64). The general idea of *dictionary learning based* methods in this chapter is that they perform denoising by first learning a redundant dictionary from large amount of image datasets. And then, the denoising is the process of representing noisy patches from an input image with dictionary elements in a linear space. Representative dictionary learning based methods are the K-clustering with Singular Value Decomposition (K-SVD) (17), Learned Simultaneous Sparse Coding (LSSC) (1), and Clustering-based Sparse Representation (CSR) (65). Though most of these methods have achieved competitive performance compared to the state-of-the-art, the sparse model is still computationally expensive (66).

Vladimir *et al.* (67) classified denoising filters according to localnonlocal and pointwisemultipoint. In this chapter, a novel taxonomy of the denoising methods has been proposed (as shown in Fig. 2.1). Learning-based approaches have been introduced as the progress made in the past several years in denoising (66). Furthermore, the comprehensive analysis, comparative evaluation of prevailing classic methods and recent promising techniques will serve as a good reference and provide insights for future research in denoising. The reasons for choosing a method are either that it is representative within a category or it is the best version in the variants of a popular method.

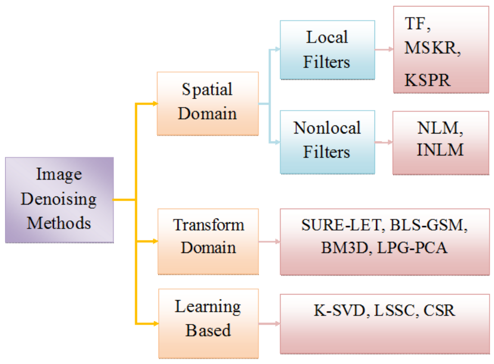


Figure 2.1: Taxonomy of state-of-the-art denoising methods.

2.2 Spatial Domain Methods

Spatial domain filters exploit spatial correlations in images. In this chapter, the spatial filters are classified into two categories: local and nonlocal filters. A filter is local if the candidate selection process used for filtering is restricted by the spatial distance in the same image. A filter is nonlocal if the candidate selection depends only on the similarity and is not restricted by the spatial distance in the same image.

2.2.1 Local filters

Since the Gaussian filter (33) was applied to image denoising, many local filters have been proposed to improve it and provide better edge-preserving ability. The anisotropic filter (38) was designed to remove "the blurring effect in Gaussian filter by smoothing the image only in the direction which is orthogonal to the gradient direction". The method in (68) utilizes the total variation minimization technique to smooth the homogenous regions of the image but not its edges. Similarly, for better edge-preserving, the Smallest Univalue Segment Assimilating Nucleus (SUSAN) filter can average all pixels in the local neighborhood which are from the same spatial region as the central pixel (69). In contrast to the above parametric methods, SKR (39) adapts and expands the kernel regression idea to denoising for removing artifacts. The intuition for developing SKR is: during the filtering process pixels from the same side of the edge would affect each other rather than pixels from different sides of the edge. The recent local filters which produce impressive results are mostly nonparametric or semi-parametric. They are summarized below:

TF Similar to the idea of early local filters (33, 38, 69), a weighted averaging scheme is adopted to perform image denoising in the trained filter (36). The difference is that the trained filter adopts the nonparametric process in which the weights are obtained from the off-line training on a large number of image datasets. In the training step, the classification process ensures best adaptation for local image patterns by changing fixed kernel coefficients into trained coefficients. The classification is based on Adaptive Dynamic Range Coding (ADRC), in which image patterns are encoded as class indexes. In the filtering process, the same classification is applied to each input noisy aperture, and accordingly filter coefficients are obtained from a Look-Up Table (LUT) stored in the previous training process.

The advantage of the trained filter is that the training process is off-line and the LUT only has to be trained once. Thus, the filtering process is so efficient that it can be used in real-time applications. The framework is also applicable to other image enhancement tasks, for example, coding artifact reduction. Moreover, it improves the adaptivity of the local neighborhood filtering by exploring the sparsity in the dataset, which is similar to the learning-based denoising methods. The disadvantage of this method is that ADRC is very vulnerable to severe image degradations (e.g. high noise levels), and same ADRC codes sometimes cannot properly represent same patch textures.

MSKR The aim of the MSKR is to improve SKR from the perspective of parameter selections by maximizing the metric $\boldsymbol{Q}$. The process of the tuning is: 1) The anisotropic patch set of the input noisy image is first detected and serves as the input of the metric $\boldsymbol{Q}$ calculation; 2) The maximization of the metric $\boldsymbol{Q}$ is iteratively carried out and the denoised image also acts as the input of the metric $\boldsymbol{Q}$ calculation; 3) The parameters in SKR are adjusted according to the results of the metric $\boldsymbol{Q}$ optimization.

A no-reference image content metric $\boldsymbol{Q}$, based on Singular Value Decomposition (SVD) of the local image gradient matrix, was proposed in (40):

$$\mathbf{Q} = s_1 \frac{s_1 - s_2}{s_1 + s_2}.$$

Here, s_1 and s_2 are dominant singular values acquired from the singular value decomposition of the gradient matrix $\mathbf{G}$. For a noisy image patch, the estimated values $\hat{s}_1$ and $\hat{s}_2$ can be approximately calculated as (40):

$$\hat{s}_1 \approx \sqrt{s_1^2 + \xi N \sigma^2}, \quad \hat{s}_2 \approx \sqrt{s_2^2 + \xi N \sigma^2}, \tag{2.2}$$

where N is the number of patches in the noisy image, ξ is the parameter for choosing which filter to use, and σ^2 is the local noise variance. The new metric can provide a better measure of the image content compared to previous no-reference metrics such as the Stein's Unbiased Risk Estimate (SURE) and the cross-validation.

This approach has several advantages such as: 1) it has low complexity; 2) it performs well on images which are corrupted by noise models rather than Gaussian (40); 3) The SVD is more robust in the presence of noise compared to most features for noisy images, which is important for image noise level estimation because it can be used as a good texture strength measure (70). The limitation of this algorithm is: it cannot handle images with too many homogeneous regions very well, because the metric $\boldsymbol{Q}$ is only sensitive to structured regions.

KSPR The main idea of this method is to transfer the model between the noisy and its corresponding clean image into a Reproducing Kernel Hillbert Space (RKHS) (41). In this space, the noise-free image can be modeled as a linear combination of the reproducing kernels, which is different from the kernels in SKR (weights in the Taylor approximation). In this model, a property of RKHS, called representer theorem, has been explored. Meanwhile, a semi-parametric variant of the representer theorem has been used to learn the edge models. Thus, the modeling in this algorithm is a L_1 norm optimization problem, with a set of basis functions to model the edge adaptively. The insight of this method is that different regions in an image should be represented by different kernels. In homogenous regions, higher weights are assigned for kernel smoothing. However, in texture regions, the sparse representation based on the basis functions is applied to edges. To distinguish different regions, images are segmented in the pre-processing stage through the mean gradient in each region.

This method can be applied to different additive noise models, especially the impulse noise. The semi-parametric formulation enables the method's good edge-preserving ability. However, when the noise model is additive Gaussian, the denoising result is worse than that of the state-of-the-art methods. Moreover, the mean gradient is not very reliable for measuring the smoothness in the presence of noise, especially Gaussian noise.

2.2.2 Non-local filters

Representative nonlocal filters (50, 51, 52, 53) make use of the weighted averaging idea in a nonlocal manner. In (50), Lebesgue's perspective was proposed to improve Gaussian and median filters. The weights of the filter are determined by the patches whose pixel values are similar to target patches. Kervrann *et al.* proposed a neighborhood filter (51), which is similar to NLM but the weights are determined by the intensity difference of patches. Similar patches are identified not only from spatial distances but also through different orientations and scales in (52, 53).

The original NLM was proposed in (42), and many improvements on NLM have been developed afterwards.

2.2.2.1 Acceleration

From the perspective of refining candidate patches, pre-selection of contributing neighborhoods by calculating mean and gradient values was proposed to accelerate NLM (43). Similarly, local variance (44) and Singular Value Decomposition (SVD) (45) were introduced to eliminate dissimilar pixels. Fast Fourier Transform (FFT) was used to accelerate the weight calculation (46), which makes the algorithm 50 times faster than the original one. The method in (47) (INLM) exploits the symmetry in the weight function, and computes the Euclidean distance by a recursive moving average filter symmetrically, which also considerably improves the efficiency of NLM. Pang *et al.* (48) utilized several critical pixels in the center instead of all pixels in the neighborhood.

2.2.2.2 Improvement of quantitative and qualitative results

Tuning the smoothing parameters was proposed in (44). In (49), a family of non-local image smoothing algorithms were designed, which approximate the application of diffusion Partial Differential Equations (PDE) on a specific Euclidean space of image patches. In order to increase the number of candidates for target patches, the authors in (4) designed a rotationally invariant block matching for NLM. In this section, the original NLM and one of the best variants of NLM are outlined as follows:

NLM The idea of NLM is based on the assumption that every patch in an image has some similar patches within the same image (42). Given a noisy image $\mathbf{v} = \{v_i | i \in \Omega\}$, $\Omega \subset \mathbb{R}^2$, the restored intensity of the pixel $\hat{\mathbf{u}}_i$ is a weighted average of all intensity values within the neighborhood I in the noisy image:

$$\hat{u}_i = \sum_{j \in I} w(i,j) v_j \tag{2.3}$$

The weights can be calculated by (42):

$$w(i,j) = \frac{1}{Z(i)} \exp - \frac{\|v_{N_i} - v_{N_j}\|_{2,a}^2}{h^2}, \tag{2.4}$$

where N_i denotes a patch of fixed size and it is centered at pixel i. The similarity $\|v_{N_i} - v_{N_j}\|_{2,a}^2$ is measured as a decreasing function of the weighted Euclidean distance. $a > 0$ is the standard deviation of the Gaussian kernel, $Z(i)$ is the normalization constant with $Z(i) = \sum_j w(i,j)$, and h acts as a filtering parameter.

NLM is the first filter that considers the self-similarity in the entire image. It is derived from the bilateral filter by the means of adding a weight function to the Euclidean distance between two patches. The nonlocal means filer is also a variation of the Neighborhood filter (51). It substitutes the Euclidean distance in the weight function with a Gaussian as shown in Eq.(2.4). The flip side of NLM is: when the noisy image is short of similar patches within itself, it produces severe artifacts and the performance degrades dramatically, which can be observed in Table 2.2.

INLM Goossens *et al.* (47) improved the original NLM in four aspects: 1) They proposed to record the noise variance at every location in the image and use a post-processing routine to remove the extra noise after the main steps; 2) The whole algorithm is iterative, which assumes that the later iterations could have better grouping results based on the preprocessed results; 3) The Euclidean distance is replaced by the Bisquare robust function, which has improved the similarity term; 4) This INLM algorithm is also applicable to remove noise in color images.

Weight calculation is very important in NLM. As it is compared in (47), "Modified Bisquare" is the most robust loss function to noise. The drawback of this method is that the post-processing local filter tends to blur details though it is effective for removing extra noise.

Discussions on spatial domain methods. Spatial domain methods discussed above can be considered as the variants of the Gaussian filter. Modeling the statistics of natural images as Gaussian distribution is problematic (71), because the local image structures can not be well described by Gaussian. This results in two deficiencies in the Gaussian filter: (1) the filtering weights are not adaptive enough to the image edges; (2) the edges are usually over-smoothed in the denoised images. On the one hand, in order to obtain adaptive weights for a linear filter, TF learns individual weights for various image patches, whose structures are coded by ADRC and saved in a LUT. NLM and INLM exploit the image self-similarity to assign adaptive weights for every patch in the image. On the other hand, SKR and MSKR avoid the denoising artifacts by circumventing the edges when filtering the image. KSPR applies different filters to homogeneous regions and texture regions, which has improved the results of SKR and MSKR.

Local filers can preserve edges well but they rely too much on the texture classification methods (ADRC in TF, the mean gradient in KSPR). Nonlocal filters can achieve better denoising results most of the time except when the image self-similarity assumption fails.

2.3 Transform Domain Methods

Transform domain methods have been researched in the context of image denoising for decades (13). Though there are a large number of variations in this category, such as Discrete Cosine Transform (DCT) , wavelets (13), wedgelets (72), curvelets (73, 74), bandlets (59, 75), contourlets (76), and steerable wavelets (15, 77), wavelets based methods are still dominant.

The wavelets (13, 78, 79) had a strong impact on noise reduction problems. Denoising methods based on wavelets usually transform the image content into multiple sub-bands at different orientations and resolution scales. Large coefficients represent the important low-frequency information. Noise and details exist in the high-frequency subbands. Thus, thresholding and various filters can be applied to the small coefficients. Finally, the image is reconstructed by inverse-transforming these wavelet coefficients back to the spatial domain. Many different kinds of wavelet thresholding methods have been proposed, for example, SureShrink (80) and VisuShrink (81). The correlations of the coefficients within a scale and between different scales have been considered in a few state-of-the-art denoising methods, which are introduced as follows:

BLS-GSM Since critically sampled (16) wavelet coefficients may cause disturbing visual artifacts, overcomplete wavelets are exploited in "the Bayes Least Squares-Gaussian Scale Mixture" (BLS-GSM) to improve this.The procedure for denoising is: 1) The noisy image is transformed into the wavelet domain; 2) It is assumed that the Gaussian Scale Mixture model can be applied to each local neighborhood (16).

$$\mathbf{v} = \sqrt{z}\mathbf{u} + \mathbf{n}, \tag{2.5}$$

where $\sqrt{z}$ is an independent positive scalar random variable, $\mathbf{v}$ is a neighborhood of observed coefficients of the pyramid representation, and $\mathbf{u}$, $\mathbf{n}$ are zero-mean Gaussian vectors. Based on this model, the center of the neighborhood can be estimated as (16):

$$E(u_c|y) = \int_0^\infty p(z|\mathbf{v})E(u_c|\mathbf{v}, z)dz, \tag{2.6}$$

where $E(u_c|\mathbf{v}, z)$ is the local Wiener estimate, and $p(z|\mathbf{v})$ is the posterior density; 3) The last step is to transform the denoised wavelet sub-bands into the spatial domain.

The main contributions of this method are twofold. First, "the full optimal local Bayesian Least Squares solution is computed for estimating coefficients" (16). Second, the vectorial form of the Linear Least Squares (LLS) is exploited in order to take advantage of the information which come from the covariance modeling of the signal and noise.

On the one hand, the pyramidal representation in the local model for spatial neighbors makes this algorithm efficient. On the other hand, BLS-GSM requires an accurate estimation of the original power spectrum density, which makes this algorithm not adaptive (82).

BM3D Inspired by the nonlocal grouping in NLM (the image self-similarity) and the redundant representation in BLS-GSM (the correlations of the wavelet coefficients within a scale and between different scales), Dabov *et al.* (14) proposed a Block-matching and 3D filtering method which achieves remarkable performance. "BM", block matching, is the process that separates the 2-D noisy image patches into the 3-D data groups, in which group patches have similar local structures. "3D" 3-D transform, which includes the 2-D transform (DCT, DFT, or periodized wavelets) within a group and the

1-D transform (Haar wavelets) across groups. BM3D is implemented by two steps. In each step (14), the 3-D transformation of groups, the shrinkage of the transform spectrum, and the inverse 3-D transformation are sequentially performed. The difference between these two steps lies in the ways of shrinking the transform spectrum. In the first step, it is hard thresholding, and in the second Wiener filtering. In each step, the aggregation is performed as weighted averaging filter to fuse the multiple estimates of each patch. The denoising process is illustrated in Fig. 2.2.

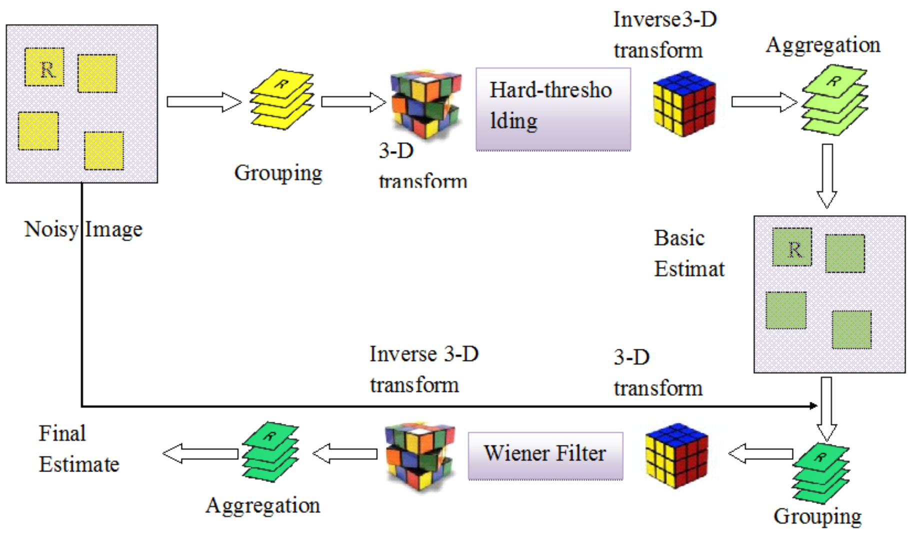


Figure 2.2: Flowchart of BM3D. Each processed block is marked as “R”.

In NLM, better weighted averaging results can be achieved if more reliable candidates for the target patch can be found (the size of the similar patch group is larger) (45). BM3D expands such 2-D self-similarity by exploiting the sparsity between grouped patches. This guarantees sufficient reliable candidates for the target patch. However, when there are unique patches, which have few similar patches in the image, BM3D produces suboptimal results. Also, the wavelet transform has the advantage that noise reduction can be applied to different subbands, which enables adaptive denoising for spatially localized details.

Exploiting the similarity of overlapping patches and the correlation of wavelet coefficients makes BM3D one of the dominant denoising methods. It is proved in (83) that BM3D is approaching optimality when the noise level is not very high. When the noise level is above 40, the denoising performance has a very sharp drop due to the ineffective patch grouping. Although there is a prefiltering technique in BM3D, it cannot improve the grouping (the pre-filter is less effective in such case). This can cause insufficient redundancy in a patch group, which has been considered by other learning-based methods. Meanwhile, the DCT transform will unavoidably cause periodic artifacts because the fixed basis functions of an image representation are not adaptive enough for all kinds of natural images.

LPG-PCA To overcome the drawbacks of the fixed basis functions, the Principal Component Analysis (PCA) was employed in (61) to build adaptive basis functions. In this LPG-PCA framework, each pixel and its nearest neighbors in a noisy image are locally grouped (block matching) into a vector variable. Then, the vector is PCA transformed and in the PCA domain the noise can be removed by shrinkage. The same procedure is repeated in the second stage, which boosts the performance.

The frameworks of LPG-PCA and BM3D are similar. The differences are: 1) The basis functions of the image representations are different. The fixed basis functions (DCT or wavelets) are used in BM3D, which are less adapted to the local geometry of the image to process. LPG-PCA relies on locally data-adaptive basis functions. Therefore, it outperforms BM3D by better preserving fine-grain edges, which are prone to have incorrect nonlocal information in BM3D; 2) In BM3D, the second stage has the 3D groups built with the original noisy image patches using the patch distances from the filtered image in the first round. In LPG-PCA, the input of the second stage is filtered patches from the first stage, and the operations are entirely the same except that the noise levels are different. This actually is not very ideal because the denoising for the first stage might contain errors, i.e., grouping errors due to the noise. The denoising applied on the errors would decay the final denoising accuracy.

Discussion on transform domain methods. Several properties of wavelets have made it the most popular method for transformation (63): 1) Multiresolution: different subbands contain different information, for instance, low frequency or high frequency information. This allows different operations on different subbands to have better precision; 2) Sparsity: due to the multiresolution characteristic, most of the wavelet coefficients are small and only the low frequency parts remain large; 3) Edge detection: large wavelet coefficients usually locate at low frequency subbands and they contain the most important information (e.g. image edges (63)).

Since (80, 81), wavelet domain denoising has become prevalent in the light of the above advantages of wavelet transforms (13, 78, 79). However, transform domain methods also have shortcomings. For instance, DCT is not ideal for representing sharp or distinctive textures, and wavelets cannot represent smooth transitions very well. Since the basis functions for most transform domain methods are fixed, they have difficulties in characterizing natural images with various patterns. Furthermore, the number of coefficients used to represent a image patch is equal to the pixel number of this patch, which tends to result in artifacts such as the ringing artifact. So a redundant dictionary is important for coping with the deficiency of wavelets. For instance, BLS-GSM outperforms previous methods by the redundant dictionary in local spatial neighborhoods. Furthermore, BM3D aims at exploring more redundancy by not just redundancy in the transform domain but also redundancy from the spatial nonlocal grouping of overlapping patches, which accomplishes brilliant denoising results. However, when the nonlocal image self-similarity assumption cannot be guaranteed well, denoising artifacts are unavoidable. In such cases, transforms such as PCA are better choices because they are adaptive to local structures.

2.4 Dictionary Learning Based Methods

Since sparse modeling was proposed by Olshausen and Field (84), training an overcomplete dictionary for the patch representation has been extensively explored in many research fields (85, 86, 87, 88, 89). In the past decade, it has been successfully applied to the field of image denoising (1, 17, 90).

K-SVD (17) achieves good performance by learning a dictionary from noisy image itself with the help of DCT transform as initial dictionaries. Each patch can be represented by a series of patches from the dictionary. Rather than learning a single dictionary for the entire image in K-SVD, classification based on the SKR is exploited in order to avoid patches with different structures being considered as similar patches in locally learned dictionaries (K-LLD) (91). Recently, a structured dictionary learning method (LSSC) has been proposed by Mairal *et al.* (1). The dictionary learning part exploits the local sparsity using a linear combination of elements from the dictionary, and the NLM framework makes use of the nonlocal sparsity within the same input image. It is observed in (67) that nonlocal methods achieve better results than local methods in general. LSSC (1) merges a better representation with the nonlocal framework, and therefore it is slightly superior to the state-of-the-art methods like BM3D at certain noise levels (as shown in the experiment section). The other similar framework was presented by Dong *et al.* (65) for incorporating dictionary learning and sparse codes clustering, which in some cases achieves even better results than LSSC (1). The recent dictionary learning based methods are described below:

K-SVD Early dictionary learning methods have the limitation that they cannot handle images of arbitrary sizes. Similar to the BLS-GSM, Elad *et al.* (17) embedded the local over-complete dictionary into a global Bayes estimator. Compared to previous image priors, image examples (dictionaries) are more adaptive to natural images. The three different dictionaries generated by K-SVD is shown in Fig. 2.3 (17).

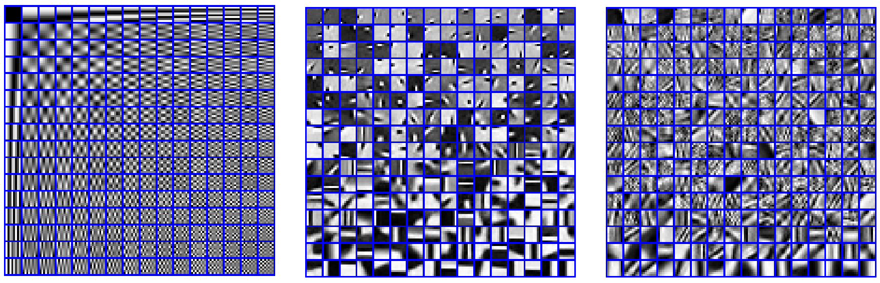


Figure 2.3: Comparison of three different dictionaries generated by K-SVD. From left to right, the images are: (a) DCT dictionary, (b) Dictionary based on natural images, (c) Dictionary based on the noisy image "lena" with $\sigma = 15$. From the figure, one can spot that the dictionary based on noisy images is more adapted to noise. As a more realistic initialization, this dictionary can build the dictionary which has "similar" atoms with the input noisy patches. After several iterations, the dictionary would be updated according to the denoised image from the previous iterations. However, the other two types of dictionaries have their disadvantages of being too distant from the input noisy patches so they might provide initializations which are not optimal. In the experimental results of K-SVD, they proved that this representation produces the best results among three.

The optimization process of K-SVD is iterative, and within each iteration there are two parts:

1) Sparse coding step: the initial dictionary is used for computing sparse approxima-

tions of all patches. The optimization process (17) is as follows:

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^k} \|\boldsymbol{\alpha}_i\|_0 \quad s.t. \|\mathbf{v}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \le \varepsilon, \tag{2.7}$$

where α_i denotes the sparse representation code of each patch $\mathbf{v}_i \in \mathbb{R}^m$, and $\mathbf{D} \in \mathbb{R}^{m \times k}$ is the current dictionary;

2) Dictionary update: in this step the dictionary is updated to improve the quality of the sparse approximation compared to the one used in the first step. For an image, a dictionary adapted to the overlapping patches is learned by (1):

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}} \|\boldsymbol{\alpha}_i\|_0 \quad s.t. \|\mathbf{v}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 \le \varepsilon, \tag{2.8}$$

where $\mathbf{A} = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_n]$ is a matrix in $\mathbb{R}^{k \times n}$, and $\mathcal{C}$ is the set of matrices in $\mathbb{R}^{m \times k}$.

Once, the dictionary is optimized, the output image $\hat{\mathbf{u}}$ can be updated as is described in (17).

K-SVD successfully introduced the idea of example learning into the denoising field, which updates the image representation with a more adaptive model. For a single patch, several similar dictionary atoms could be used for its reconstruction and they will be updated with the information from the noisy image during the approximation process. However, the dictionary used in this method is still unstructured and it requires a considerable number of computations. Due to this computational burden, it is not very likely to use K-SVD for large patches in the denoising process (the size of the dictionary is very limited).

LSSC Though K-SVD builds a redundant dictionary for patch representations, the searching within the dictionary is not very reliable because even a slight change on the input patch might lead to very different dictionary atoms, which is not desirable. Mairal *et al.* (1) proposed a novel improvement to K-SVD by a combination of the NLM framework and modified sparse coding. Considering that similar patches in an image should have similar sparse decompositions, introducing the NLM framework into sparse coding would significantly speed up the process of searching for candidate atoms in an unstructured dictionary. The simultaneous sparse coding can be formulated as (1):

$$\min_{(\mathbf{A}_i)_{i=1}^n, \mathbf{D}\in\mathcal{C}} \sum_{i=1}^{n} \frac{\|\mathbf{A}_i\|_{p,q}}{|S_i|^p}$$
$$s.t. \forall i \sum_{j\in S_i} \|\mathbf{v}_j - \mathbf{D}\alpha_{ij}\|_2^2 \leq \varepsilon_i, \tag{2.9}$$

where S_i represents the clusters after k-means clustering, each of which contains similar patches. D is updated while the denoised image is estimated. $A_i = [\alpha_i j]_{j\in S_i} \in \mathbb{R}^{k\times|S_i|}$.

This clustering before sparse coding significantly speeds up the whole algorithm, and it guarantees that each patch in the current image can exploit the sparsity in the cluster or the training dataset. The output of each denoised pixel is calculated as a weighted average in each cluster:

$$\hat{\mathbf{u}} = diag(\sum_{i=1}^{n}\sum_{j\in S_i} \mathbf{R}_j \mathbf{1}_m)^{-1} \sum_{i=1}^{n}\sum_{j\in S_i} \mathbf{R}_j \mathbf{D}\boldsymbol{\alpha}_{ij}, \tag{2.10}$$

where $\mathbf{R}_j$ in $\mathbb{R}^{m\times n}$ is the binary matrix which can obtain the patch according to its index from the image. $\mathbf{1}_m$ is a vector composed of ones and its size is m.

LSSC slightly outperforms BM3D by exploiting adaptive basis functions for image representation. It can solve the issue that the unique patches in an image cannot have enough patches in the same image that are similar to them. It also achieves better results compared to K-SVD because: 1) LSSC includes a patch clustering process, which exploits the sparsity from the image self-similarity; 2) Only l_0 norm is used in both the learning and reconstruction processes. This method has been applied to other restoration applications such as image deblurring (92), which indicates the general applicability of this sparse coding model. However, LSSC still has the similar nonhierarchical dictionary as K-SVD, which is very prone to the reconstruction artifacts.

CSR Clustering-based sparse representation (65) was designed to incorporate dictionary learning with structural clustering to exploit two kinds of sparsity. The sparse codes are encoded with respect to the average, which builds up an connection between clustering and sparsity. The optimization process can be described as (65):

$$\boldsymbol{A} = \arg\min_{\boldsymbol{A}} \frac{1}{2}\|\mathbf{v} - \mathbf{D}\boldsymbol{A}\|_2^2 + \lambda_1\|\boldsymbol{A}\|_1$$
$$+\lambda_2 \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k} \|\boldsymbol{\alpha}_i - \boldsymbol{\beta}_k\|, \tag{2.11}$$

where $\boldsymbol{A}$ is the sparse matrix, $\boldsymbol{\beta}$ denotes sparse coefficients for the centroid vectors, $\mathbf{v}$ is the noisy image, and $\mathbf{D}$ is the learned redundant dictionary. K is the number of the clusters, and $\mathcal{C}_k$ is the number of elements in each cluster.

The first difference between CSR and LSSC is that CSR does not need any initial dictionary (K-SVD adopts DCT as the initial dictionary; LSSC uses a learned dictionary from high quality image datasets as the initial dictionary). The pure online training process is performed on the input noisy image and the dictionary is updated by k-means clustering and PCA. The second difference between CSR and LSSC is: l_1 norm is used in CSR to characterize nonlocal sparsity rather than l_2 norm.

Discussions on dictionary learning based methods: Over-complete dictionaries learned from clean or noisy image patches provide adaptive representations for image denoising. Earlier sparse coding based methods (e.g., K-SVD) search for the optimal decomposition of a patch in the whole dictionary while updating the dictionary with the information from the input. Though most of them (17) (91) managed to achieve satisfactory denoising results, they have a disadvantage that similar patches might have very different sparse decompositions (17). LSSC improves this situation by applying clustering in sparse decompositions. However, its performance largely depends on the initial dictionary trained offline on high quality images and the nonlocal grouping results. Later, CSR uses a similar framework but reduces the computational complexity significantly.

One major shortcoming of this whole category is the computational complexity. First, these algorithms always undergo several denoising iterations. For example, l_0 norm related optimization is Non-deterministic Polynomial-time (NP) hard. Second, their dictionaries are unstructured, which further causes a computational burden in the form of full search in the dictionary during the sparse decomposition (17) (90) (65). For instance, if the size of the dictionary is K and the times of iterations are n, then the computational complexity is $O(nK)$, which is very inefficient. More comparisons on computational time can be seen in Table. 2.1.

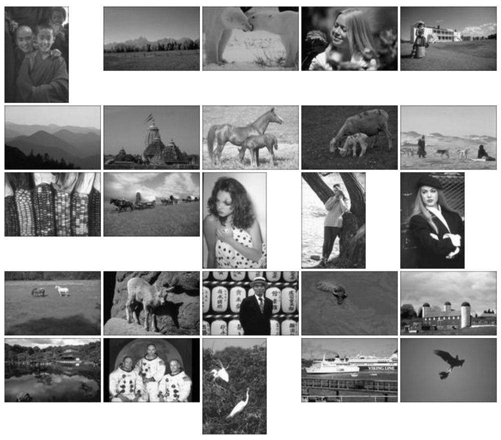


Figure 2.4: The sample images from the Berkeley Dataset.

2.5 Performance Comparison Of Representative Image Denoising Methods

2.5.1 The Image Database

2.5.1.1 Source Image Content

The image database contains various source images with diverse image content. These images include pictures of human faces, natural scenes, and man-made objects. Most of them are extensively used by researchers in the field of image denoising. The first dataset (200 images) we use is the Berkeley segmentation dataset [1]. The resolution of these images is either 481×321 or 321×481. The other dataset we employ contain the standard test images [2] as shown in Fig. 2.5. The size of these images is either 512×512 or 256×256. Moreover, some large images are selected from flickr[3] to test the scalability of all the algorithms. Typically, we used test images with the resolution of 2048×1361, 721×1024, and 800×600.

2.5.1.2 Image Degradation Model

Additive White Gaussian noise with standard deviations $\sigma = 10, 20, 25, 35, 75$ was added to the testing images (We cannot show the full results here due to the space

[1]http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[2]The 8 images are: " Man", "Squares", "Barbara", "House", "Peppers", "Lena", "ZeldaG", "Cameraman"

[3]http://www.flickr.com/

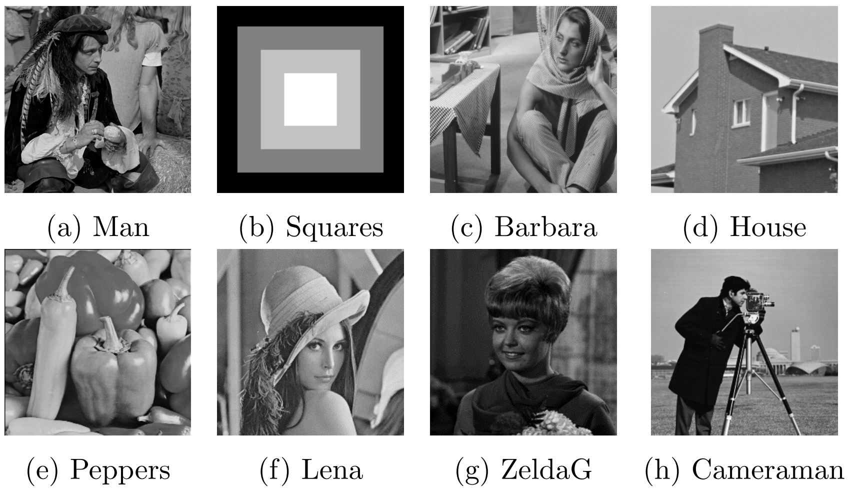

(a) Man (b) Squares (c) Barbara (d) House (e) Peppers (f) Lena (g) ZeldaG (h) Cameraman

Figure 2.5: The standard testing images used in the experiment.

limit). Fig.2.9 (b), Fig.2.10 (b), and Fig. 2.11 (b) illustrate the distortion on an image after adding Gaussian noise.

2.5.2 Test Methodology and Discussion

In this section, several criteria have been used to compare the above methods: Peak Signal to Noise Ratio (PSNR)(mathematical distance formulations), method noise (10), the Structural SIMilarity (SSIM) (93) (structural), Visual Information Fidelity (VIF) (94), and visual quality of the reconstructed images.

The algorithms are selected to provide a comprehensive comparison among different categories. From the spatial filters, TF (36), MSKR (40), KSPR (41), NLM (42), and INLM (47) are chosen. Among transform domain methods, BM3D (14), BLS-GSM (16), and LPG-PCA (61) are included. In the dictionary learning based category, we select K-SVD (17), LSSC (1), and CSR (65). The implementations by the respective authors are used for all experiments. However, as all the codes are written in MATLAB and run very slowly, we revised the main denoising functions into C and compiled them (MSKR, NLM, CSR) into Mex functions. This helps the later time complexity comparison in the following section.

2.5.2.1 Quantitative Comparison- PSNR and SSIM comparisons

PSNR is employed to provide quantitative evaluations of the denoising results. PSNR is defined as:

$$PSNR = 10 \log_{10}(\frac{L^2}{MSE}), \tag{2.12}$$

where L is the dynamic range of the image and MSE is the mean squared error between the original and the reconstructed image.

SSIM (93) is a quality metric more correlated to human perception. SSIM is calculated within local windows using:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{2.13}$$

$C_1 = (K_1L)^2$, $C_2 = (K_2L)^2$ are constants to avoid instability when ${\mu_x}^2 + {\mu_y}^2$ or ${\sigma_x}^2 + {\sigma_y}^2$ are too close to zero (93). $K_1 \ll 1$, $K_2 \ll 1$ are small constants, and L is the dynamic range of the intensity values. σ_x and σ_y are standard deviations. μ_x and μ_y are the mean intensities.

All methods have been tested with the optimal parameters mentioned in the original papers. From the quantitative results, one can see that, in spatial domain, the overcomplete kernel based method (KSPR) outperforms the other local filters in high noise levels. One can also see that, image self-similarity is a sparse model for most natural images, which makes nonlocal methods perform better than local counterparts. When the image contains enough self-similarity (e.g. "squares"), NLM can be comparable to BM3D and the dictionary learning based techniques. However, when the standard deviation of the Gaussian noise is high, the performance of NLM drops dramatically due to the difficulty of utilizing self-similarity. The pre-processing in INLM overcomes this drawback, and INLM is the best performing one among the spatial domain methods. In wavelet domain, BLS-GSM adopts overcomplete wavelets and improves over the critically sampled wavelet method (e.g. SURE-LET). BM3D employs not only overcomplete basis functions but also nonlocal grouping, which contributes to their better performance. LPG-PCA tends to generate better results than BLS-GSM when the noise level is low, for instance, 20, and when the image contains many repetitive patterns (e.g., Squares). This is due to the fact that the nonlocal clustering requires the structures in an image not totally corrupted. BLS-GSM outperforms LPG-PCA in high noise levels because the multiresolution structures in wavelets are still functioning. In the dictionary learning based category, the nonlocal grouping and the redundant dictionary in LSSC make it the best performing method, and slightly outperform BM3D. K-SVD is inferior to LSSC because its dictionary is a global model and the sparsity in the image itself is not adopted. At almost all noise levels, CSR is better than BM3D when the image contains a lot of repetitive patterns. This is because the iterations in CSR yield better grouping on the results of the previous denoised images.

The results of SSIM (shown in Table 2.2) are mostly consistent with those of PSNR (shown in Table 2.2). However, there is an exception that: the SSIM of the denoising results from spatial filters drops more dramatically than PSNR in high noise levels.

Method noise comparison The method noise (10) was designed to show how much detail has been removed during the denoising process. Let U be the noisy image and $\hat{V}$ the denoised image. Then, the method noise is defined as the image difference (10):

$$n(\hat{U}, U) = U - \hat{V}. \tag{2.14}$$

With this metric, it is easy to subjectively evaluate whether a method has removed too many structures from the input image. Fig. 2.7 displays the method noises of different algorithms when the standard deviation of the Gaussian noise is 20. The fewer details we can see in the method noise images, the better details have been preserved in the denoised images. The brief comparison is listed blow:

- In images processed by MSKR, strong edges are not well preserved because metric Q only takes structured regions into consideration.
- KSPR loses too many details in structured regions, and strong edges are not preserved very well. KSPR is supposed to solve different optimization problems within different regions (strong edges or smooth regions). The problem exists in the computing of mean gradients in the noisy image, which will be easily affected by the Gaussian noise.
- The trained filter has the similar problem as KSPR, but it is slightly better than KSPR in smooth regions. Though they are both local filters, the trained filter

is more adaptive because the filtering process exploits local sparsity by adaptive training on a natural image dataset.

- Compared to local spatial filters, SURE-LET and BLS-GSM preserve edges better, but some details have been lost in smooth regions. In the result of BM3D, we can barely see edges or details in smooth regions. This shows the strength of the nonlocal grouping.

- It is easy to observe that the method noises of all the best algorithms (BM3D, LSSC, CSR, K-SVD) resemble the Gaussian noise.

Summary: From the above results, several conclusions can be drawn: 1) nonlocal methods are better than local methods in quantitative results, because image self-similarity is more helpful to average out the additive Gaussian noise compared to local kernels or polynomial approximations (NLM outperforms local methods); 2) adaptive basis functions in a transform are better than the fixed basis for representing an image (LSSC sometimes outperforms BM3D); 3) multiresolution representations are better than single resolution ones (BM3D is better than LPG-PCA in most cases); 4) over-complete image models are better (KSPR is better than MSKR at high noise levels); 5) The combination of the three (1), 2), 3)) makes the state-of-the-art algorithms (LSSC, CSR, BM3D). These conclusions we obtain are in line with the previous qualitative analysis in Sec. 2.4.

As it is mentioned in (10), good quantitative results do not guarantee good visual quality of the reconstructed images. So in real applications, the visual quality is still an important metric.

2.5.2.2 Subjective quality comparison

VIF is an criterion highly correlated with human visual quality for image quality assessment. The whole VIF is built on Natural Scenes Statistics (NSS), distortion, and Human Visual System (HVS) modeling (94). As is shown in Sheikh's evaluation work (95), VIF performs the best among all the image quality metrics.

$$VIF = \frac{\sum_{j \in subbands} I(\overrightarrow{C}^{N,j}; \overrightarrow{F}^{N,j} | s^{N,j})}{\sum_{j \in subbands} I(\overrightarrow{C}^{N,j}; \overrightarrow{E}^{N,j} | s^{N,j})} \tag{2.15}$$

where $\overrightarrow{C}^{N,j}$ represent N elements of the random field that describes the coefficients from subband j. $s^{N,j}$ is the random field of positive scalars. $\overrightarrow{E}^{N,j}$ and $\overrightarrow{F}^{N,j}$ represent the visual signal at the output of the human visual system model.

Fig. 2.6 depicts the comparison of VIF results using different methods. Compared to the previous quality metrics, VIF is the closest to human perception. For example, the average PSNR and SSIM of KSPR is higher than those of TF and MSKR. But if we examine the visual results in Fig. 2.9, it can easily be seen that there are more artifacts in KSPR than in TF and MSKR.

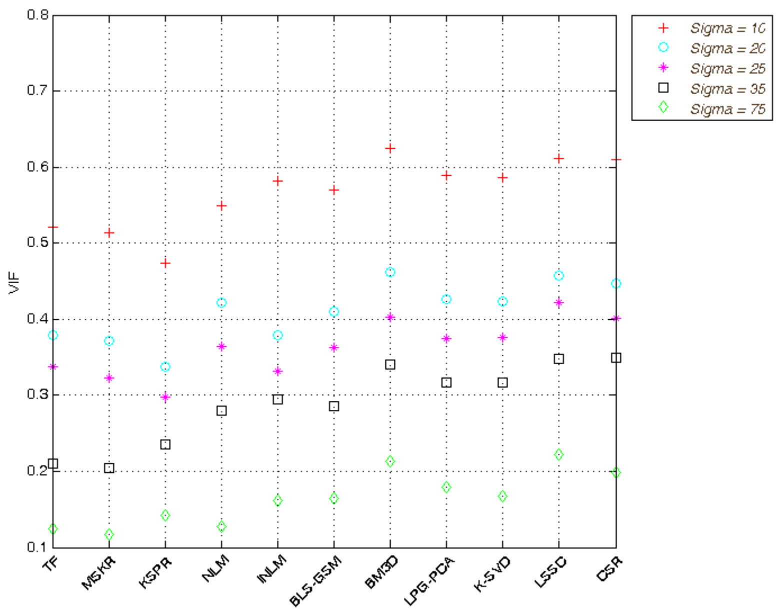


Figure 2.6: Average VIF comparison.

In our experiments, the differences in **visual quality** between the various denoising methods can be inspected in the examples shown in Fig. 2.9, Fig. 2.10. From those figures, one can observe that in spatial domain methods local filters achieve good visual results but still blur the image details too much. Among them, KSPR keeps lots of details but introduces some artifacts. MSKR fails to denoise images corrupted by high level noises because the intrinsic idea of SKR is to denoise according to the direction of the edge. When high level noises destroy the edges of images, MSKR cannot perform well. Among all the local spatial filters, at low noise level, TF generates the best visual results because of its adaptive representations. Nonlocal means has the best visual results among spatial domain methods because it fully exploits the sparsity in an image. In wavelet based methods, BLS-GSM can effectively remove noise but the artifacts are quite noticeable. LPG-PCA does not introduce new artifacts but the entire image is a bit oversmoothed. BM3D is the best among them that can well retain edges and details. However, when the noise level is above 50 (Fig. 2.10), BM3D generates relatively good results but brings significant amounts of artifacts. The details and edges are well preserved in the results of dictionary learning based methods, even though the use of the sparsity constraint in the regularization term causes ringing effects at some noise levels.

Summary: In most cases, higher quantitative results yield better visual results. However, 1) In terms of the edge preserving ability, dictionary learning methods achieve best results because dictionary atoms usually capture the most distinctive features in an image, e.g. image edges, etc. In this way, image reconstruction based on these atoms could improve the restoration results; 2) For certain textures (oscillatory patterns), wavelet based methods work better than the other two categories. For instance, the separation of diagonal orientations in BLS-GSM helps to remove the noise in diagonally oriented image regions; 3) Regarding the highly corrupted images, learning-based methods work better. This shows even more obviously in visual results than the quantitative results. The reason is that it is very difficult to reconstruct clearer images from very limited information from the highly corrupted input noisy images. However, the learning-based methods can provide dictionaries learned from various high quality image patches, which can provide a better source for image reconstruction; 4) Basis functions vary from pixel to pixel are helpful to preserve fine details in images compared to global models and models learned for each cluster. This is not shown in those PSNR or SSIM values directly. This is because pixel-based basis functions are more adaptive to image details.

2.5.3 Execution Time

The computation time of representative denoising methods are also evaluated in this section from a practical point of view. Our experiments were carried out on a workstation with a 3.0-GHz Intel(R) 2 Core CPU. The performance is shown in Table 2.1. The computation time has been averaged over twenty runs.

Table 2.1: Relative computation time of representative denoising algorithms (in minutes). Although the difference in computation time is about a factor of 1000 to 10000, the comparison is still important because it shows the trend that more high-performance denoising methods consume much longer time compared to the low-performance ones. The compromise we need to make between complexity and efficiency should be taken into consideration when we design a new denoising algorithm. This table can demonstrate an intuitive idea of how expensive those algorithms are.

Method	2048×1361	721×1024	800×600	512×512	481×321	256×256
TF	0.930	0.264	0.171	0.094	0.056	0.011
MSKR	117.55	37.22	30.04	20.93	12.85	4.37
KSPR	321.23	92.16	59.95	29.89	24.70	12.15
NLM	56.73	14.88	7.36	3.21	1.85	1.07
INLM	11.426	3.054	1.959	1.129	0.554	0.20
BLS-GSM	2.853	0.723	0.496	0.237	0.167	0.061
BM3D	0.973	0.278	0.125	0.085	0.038	0.017
LPG-PCA	160.35	40.89	21.74	12.47	7.24	3.12
K-SVD	1.168	1.033	1.792	1.347	1.331	1.568
LSSC	188.36	39.42	34.57	13.16	10.75	4.03
CSR	140.25	49.89	36.65	19.09	12.07	8.44

Compared to the critically sampled wavelet transforms, the dictionary learning based overcomplete representations make algorithms less efficient (to the extent of hundreds of times slower). Also, the local spatial filters shown in the table have lots of iterations, which makes them comparatively slow. Among all the methods discussed, trained filter is almost the most efficient one (similar to BM3D) because it has a thorough off-line training process which reduces the actual online denoising burden. But it does increase the demands for the memory.

Summary: Under the same image resolution, the most efficient denoising method in our evaluation is TF. The reason is that the offline learning in this algorithm accomplishes most of the denoising task to save time in the online testing. Then, the next efficient algorithms are transform domain methods (BM3D, BLS-GSM) due to the multiresolution structure of wavelets. All the other methods which involve iterations run quite slowly (MSKR, LSSC, CSR). Also, the pixel by pixel model selection is very time-consuming as well (KSPR, LPG-PCA). The nonlocal methods in the spatial domain are also expensive because of their high searching complexity.

Considering the scalability of all approaches, most methods, such as TF, BLS-GSM, and BM3D, scale linearly with the increase of the image size. However, LSSC, LPG-PCA, and CSR slow down dramatically due to that the optimization of the regularization is highly affected by the image size.

2.6 Conclusion

In this chapter, we reviewed and compared representative denoising methods both qualitatively and quantitatively. These methods have been divided into three categories: spatial domain, transform domain and dictionary learning based. Extensive experiments were conducted to evaluate the performance of all the algorithms.

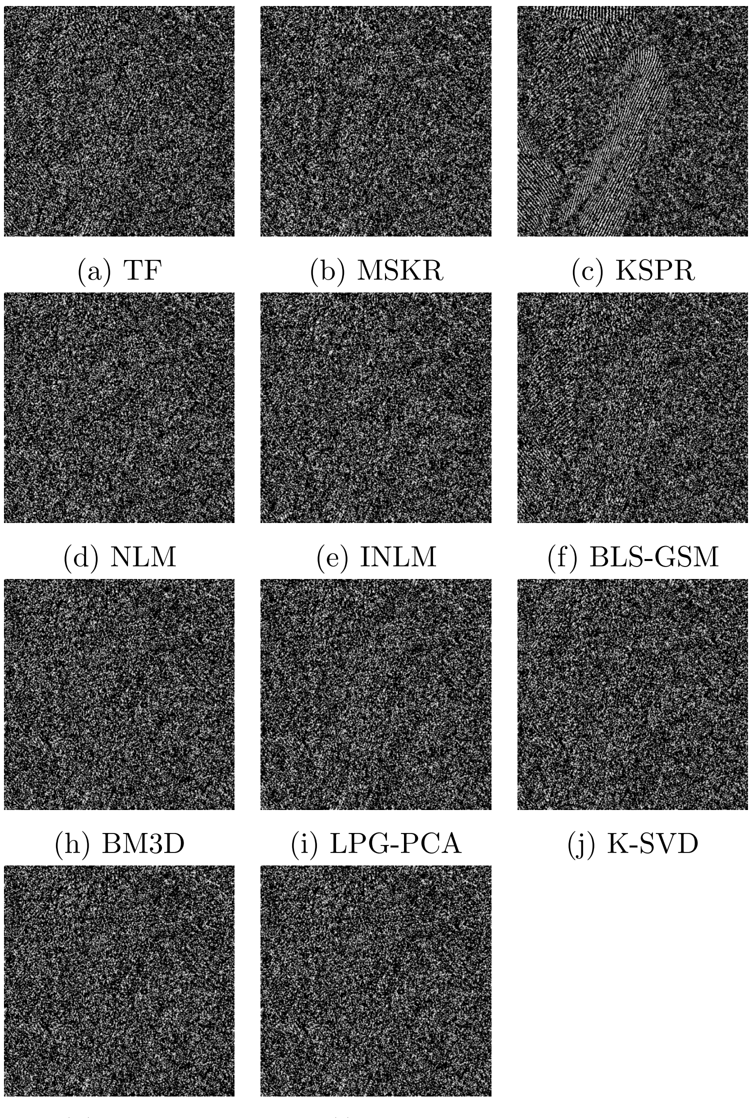
Through analytical comparison, it was found that image representations with overcomplete basis functions improve the performance within each category. In spatial filters, KSPR improves kernel regression based methods in high noise levels. In the transform domain, overcomplete wavelets are used in BLS-GSM to overcome the shortcomings of critically sampled wavelets. In dictionary learning based algorithms, it has been proved that the redundant dictionary based K-SVD outperforms the DCT based K-SVD (17). In general, overcomplete basis functions are more adaptive to image contents, which can bring better denoising results. The major disadvantage of overcomplete representations is that they usually result in computational burden. Another interesting trend observed in these results is the importance of nonlocal grouping. In each category, the performance of the methods with nonlocal grouping is significantly better than that of the methods without nonlocal grouping. For instance, NLM outperforms TF, BM3D surpasses BLS-GSM, and LSSC enhances K-SVD. In addition, adaptive basis functions in image representations contribute better to edge-preserving, which has been proved in our evaluation that dictionary learning based methods generally produce better visual results. Moreover, multiresolution structures in the transform domain benefit the edge/detail preserving.

It is clear from the comparison in this chapter that all three categories are important denoising techniques for various applications. In applications that require high efficiency, some of the local spatial filters or transform domain filters are more appropriate, because nonlocal spatial filters lead to high searching complexity. If the memory and complexity were not a major concern for the users, dictionary learning based methods would be more applicable because the online training and iterations are not practical in real time systems but they significantly boost the performance. Moreover, algorithms contain multiresolution structures tend to be more efficient than single resolution ones.

Dictionary learning based methods have produced the competitive denoising results compared to the state-of-the-art both objectively and subjectively so far. Therefore, the future research in denoising can be focused on improving the applicability of the learning-based techniques in the following aspects: 1) more accurate sparse decompositions; 2) more structured dictionaries; 3) more efficient optimization processes.

Table 2.2: PSNR and SSIM Comparison. In each row of this table, the upper one is PSNR (dB) value and the other one is SSIM value. All the results reported are average values over 5 experiments, having different realizations of the noise.

Image	Noise	TF	MSKR	KSPR	NLM	INLM	BLS-GSM	BM3D	LPG-PCA	K-SVD	LSSC	CSR
Man		27.16	28.62	28.5	28.94	29.24	29.62	30.1	29.62	29.62	**30.27**	30.09
	$\sigma = 20$	0.8102	0.78	0.889	0.9023	0.8992	0.8184	**0.9208**	0.9056	0.9057	0.9201	0.8325
		22.68	20.35	26.04	24.03	26.67	27.05	27.54	26.89	26.96	**27.57**	27.47
	$\sigma = 35$	0.5222	0.4228	0.7983	0.791	0.8136	0.7241	**0.8573**	0.828	0.825	0.8552	0.7443
		12.99	12.13	21.48	14.82	23.51	23.96	**24.46**	23.57	23.54	24.28	24.38
	$\sigma = 75$	0.3844	0.1083	0.5721	0.4376	0.6681	0.5798	**0.73**	0.679	0.6753	0.7156	0.6138
Squares		29.96	31.43	29.4	33.82	39.87	38.24	44.55	41.88	40.55	**45.59**	45.44
	$\sigma = 20$	0.6979	0.6764	0.6454	0.7723	0.9544	0.9469	0.9749	**0.9818**	0.9515	0.9809	0.9737
		24.18	22.05	26.62	25.67	34.41	33.9	39.33	36.1	35.42	**41.93**	39.66
	$\sigma = 35$	0.3435	0.2726	0.4764	0.388	0.9009	0.8912	0.9422	0.9469	0.9067	**0.9621**	0.9408
		12.37	12.59	21.11	14.94	27.34	28.38	32.02	28.61	28.32	**34.25**	31.42
	$\sigma = 75$	0.0885	0.0596	0.2932	0.0994	0.6986	0.7699	0.874	0.7942	0.7502	**0.903**	0.8254
Barbara		28.74	29.57	25.52	30	31.22	29.34	**32.08**	31.66	31.11	31.82	32.15
	$\sigma = 20$	0.8191	0.8686	0.8434	0.928	0.9433	0.8478	**0.9509**	0.9454	0.9378	0.9471	0.9055
		24.59	19.75	23.94	24.31	28.21	26.18	29.16	28.36	27.73	28.96	**29.25**
	$\sigma = 35$	0.6219	0.3969	0.743	0.7971	0.8868	0.7395	**0.9078**	0.8931	0.8704	0.9064	0.8526
		11.83	11.82	20.89	14.83	24.16	23.27	**25.15**	24.19	23.2	25.14	25.12
	$\sigma = 75$	0.3592	0.1154	0.5217	0.4089	0.7485	0.5786	**0.7916**	0.7575	0.6951	0.791	0.7153
house		29.49	32.48	30.24	31.34	32.85	32.54	33.77	33.07	33.16	**34.11**	33.86
	$\sigma = 20$	0.7011	0.8514	0.7885	0.7742	0.8583	0.8511	0.8721	0.8671	0.861	**0.8844**	0.8737
		24.92	20.15	27.23	24.95	30.31	30.08	31.38	30.42	30.32	**31.67**	31.4
	$\sigma = 35$	0.4116	0.2943	0.6838	0.4553	0.8109	0.8037	0.8365	0.825	0.8127	**0.8407**	0.839
		11.3	11.77	22.56	15.01	25.54	26.42	27.51	26.18	25.53	**27.75**	27.3
	$\sigma = 75$	0.1168	0.0789	0.4911	0.1332	0.6747	0.7016	0.764	0.7183	0.6849	**0.7792**	0.7675
peppers		28.65	30.07	27.38	29.55	26	30.57	31.29	30.54	30.76	**31.37**	31.18
	$\sigma = 20$	0.7914	0.874	0.8408	0.8026	0.8583	0.87	**0.8863**	0.874	0.8741	0.8833	0.8829
		23.29	19.76	25.53	24.17	24.61	27.83	**28.52**	27.69	28	28.49	28.37
	$\sigma = 35$	0.4308	0.3584	0.7547	0.5208	0.802	0.806	0.8335	0.8175	0.822	0.8311	**0.8336**
		12.37	11.86	21.71	14.72	22.16	24.07	**24.73**	23.52	23.84	24.66	24.5
	$\sigma = 75$	0.1643	0.1089	0.5602	0.1722	0.6755	0.6791	**0.7364**	0.697	0.6926	0.7278	0.7353
lena		28.06	29.51	28.02	29.15	29.93	29.73	30.43	30.02	29.92	**30.51**	30.42
	$\sigma = 20$	0.7032	0.8446	0.799	0.7895	0.8508	0.8403	0.861	0.853	0.8456	**0.8626**	0.8572
		23.82	19.59	25.82	24.07	27.28	27.11	27.81	27.21	27.19	**27.9**	27.82
	$\sigma = 35$	0.3817	0.3633	0.7093	0.5204	0.7765	0.7667	0.7994	0.783	0.7729	**0.8005**	0.7986
		10.16	11.78	21.89	14.8	23.85	23.94	**24.65**	23.62	23.54	24.6	24.62
	$\sigma = 75$	0.1669	0.1173	0.5172	0.1801	0.6355	0.6382	0.691	0.6516	0.6238	0.6891	**0.6945**
Zelda		29.48	31.22	30.43	30.52	31.46	31.83	**32.3**	31.68	31.62	32.06	32.13
	$\sigma = 20$	0.7071	0.8491	0.8136	0.7724	0.8542	0.8583	**0.8736**	0.8615	0.8536	0.8647	0.8683
		25.17	22.21	27.32	24.71	28.75	29.32	**29.75**	28.99	29.11	29.58	29.66
	$\sigma = 35$	0.4156	0.4587	0.6933	0.4459	0.7764	0.79	0.8124	0.7919	0.7865	0.8068	**0.8126**
		11.88	12.42	22.31	15.03	25.69	26.11	**26.59**	25.65	25.71	26.09	26.54
	$\sigma = 75$	0.0577	0.0653	0.4825	0.1157	0.6406	0.6661	0.7159	0.6641	0.6641	0.7005	**0.718**
Cameraman		28.01	28.65	26.4	28.9	29.42	29.58	30.48	29.71	29.96	**30.59**	30.56
	$\sigma = 20$	0.7183	0.8428	0.792	0.7744	0.854	0.8466	0.8749	0.8574	0.8621	**0.8768**	0.8586
		23.39	20.19	24.86	24.14	27.11	26.96	27.93	27.11	27.36	27.92	**28.56**
	$\sigma = 35$	0.4293	0.4246	0.694	0.4893	0.788	0.776	0.8211	0.7981	0.7949	**0.8264**	0.7996
		11.73	12.02	21.25	14.97	23.34	23.43	24.33	23.46	23.64	24.41	**25.11**
	$\sigma = 75$	0.2404	0.2122	0.4909	0.3145	0.6447	0.7194	**0.7818**	0.6767	0.7438	0.732	0.5948
Average		21.51	20.92	25.27	23.22	28.04	28.31	29.83	28.74	28.59	**30.06**	29.88
		0.4618	0.4352	0.6622	0.5327	0.7922	0.7712	**0.8379**	0.8112	0.8005	0.8369	0.8058



(a) TF (b) MSKR (c) KSPR

(d) NLM (e) INLM (f) BLS-GSM

(h) BM3D (i) LPG-PCA (j) K-SVD

(k) LSSC (l) CSR

Figure 2.7: The comparison of method noise.

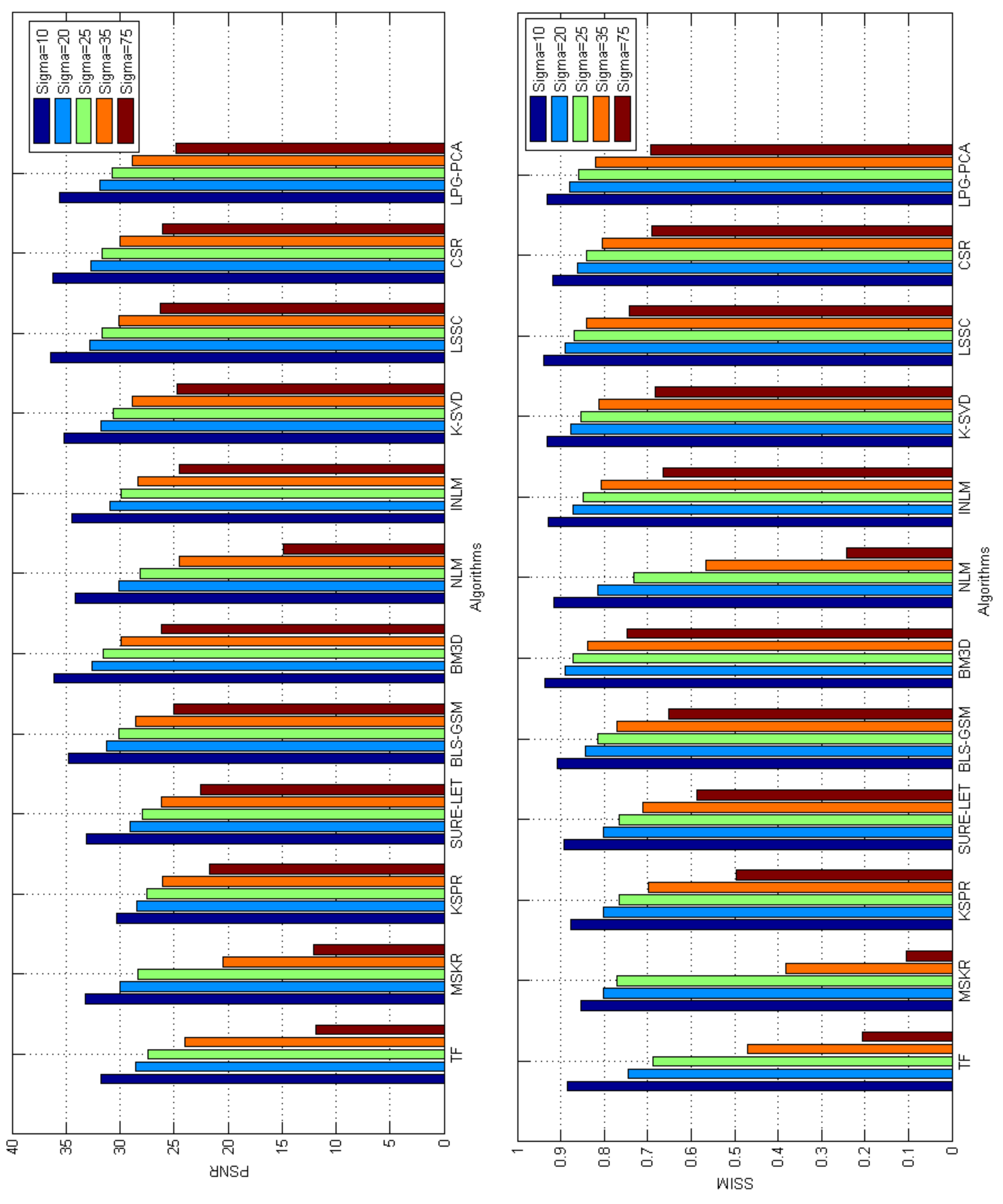


Figure 2.8: Average PSNR and SSIM comparison.

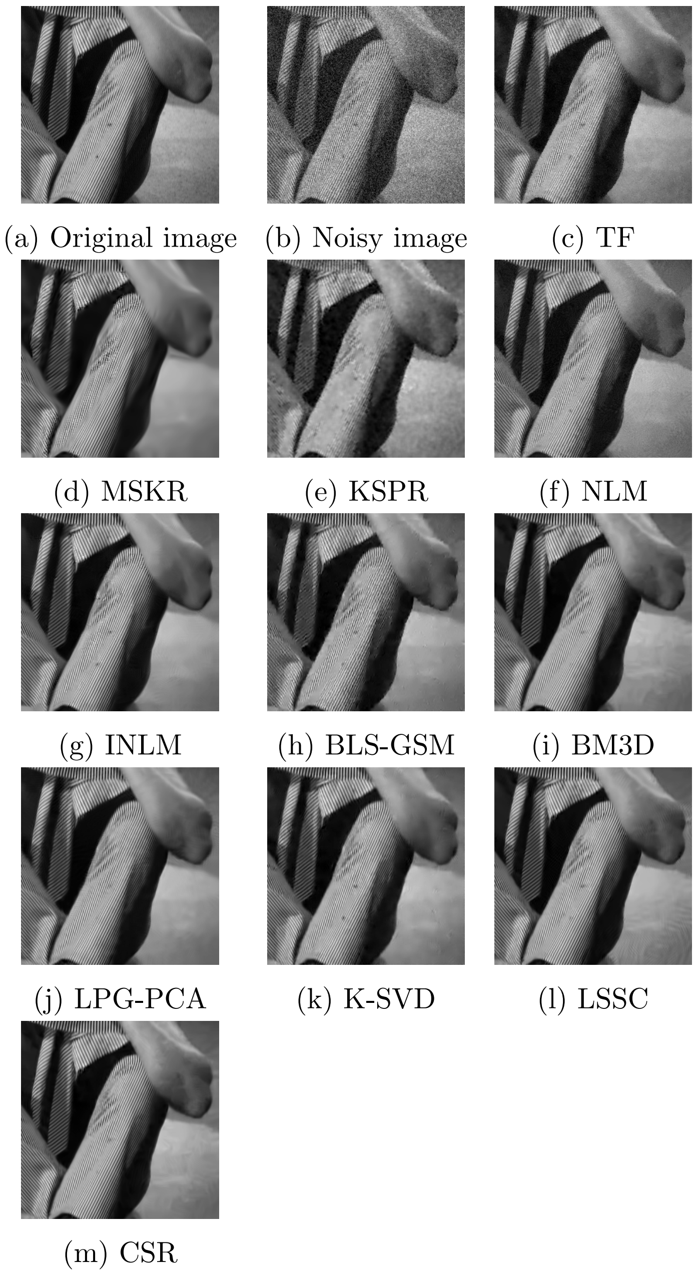
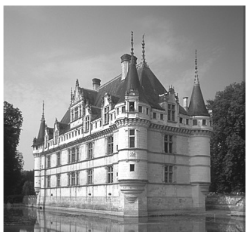
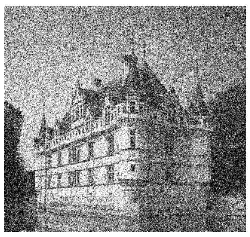
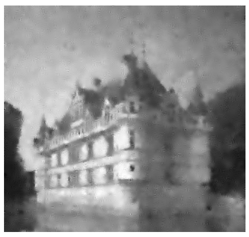
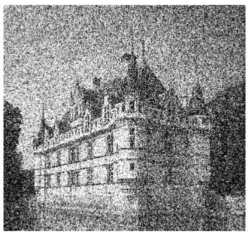


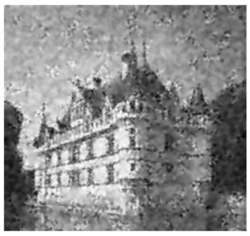
Figure 2.9: Denoising results on "Barbara" with $\sigma = 20$.

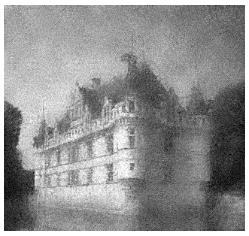

(a) Original image

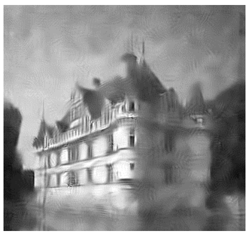

(b) Noisy image

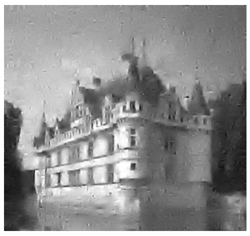

(c) TF

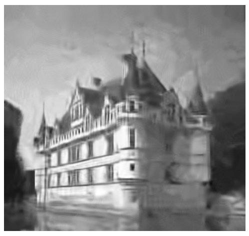

(d) MSKR

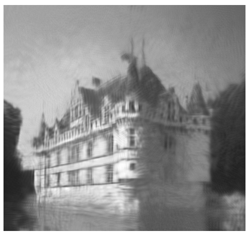

(e) KSPR

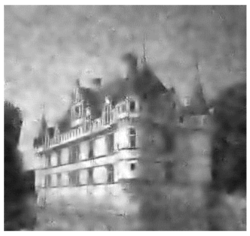

(f) NLM

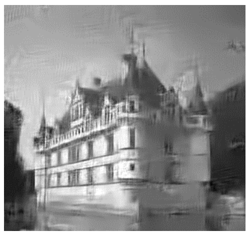

(g) INLM


(h) BLS-GSM


(i) BM3D


(j) LPG-PCA


(k) K-SVD


(l) LSSC

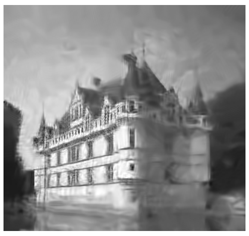
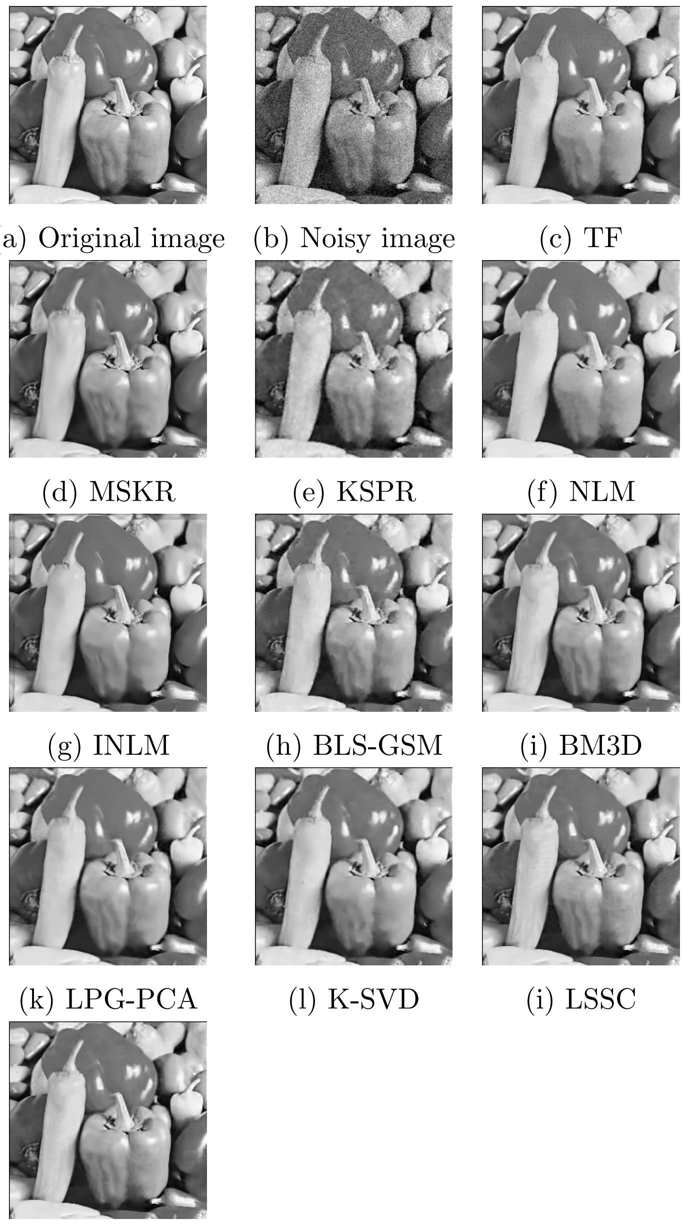

(m) CSR

Figure 2.10: Comparison of visual results on “Castle” from the Berkeley dataset with σ = 75.



(a) Original image (b) Noisy image (c) TF

(d) MSKR (e) KSPR (f) NLM

(g) INLM (h) BLS-GSM (i) BM3D

(k) LPG-PCA (l) K-SVD (i) LSSC

(m) CSR

Figure 2.11: Comparison of visual results on “Peppers” with with $\sigma = 20$.

3

Nonlocal Spatial filter for Image Denoising

3.1 Chapter Abstract

One of the most popular denoising methods based on self-similarity is called Nonlocal Means (NLM). Though it can achieve remarkable performance, this method has a few shortcomings, such as the computationally expensive calculation of the similarity measure, and the lack of sufficient candidates for some target patches. In this chapter, we propose to use clustering based on moment invariants as pre-classification, and Rotationally Invariant Block Matching (RIBM) to improve block matching. Experimental results show that the proposed technique can perform denoising better than the original NLM both in PSNR and visual results, especially when the noise level is high.

This chapter is based on the following work:

R. Yan, L. Shao, S. D. Cvetkovic and J. Klijn, "Improved Nonlocal Means Based on Pre-Classification and Invariant Block Matching", IEEE/OSA Journal of Display Technology, Vol. 8(4), pp. 212-218, April 2012.

3.2 Introduction

Image denoising is often applied in display systems to improve the image quality, because source images are usually corrupted by various additive noises. There are many denoising methods in both spatial and frequency domains. Among spatial domain methods, prevailing techniques include bilateral filter (37), trained filter (36), and Nonlocal

Means (NLM) (42) based filters, etc. State-of-the-art transform domain algorithms are GSM based method (16), and BM3D (14). As transform based methods require complex Fourier or wavelet transforms, which are usually not affordable by display devices due to hardware limitations, spatial techniques tend to be more practical.

Many natural or texture images contain repetitive patterns. One of the popular denoising methods, NLM (42), exploits this image characteristic and produces promising results both objectively and subjectively. The main idea is to replace each pixel with a weighted average of other pixels from similar neighborhoods. The main difference between NLM and previous approaches is that NLM takes advantage of the image correlation in a nonlocal manner, i.e. in the whole image or sub-image, rather than just in a small neighborhood as in local filters.

However, the original NLM algorithm is computationally intensive, especially full search within a large region is very time-consuming. Accordingly, there has been a lot of work exploring how to improve NLM. On the one hand, some of them focus on the acceleration of NLM. The most time-consuming part of NLM is the weight calculation, so a lot of methods are dominantly based on how to eliminate dissimilar patches before weighted averaging. In (43), pre-selection of contributing neighborhoods based on mean and gradient values was proposed. Similarly, local variance (44) and SVD (45) have been introduced to eliminate dissimilar pixels. In order to accelerate the weight calculation, Fast Fourier Transform has been proposed in (46), which is approximately 50 times faster than the original NLM. The method in (47) exploits the symmetry in the weight function, and computes Euclidean distance by a recursive moving average filter symmetrically, which also considerably improves the efficiency. Pang et al. (48) utilized several critical pixels in the center instead of all pixels in the neighborhood.

On the other hand, for the improvement of quantitative and qualitative results, some tuning of the smoothing parameters has been proposed in (44). In (49), a family of nonlocal image smoothing algorithms were designed which approximate the application of diffusion PDE's on a specific Euclidean space of image patches, and it can preserve the structures in the original image domain. In order to increase the number of candidates of noisy patches, the authors in (4) proposed a rotationally invariant block matching measure for nonlocal image denoising, which involves several steps such as estimating the rotation angle, rotating the block via interpolation and then applying standard block matching.

Mainly focusing on improving the denoising performance of the original NLM, we propose a pre-classification based NLM algorithm which also exploits RIBM (4) as a similarity term for block matching. To reduce the interference of the additive noise for pre-classification, Gaussian blur is first applied as a pre-filter before patch classification. Then, we use the K-means clustering on the moment invariants of image blocks as pre-classification for weighted averaging instead of a full search in the original NLM. After that, the RIBM is adopted as a rotation and mirror invariant similarity measure, which can provide more candidates for the final weighted averaging process. Symmetrical calculation of weights is also adopted before weighted averaging. The experimental results show that this method outperforms the original NLM in terms of both PSNR and visual quality.

3.2.1 Contributions

This chapter contains several contributions

- It introduces an RIBM based invariant similarity measure for increasing the number of the candidates for NLM filtering.
- It shows that our proposed method performs better compared to original NLM at high noise level.

3.3 Related Work: original nonlocal means filter and its improved algorithms

The idea of NLM is based on the fact that patches in an image always have self-similarity (37). Given a noisy image , the restored intensity of the pixel $NL(v)(i)$, is a weighted average of all intensity values within the neighborhood I. Let us denote (42):

$$NL(v)(i) = \sum_{j \in I} w(i,j)v(j), \tag{3.1}$$

where $v(j)$ is the intensity at pixel j, and $w(i,j)$ is the weight assigned to $v(j)$ for measuring the similarity between pixel i and j. The weights can be calculated by (42):

$$w(i,j) = \frac{1}{Z(i)} \exp - \frac{\|v(N_i) - v(N_j)\|_{2,a}^2}{h^2}, \tag{3.2}$$

where N_i denotes a patch of fixed size and it is centered at the pixel i. The similarity is measured as a decreasing function of the weighted Euclidean distance. $a > 0$ is the standard deviation of the Gaussian kernel, $Z(i)$ is the normalization constant with $Z(i) = \sum_j w(i, j)$, and h acts as a filtering parameter.

To find a set of reliable candidates that are similar to the current patch from a whole image, two categories of methods are applied: 1) Pre-classification; 2) Defining new similarity terms.

Pre-classification was previously used to avoid full search and to improve the efficiency of NLM. However, in most of the improved versions of NLM, it is also a contributing factor leading to better denoising results. Mahmoudi *et al.* in (43) have proposed to exploit the mean value and local average gradient vectors to exclude dissimilar patches. However, the criterion can only be applied when both gradient magnitudes of two patches are above a chosen threshold, which is easily affected by noise. Similarly, there is a pre-classification method based on patch variance (44). The drawback of the above methods is that their measures are too heuristically designed but not based on statistical properties. In (45), the SVD on the gradient field was proposed to provide statistical features (structure information) for k-means clustering. However, the gradient field is still sensitive to the change of noise level. Another shortcoming of the original NLM is that: when an image lacks repetitive patterns, many patches will not have sufficient candidates for weighted averaging. This will dramatically affect the visual results of NLM. To overcome this, Thaipanich *etal.* (45) proposed a rotated block matching scheme for obtaining more similar patches but it is only limited to several rotation angles.

Defining new similarity terms is another way to ensure reliable sets of candidates. In (4), moment invariants based block matching was proposed, which is invariant under rotation, mirroring and noise.

To achieve the goal of finding more reliable sets of candidates, our method exploits both pre-classification and defining a new similarity term. On the one hand, we want to increase the chance of finding candidates for non-repetitive patterns (45). Thus, pre-classification is used to provide candidate sets that can be from all over the image. On the other hand, because the moment invariants we use in pre-classification are rotation invariant, the neighborhoods will potentially contain rotationally unaligned candidates.

It is therefore necessary to define a new similarity term which can estimate the rotation angle during the matching process. This is where RIBM comes into play.

3.4 Improved Nonlocal Means Based on Pre-classification and Invariant Block Matching

The processing pipeline of our method is shown in Fig. 3.1. Given a noisy image, our goal is to produce a denoised image in which the noise has been removed and most of the details are retained. Similar to Eq. (3.1) and (3.2), our improved NLM can be formulated as:

$$NL(v)(i) = \sum_{j \in L} w_R(i,j)v(j), \tag{3.3}$$

$$w_R(i,j) = \frac{1}{Z_R(i)} \exp - \frac{d_R(i,j)}{h^2}, \tag{3.4}$$

where $w_R(i,j)$ depends on a new distance measure $d_R(i,j)$ and $Z_R = \sum_j w_R(i,j)$. The weighted average is performed within each cluster. L represents the number of elements within a cluster. The effect of clustering is demonstrated in Fig. 3.5. Now, the filtering problem becomes how to calculate the new weights $(w_R(i,j))$, which is explained in Sec. 3.4.2 and Sec. 3.4.3.

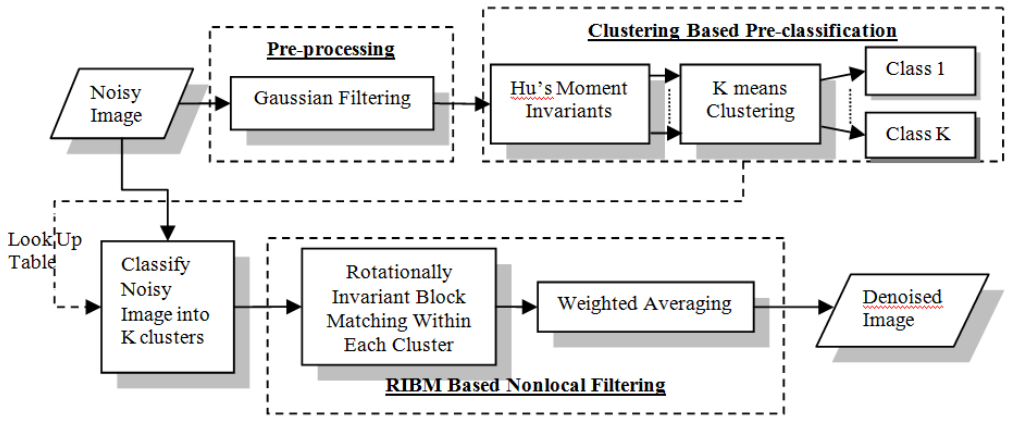

Pre-processing
Clustering Based Pre-classification
Noisy Image
Gaussian Filtering
Hu's Moment Invariants
K means Clustering
Class 1
Class K
Look Up Table
Classify Noisy Image into K clusters
Rotationally Invariant Block Matching Within Each Cluster
Weighted Averaging
Denoised Image
RIBM Based Nonlocal Filtering

Figure 3.1: Block diagram for the proposed denoising method.

The differences between our approach and NLM are:

- Gaussian blur provides the pre-processing for pre-classification. The effect is illustrated in Fig. 3.2. In the original NLM, there is no pre-processing step.
- K-means clustering on moment invariants of the blurred noisy image serves as the pre-classification for our filtering process. In the original NLM, all target patches have fixed size of candidate sets, which is either the whole image or the neighborhood centered at them.
- RIBM is calculated on the input noisy patches which have been clustered by using a Look-Up Table (LUT) from the step above. This step introduces a new similarity term for nonlocal filtering. The calculation of weights is explained in Eq. (3.11). In the original NLM, the similarity term just relates to the Euclidean distance as shown in Eq. (3.1).

3.4.1 Pre-processing

Classification is usually based on the structural information in an image. However, pixel based pre-classification in a noisy image will not be very accurate (46). Therefore, we consider to use a pre-filter to smooth the image with the purpose of avoiding coding the wrong pattern, especially for high noise levels.

Here we adopt the Gaussian filter (33) as a pre-filter for our subsequent classification due to its efficiency. The result of the Gaussian filtering is for guiding the classification, but the input for the actual denoising is still the original noisy image. Define a $(2m+1) \times (2m+1)$ mask, the center of it is (0, 0) and other x, y range from (-m, -m) to (m, m). The element in this mask is defined as:

$$G_\sigma(x,y) = exp(-\frac{x^2+y^2}{2\sigma^2}), \tag{3.5}$$

where $x, y = -m, ..., 0, ..., m$ and σ is the standard deviation of the Gaussian distribution. Normalization is necessary if we need to maintain the brightness level of the image:

$$S_\sigma = \sum_{x=-m}^{m} \sum_{y=-m}^{m} G_\sigma(x,y), \tag{3.6}$$

$$G_{k\sigma}(x,y) = G_\sigma(x,y)/S_\sigma. \tag{3.7}$$

The result of Gaussian blur for the whole image is given by:

$$Gb = G_{k\sigma} * v, \tag{3.8}$$

where v is the intensity of the input noisy image and $*$ denotes the convolution operation. In our implementation, a large σ is not necessary, because most details of the input noisy image should be retained and Gaussian blur with a large σ might introduce artifacts. After the Gaussian filter is applied on the input noisy image, the blurred image serves as the input of classification. This is exemplified by the images in Fig. 3.2.

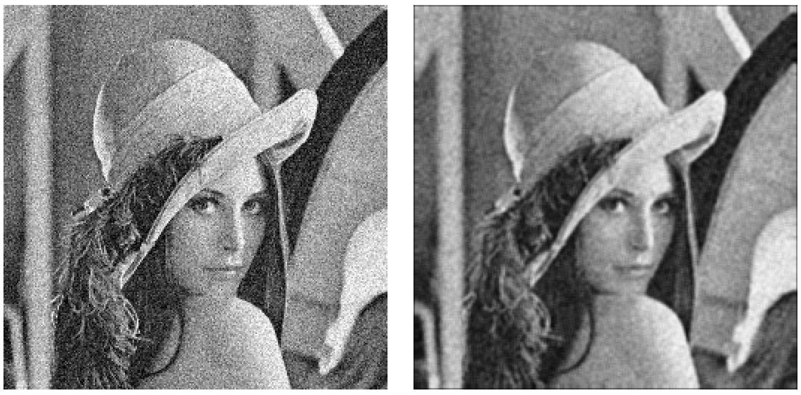


Figure 3.2: Illustration of Gaussian blur. The sigma in Gaussian filter is 10, and the m in Eq. (3.6) is 4.

3.4.2 Clustering Based Pre-classification

Moment invariants have been applied in many fields, such as face recognition, character recognition and motion estimation. They have been proved to be a robust image descriptor, which is invariant under translation, changes in scale, and also rotation. In (96), higher order moment invariants were proved to be more vulnerable in the case of additive white noise. Therefore, in our algorithm, Hu's moment invariants (96) are applied, which has the highest order of 2, as feature descriptor (1×7 vector) for K-means clustering. Given an $N \times N$ image and an $n \times n$ patch which is centered at location i ($i = 1, 2, ..., N \times N$), the moment invariants of this patch can be represented by a 1×7 vector. Then, for the whole image, we have $N \times N$ such vectors which serve as the input vectors of the K-means clustering. The process can be conducted as follows:

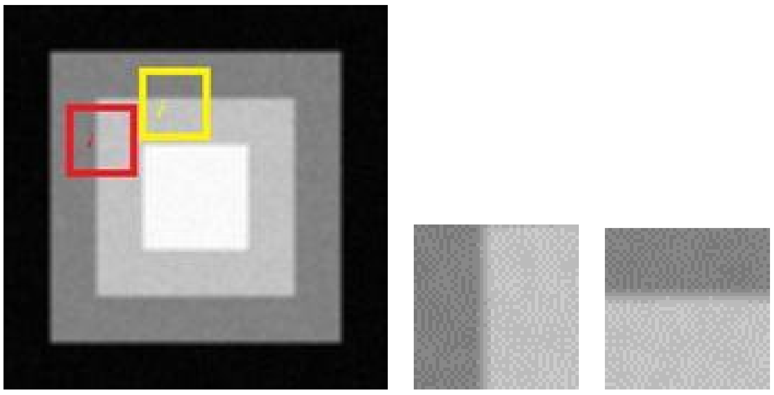
$$\arg\min_{c} \sum_{k=1}^{K} \sum_{H(Gb(i)) \in Hm_k i=1,2,...,N\times N} |H(Gb(i)) - \mu_k|^2, \tag{3.9}$$

where $Gb(i)$ represents a Gaussian blurred patch centered at i. $H(\cdot)$ outputs the moment invariants of an input patch. μ_k is the mean vector for the kth cluster Hm_k. Afterwards, we can obtain K clusters Hm_1, Hm_2, Hm_k, ..., Hm_K. Each cluster, Hm_k is composed of L (different in every cluster) vectors Hm_{kl} $(k = 1, ..., K, l = 1, 2, ..., L)$. Here we use the most common algorithm which applies an iterative refinement technique (97). In this stage, K-means clustering provides the preselected candidates for the following weighted averaging. The classification information (the coordinates of the patch center) is stored in the LUT. Weighted averaging is performed within each cluster during the following process.

3.4.3 RIBM Based Nonlocal Filtering

Similarities that are calculated based on raw intensity values are sensitive to noise, scale difference, rotation and mirroring. NLM involves pixels that have possible similar neighborhoods in the averaging process, which is accomplished by global or semi-global search. However, due to the variety of image contents, many patches could probably lack reliable candidates within the image or the neighborhood, which leads to degraded denoising quality. In denoised image of NLM algorithm, noisy pixels were used as the denoised pixels when the patches centered at those points do not have sufficient candidates. This causes a lot of artifacts, especially when the noise level is high.

Sven *et al.* (4) used moment invariants for improving block matching. It only improves the matching performance of NLM within a spatial neighborhood, which is centered at the target patch. In order to define a candidate set that contains different patches from all over the image, we first perform clustering based on moment invariants, which is applied to the Gaussian blurred image. The patches within each cluster could have a similar structure but in different orientations. As shown in Fig. 3.1, the input noisy image is classified into K clusters by using a LUT from clustering based Pre-classification. Within each cluster, the moment invariants of all the patches are similar but their orientations of the patches might be different. In order to obtain better weight calculation during the nonlocal filtering process, angle estimation based on moment invariants is carried out for block matching.



(a) Square image (256 × 256) (b) Patch N_i (c) Patch N_j

Figure 3.3: Illustration of RIBM from noisy image corrupted by white Gaussian noise with standard deviation 10 and zero mean and then blurred by Gaussian filter with $\sigma = 5$. If it is in NLM, these two patches N_i and N_j will have small weight for each other due to their Euclidean distance is high. If we rotate patch N_i by 90 degrees counterclockwise, their Euclidean distance will be much smaller. The difficult part is in noisy patches we cannot easily estimate rotation angle. RIBM (4) solved this problem by using moment invariants.

The problem lies in how to estimate the rotation angle between two corresponding patches N_i and N_j. The general idea of RIBM is as follows (4): 1) If patch N_j is a mirrored version of N_i, N_j needs to be mirrored to get $N_j^{'}$. Otherwise, $N_j^{'}$ is the same as N_j (Eq.(3.10)); 2) The rotation angle between patch N_i and $N_j^{'}$ (Eq.(3.10)) is calculated; 3) For each pixel in N_i, the corresponding pixel in $N_j^{'}$ should be found after using the estimated rotation angle (Eq.(3.10)); 4) The final distance measure is the sum of the intensity differences between pixels in N_i and corresponding pixels in $N_j^{'}$ (Eq.(3.11)).

The implementation of RIBM weight calculation is explained as follows. More details can be found in (4):

First, the definition of block centroid is given as follows: a) Given that $N_j^{'}$ is a noisy and rotated patch of the patch N_i. To define the centroid, a coordinate system originated at the block center is used for marking the position of pixels in a path (4).

$$c_i = \begin{pmatrix} \frac{\int_i x_b \cdot v(x_b,y_b) dx_b dy_b}{\int_i v(x_b,y_b) dx_b dy_b} \\ \frac{\int_i y_b \cdot v(x_b,y_b) dx_b dy_b}{\int_i v(x_b,y_b) dx_b dy_b} \end{pmatrix},$$

where $v(x_b, y_b)$ refers to the intensity value within the patch N_i. Let $\vec{c_i}$ denote the normalized vector starting from the centroid of N_i (The definition of this coordinates can be found in (4)).

b) In the calculation of the rotation angle, $\vec{c_i}$ can be represented by vector $u = (u_1, u_2)^T$. Let $m_{i,j}(u)$ be a function that can be defined as the following equation (4):

$$m_{i,j}(u) = \{ \begin{array}{cc} (-u_1, u_2)^T & \phi_7(i) \cdot \phi_7(j) < 0 \\ (u_1, u_2)^T & else \end{array} .$$

In our implementation, the seventh moment of Hu ϕ_7 has been applied to compute $m_{i,j}(u)$, because the sign of ϕ_7 remains comparatively stable under noise but will change under mirroring.

Second, how to estimate the rotation angle?

To align the orientations of candidates for target patch N_i, we need to estimate the rotation angle between N_i and every candidate $N_j \in (N_1, N_2, N_{i-1}, ..., N_{i+1}, N_L)$ which is within the same cluster. In this chapter, the pixels of a block are defined as vectors from the block's center, so all the vectors should be rotated by the same angle. Based on this, the rotation angle of two blocks can be estimated by the rotation angle of two blocks' centroids $\vec{c_i}$ and $\vec{c_j}$ (4). Therefore, the rotation matrix between two blocks can be expressed as (96):

$$R_u = \begin{pmatrix} u_1 & -u_2 \\ u_2, & u_1 \end{pmatrix}$$

Thereafter, the corresponding point of q_i (any point in patch N_i) is q_j (rotated corresponding point in another patch N_j) (4):

$$q_j = m_{i,j}(R_{i,j} \cdot q_i). \tag{3.10}$$

The m function here is for compensating the mirroring in Eq. (3.10). Finally, the similarity term which replaces $\|v(N_i) - v(N_j)\|_{2,a}^2$ in the original NLM weights

(Eq.(Eq:weights)) can be calculated by (4):

$$d_R(i,j) = \sum_{q_i \in i} (v_i(q_i) - In(v_j, q_j))^2, \tag{3.11}$$

where In denotes the bilinear interpolation. For each point q_i in patch N_i, after rotation and interpolation, we can get its corresponding point q_j in patch N_j. However, in Eq. (3.2), for each point q_i in patch N_i, its corresponding point in another patch N_j has the same 'within-patch' coordinates: $q_i = q_j$. Our weight function is then defined by Eq. (3.4).

3.5 Experimental Validation

In our experiments, the image data set is defined as: "Man.png", "JFKgray.bmp", "coinlaundry.bmp", "Barbara.TIF", "house.png", "peppers256.png", "lena.tif", "ZeldaG.tif", "Cameraman.png". For performance evaluation, we compare our proposed method with the original NLM and a recent related method (4) based on this dataset. The evaluation metrics we adopt in our experiments are Peak signal-to-noise ratio (PSNR) and Structural SIMilarity (SSIM).

PSNR is employed to provide quantitative evaluations of the denoising results. PSNR is defined as:

$$PSNR = 10 \log_{10}(\frac{L^2}{MSE}), \tag{3.12}$$

where L_D is the dynamic range of the image and MSE is the mean squared error between the original and the reconstructed images.

SSIM (93) is a metric which is more consistent with human subjective perception. SSIM can be calculated as follows (93):

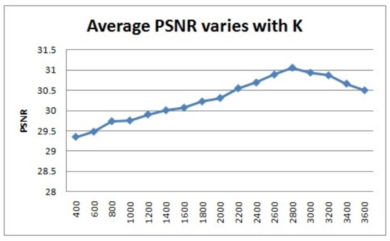
$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{3.13}$$

$C_1 = (K_1L)^2, C_2 = (K_2L)^2$ are constants to avoid instability when $\mu_x^2 + \mu_y^2$ or $\sigma_x^2 + \sigma_y^2$ are too close to zero [20]. L is with the dynamic range of the intensity values, and $K_1 \ll 1$, $K_2 \ll 1$ are small constants. μ_x and μ_y are the mean intensities. σ_x and σ_x are standard deviations (the square root of variance).

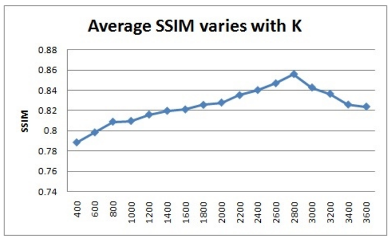
3.5.1 The parameters of clustering

We implemented our clustering method based on moment invariants. For standard K-means clustering, there are several parameters which need to be decided. The type of distance we use, the number of clusters we assign, and the length of vectors we use in our NLM based framework.

Here we exploit the Euclidean distance for measuring the distance between two feature vectors as (45) did. According to (98), we choose the patch size to be 9 × 9. To test how the performance of the method varies with different values of K, we vary K in the range of 400 and 3600. The average PSNR and SSIM scores of our dataset (standard deviation of Gaussian noise is 20) using different numbers of clusters are shown in Fig. 3.4.



(a) Average PSNR



(b) Average SSIM

Figure 3.4: PSNR and SSIM vary with K

The changing trends of PSNR and SSIM are roughly the same: when K becomes larger, there are more clusters representing different types of details. However, if the K goes too high, some clusters will not have enough candidates. As a result, the PSNR and SSIM go down after the climax. Therefore, if complexity is not a concern, we can choose the optimal value of K depending on the size of the input noisy image. For our testing set, all the images are 256 × 256, so we choose $K = 1800$ (when $K = 2800$ it takes more than twice of the time as $K = 1800$ takes.) to guarantee enough candidates for each patch according to the variation of visual results when we change K. The visualization of the K-means clustering on Lena is shown in Fig. 3.5.

As is shown in the visualization, our clustering method can detect edges and details according to different structures and light variation. Compared to the clustering in

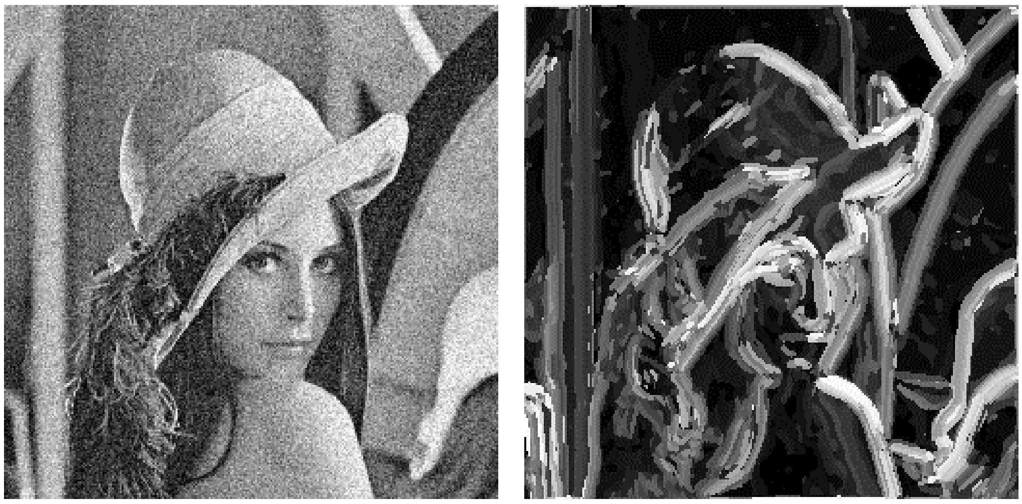


Figure 3.5: The clustering results. Different classes have been represented in different grayscales. The grayscale level [0,255] has been divided by K, and each class k is represented by $k \times (255/K)$. In this figure, the K is 256.

(45), our method distinguishes the grayscale variation better, which results in more adaptive classifications.

3.5.2 Denoising quality and comparison

In this experiment, those images are corrupted by additive Gaussian noise with standard deviation $\sigma = [15, 20, 25, 30, 35, 50, 100]$. The smoothing parameters h in Eq. (3.4) are the same as h in (42) for fair comparison. In all our experiments, h have been fixed to 12σ. The σ in Eq. (3.5) has been fixed to $0.5 \times \sigma$ (half of the standard deviation of input noisy image). The radius of the patch size in Eq. (3.5) is 4 ($m = 4$). The PSNR and SSIM comparisons are listed in Table 3.1 and 3.2. From the quantitative results we can see that, when the standard deviation of Gaussian noise is high, the original NLM suffers, because intensity difference is the only similarity term, which will easily result in difficulties of finding similar patches. The proposed method significantly outperforms both the original NLM and the method in (4), especially for highly noisy images.

The difference in visual quality between the three methods can be inspected in the examples shown in Fig. 3.6, Fig. 3.7 and Fig. 3.8. We observe that the proposed method can not only preserve better details but also remove severe noise. The method in (4) employs RIBM but it is applied to neighborhoods, which may cause lack of proper candidates when the variation of the textures is strong. So certain regions still remain noisy. Our algorithm overcomes this by obtaining sufficient reliable candidates from K-means clustering. Fig. 3.9 shows visual results on a typical image when the standard deviation of Gaussian noise is 50. We can see that the original NLM is almost ineffective. When the noise level is high, the intensity based matching between patches is vulnerable to noise. Our scheme has adopted Gaussian blur as pre-processing and moment invariants are robust in noise inference as well. The method in (4) has reconstructed rough structure but still failed to retain details. Our algorithms preserves the main structures much better compared to other approaches (the original NLM and (4)). It demonstrates that using clustering before weighted averaging can ensure most patches to get reliable candidates.

3.6 Conclusion

In this chapter, we proposed an improved NLM method. It applies moment invariants based K-means clustering on the Gaussian blurred image, which provides better classification before weighted averaging. In addition, RIBM adds more 'similar patches' which have been rotated by certain angles to make them more correlated to the reference patch. Experimental results show that clustering on moment invariants is very effective for pre-classification. The proposed algorithm can effectively reconstruct finer details and at the same time introduce fewer artifacts than the other methods. The K-means clustering used in our proposed method is a time-consuming part. In future work, we will investigate more efficient clustering methods to speed up the pre-classification step.

Table 3.1: Average PSNR and SSIM.

Images	Algorithms	15	20	25	30	35	50
Man	NLM	30.6096	28.9415	27.2551	25.6032	24.0274	19.8203
	Ref(4)	31.9042	28.9987	27.1129	25.3949	24.1578	20.6564
	Proposed	31.8033	30.1200	28.1805	26.6425	25.7073	23.1468
JFK	NLM	31.7868	30.1013	28.2655	26.3589	24.5918	20.1412
	Ref(4)	31.8556	30.1111	28.2138	26.0098	25.0907	22.3415
	Proposed	32.8840	30.9815	29.0523	27.2328	26.3026	24.9040
Coinloaund	NLM	30.1205	28.7988	27.2832	25.6843	24.1319	19.8145
	Ref(4)	30.7659	28.8802	27.3558	26.0087	24.6223	21.0876
	Proposed	31.3885	30.0122	28.4804	26.9449	26.1124	23.7858
Barbara	NLM	32.1434	30.0025	27.9899	26.0446	24.3105	19.8429
	Ref(4)	32.3352	29.8816	28.0953	26.5455	25.9618	22.3722
	Proposed	33.4783	31.1976	29.0268	28.0454	26.9828	25.8292
House	NLM	33.5255	31.3356	29.0487	26.8699	24.9479	20.1644
	Ref(4)	33.6413	32.0187	29.1559	27.0901	25.9089	22.7390
	Proposed	33.8645	31.8396	29.8417	28.6245	27.6680	25.3085
Peppers	NLM	31.4137	29.5473	27.6721	25.8659	24.1645	19.7590
	Ref(4)	32.0053	30.5619	27.8097	26.6315	24.4551	21.0076
	Proposed	33.6801	31.1678	29.0657	27.4582	26.2204	23.6793
Lena	NLM	30.8749	29.1526	27.3969	25.7290	24.0721	19.8365
	Ref(4)	31.1523	30.0052	27.4430	25.8831	24.4548	21.1243
	Proposed	33.2810	30.7653	28.5930	27.0240	26.0495	24.2728
Zelda	NLM	32.4352	30.5207	28.4973	26.4933	24.7115	20.1715
	Ref(4)	32.5744	30.6654	28.9895	26.0971	25.0092	22.5567
	Proposed	33.0517	31.2918	30.0415	28.9240	27.6842	26.6160
Cameraman	NLM	30.3578	28.8968	27.1939	25.7950	24.1379	19.9039
	Ref(4)	31.1236	29.6548	28.2336	26.6559	25.8819	22.0012
	Proposed	31.9462	31.0414	30.3732	27.6452	27.1293	23.5498

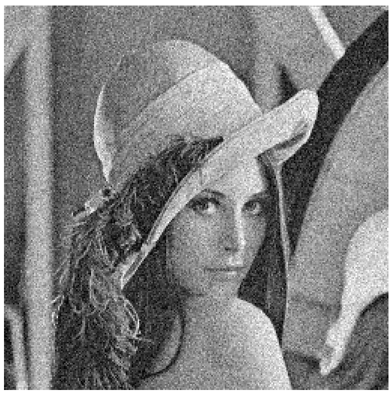
Table 3.2: Average PSNR and SSIM of several algorithms.

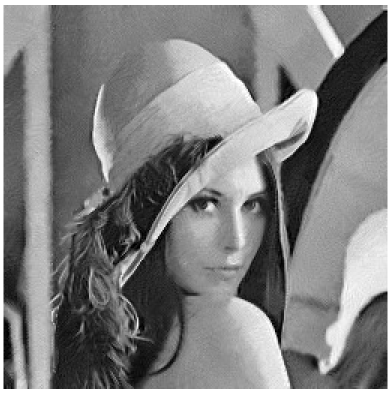
σ	NLM		Ref(4)		Proposed	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
15	31.47	0.8509	31.93	0.8575	32.82	0.8938
20	29.70	0.7811	30.09	0.7948	30.94	0.8424
25	27.84	0.6983	28.05	0.7243	29.18	0.7694
30	26.05	0.6144	26.26	0.6733	27.62	0.7035
35	24.34	0.5360	25.06	0.5963	26.65	0.6630
50	19.94	0.3947	21.77	0.4731	24.57	0.5741
100	11.52	0.2535	13.51	0.2685	16.56	0.4427

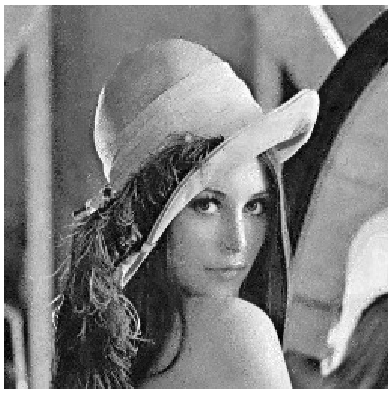
Table 3.3: Average quantitative denoising results on 9 images.

Images	Metrics	20	30	50
Man	PSNR	30.12	26.64	23.15
	SSIM	0.8512	0.8106	0.6747
JFKgray	PSNR	30.98	27.23	24.90
	SSIM	0.7957	0.6232	0.4394
Coinlaund	PSNR	30.01	26.94	23.79
	SSIM	0.8187	0.7483	6013
Barbara	PSNR	31.20	28.05	25.83
	SSIM	0.8944	0.8207	0.873
House	PSNR	31.84	28.62	25.31
	SSIM	0.8261	0.6273	0.4662
Peppers	PSNR	31.17	27.46	23.68
	SSIM	0.8686	0.6939	0.4881
Lena	PSNR	30.77	27.02	24.27
	SSIM	0.849	0.675	0.5615
ZeldaG	PSNR	31.29	28.92	26.62
	SSIM	0.8251	0.6519	0.5271
Cameraman	PSNR	31.04	27.65	23.55
	SSIM	0.8528	0.6804	0.5356
Average	PSNR	30.94	27.62	24.57
	SSIM	0.8424	0.7035	0.5741

(a) Original image


(b) Noisy image


(c) NLM


(d) Ref.(4)

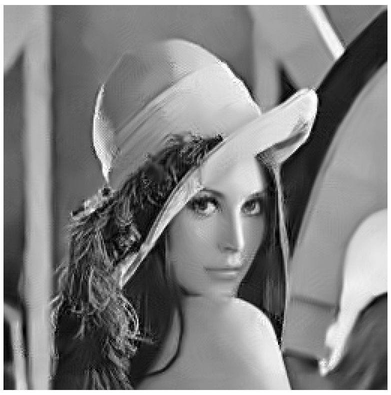
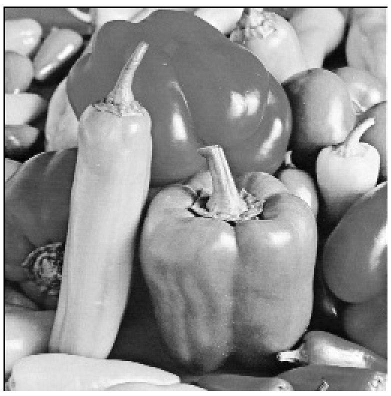
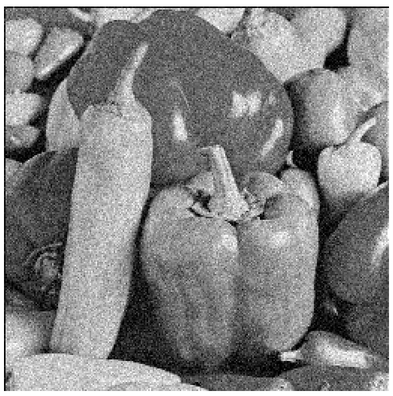
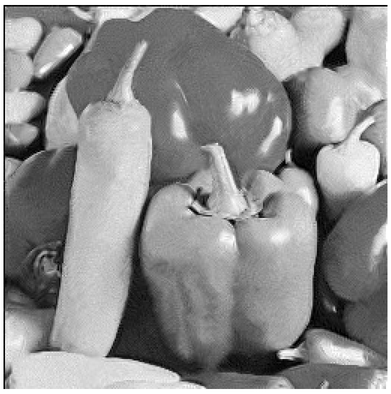
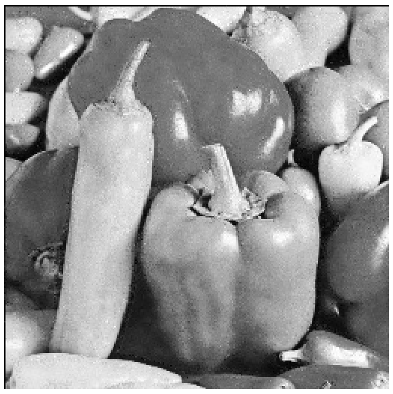

(e) Proposed method

Figure 3.6: Comparison of visual results when sigma equals to 20.


(f) Original image


(g) Noisy image


(h) NLM


(i) Ref. (4)

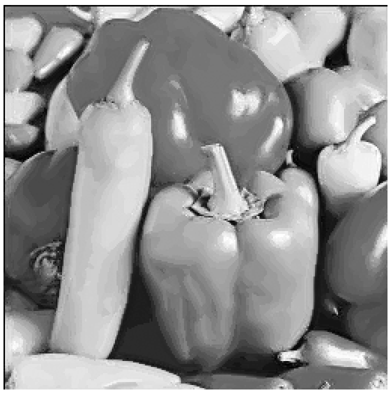
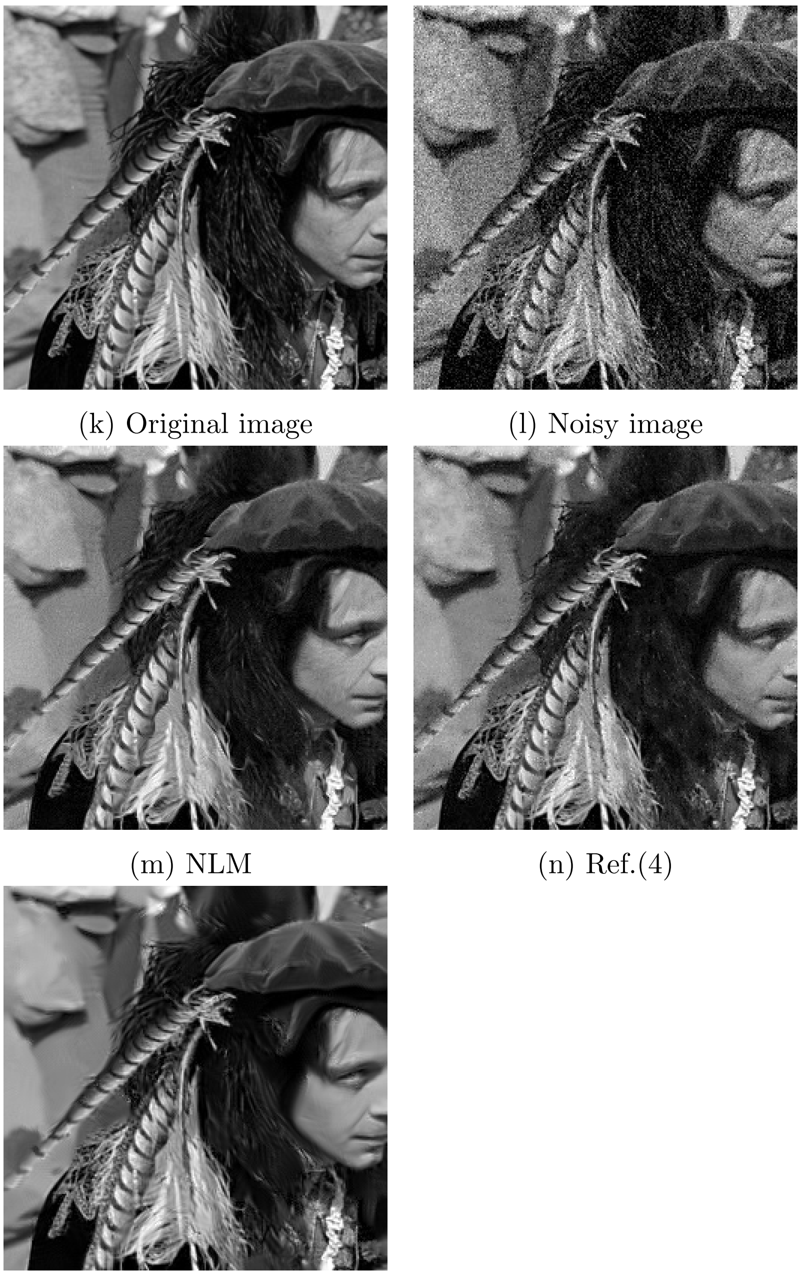

(j) Proposed method

Figure 3.7: Comparison of visual results when sigma equals to 20.



(k) Original image

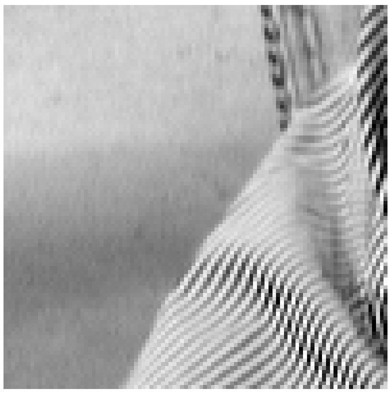
(l) Noisy image

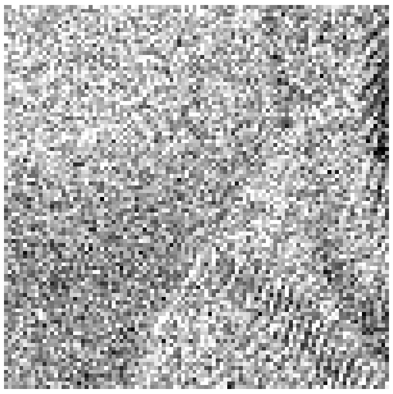
(m) NLM

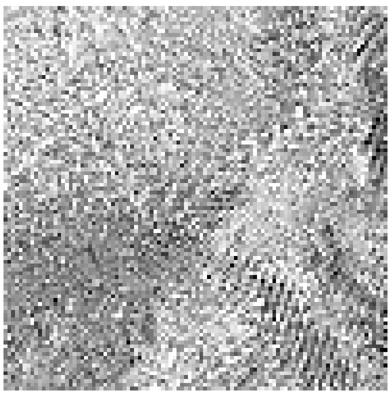
(n) Ref.(4)

(o) Proposed method

Figure 3.8: Comparison of visual results when sigma equals to 20.


(a) Original image


(b) Noisy image


(c) NLM

(d) Ref.(4)

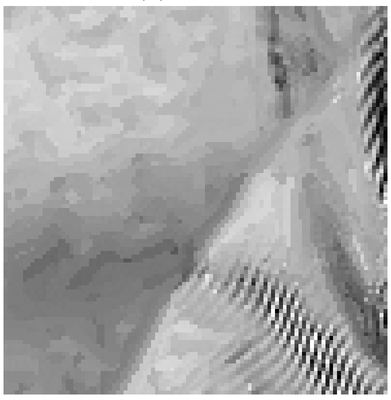

(e) Proposed method

Figure 3.9: Comparison of visual results when sigma equals to 50.

4

Image Denoising in Wavelet Domain using Hierarchical Dictionary Learning

4.1 Chapter Abstract

Exploiting the sparsity within representation models for images is critical for image denoising. The best currently available denoising methods take advantage of the sparsity from image self-similarity, pre-learned and fixed representations. However, most of these methods still have difficulties in tackling high noise levels or noise models other than Gaussian. In this chapter, the multi-resolution structure and sparsity of wavelets are employed by nonlocal dictionary learning in each decomposition level of the wavelets. Experimental results show that our proposed method outperforms two state-of-the-art image denoising algorithms on higher noise levels. Furthermore, our approach is more adaptive to the less extensively studied uniform noise.

This chapter is based on the following work:

R. Yan, L. Shao and Y. Liu, "Nonlocal Hierarchical Dictionary Learning Using Wavelets for Image Denoising", IEEE Transactions on Image Processing, vol. 22, no. 12, pp. 4689-4698, Dec. 2013.

4.2 Introduction

Image denoising as a low-level image processing operator is an important front-end procedure for high-level visual tasks such as object recognition, digital entertainment, and remote sensing imaging. In real camera systems, the noise has various sources, such as fixed pattern noise, thermal noise, and quantization noise. This chapter focuses on denoising, defined as the recovery of an underlying image from an observation that has been corrupted by zero mean additive noise, which can be formulated as:

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \boldsymbol{\eta}_i, \tag{4.1}$$

where $\boldsymbol{x}_i$ is the vectorized original image patch with the ith pixel at its center, $\boldsymbol{y}_i$ is the i-th patch of the observed image $\boldsymbol{y}$, and $\boldsymbol{\eta}_i$ is the independent additive noise. The concrete noise models we consider are Gaussian noise and uniform noise. The Gaussian distribution can be used to approximate the noise generated by the intrinsic thermal or electronic fluctuations of the acquisition devices (99)(100). The quantization noise, which can be approximated as the uniform distribution, is caused by image quantization.

Sparse coding, as a popular topic for many signal processing tasks, is widely used in solving the image denoising problem (1, 16, 17, 91). The generative model for sparse representation of the denoising problem is:

$$\boldsymbol{y}_i = \boldsymbol{D}\boldsymbol{\alpha}_i + \boldsymbol{\eta}_i, \tag{4.2}$$

where $\mathbf{D}$ is the dictionary [1], and $\boldsymbol{\alpha}_i$ is the representation matrix for patch $\boldsymbol{y}_i$. Recent methods based on sparse representations can be categorized into two groups according to the dictionary basis: fixed basis or learned basis.

Though the fixed basis dictionaries have a large number of variations (Discrete Cosine Transform (DCT) (12), wedgelets (72), curvelets (73, 74), bandlets (59, 75), contourlets (76), and steerable wavelets (15, 77)), wavelet (13) based dictionaries have a clear advantage for image denoising because of the merits of the wavelet transform:

[1]The dictionary contains a series of vectors trained from images for linearly representing images/patches. Elements in the dictionary are usually unit norm functions called atoms (101).

sparsity, multi-resolution and similarity with the human visual system (102). For instance, over-complete wavelets (103) are exploited in BLS-GSM (16) to remove the artifacts of critically sampled wavelet coefficients. BM3D (14) generates remarkable results by applying a sparse wavelet transform on a group of similar image patches. Though wavelet-based methods have achieved state-of-the-art PSNR performances, the shrinkage or modification of wavelet coefficients sometimes produce low-frequency noise and edge ringing. On a different track, the multi-resolution property of wavelets enables the idea of learning multi-scale dictionaries (104) (105), which can produce sparser representations compared to single-scale wavelet dictionaries. However, sampling techniques involved in (105) increase the computation complexity. A prior distribution, over the basis function coefficients, peaked at zero and tapering away smoothly was used in (104). The smooth prior causes decreased coding efficiency by not forcing inactive coefficients to have values exactly equal to zero.

Different from the fixed basis dictionaries, which are usually restricted to images of a certain type, the atoms of the learned basis dictionaries can also be empirically learned from image examples, which will apply to any family of images (106). Representative learned dictionaries include adaptive learned dictionary (K-SVD) (17), locally learned dictionary (KLLD) (91), and learned simultaneous sparse coding (LSSC) (1). These sparse coding methods can work effectively on denoising because their learned dictionaries give more adaptive image priors for Bayesian estimation than the one based on fixed basis dictionaries. However, the way they identify low frequency and high frequency image information is by the magnitudes of the sparse coefficients, which are obtained from the single-scale dictionary. Consequently, for the high frequency information of an image, it is difficult to distinguish noise from image details using sparse coefficients. Therefore, in some of the previous sparse coding methods, even though the quantitative results are promising, the artifacts in the denoised images are quite noticeable. For instance, in KLLD, a clustering on the weights of the steering kernel regression is performed before the local dictionary learning. Within each cluster, Principal Component Analysis (PCA) is applied to model the cluster according to the geometric information. This scheme works well for medium or low noise levels. However, the clustering is not very reliable at high noise levels due to the fact that weights of steering kernel regression are vulnerable to severe noise. In this chapter, we address this issue by applying the sparse coding framework to wavelet coefficients due to their multi-resolution properties. In the wavelet domain, even when the noise level is high, the noise and image details still mostly exist in the higher subbands, which makes the denoising in lower subbands easy to succeed. This is much more invariant to the noise compared to steering weights used in KLLD. In this way, the over-complete dictionary we build is more invariant to noise compared to KLLD.

The other way to consider denoising methods is from the point of view of their "locality" - either local, or nonlocal (67). A filter is local if the candidate selection process (finding similar patches or pixels) used for filtering is restricted by the spatial distance. A filter is nonlocal if the candidate selection depends only on the photometric similarity and is not restricted by the spatial distance. Under such definitions, the above methods are all local methods except for BM3D and LSSC. As shown in (67) and (1), BM3D and LSSC perform the best among the state-of-the-art algorithms, because they employ the image self-similarity. Apart from the better pre-filtering, LSSC slightly outperforms BM3D in certain noise levels because of its adaptive learned basis functions. This motivates us to adopt the nonlocal idea in our proposed scheme. In the wavelet domain, the coefficients have correlations within each decomposition level between subbands. In this chapter, a nonlocal clustering is proposed in the wavelet domain before the dictionary training to guarantee that in each decomposition level the scale dictionary is made of a series of sub-dictionaries corresponding to different clusters. In this case, the combined scale dictionary is redundant and structured for the wavelet coefficient reconstruction step.

Previous work by Ophir *et al.* (107) aims at combining the merits of multi-scale representations and learned dictionaries. In their approach, the sub-dictionary is suggested to be trained in several different ways. First, it can be trained using patches within each subband. Second, patches coming from all subbands at the same direction can be used to train the sub-dictionary. Finally, the sub-dictionary can also have multi-band atoms, which come from the three subbands of the same decomposition level. Compared to their method, in our algorithm, the dictionaries are "nonlocal for each decomposition level" because we consider the similar patches from all three subbands in the same scale rather than just patches in the same subband or multi-atoms from the same relative positions of all subbands. In this way, the extra sparsity between subbands can be exploited. In our proposed scheme, we intend to merge the advantages of the nonlocal dictionary and the hierarchical representation of wavelets.

To summarize, in this chapter, we propose an iterative nonlocal multi-scale dictionary learning scheme in the wavelet domain. Extensive experimental results show that our proposed method can achieve competitive performance compared to the state-of-the-art. The contributions of this chapter are twofold:

- Though training different dictionaries in each cluster has been employed in KLLD, we apply clustering to the wavelet coefficients in the same decomposition level. To the best of our knowledge, this has never been done in any previous work. In this way, our dictionary learning exploits sparsity from both the inner and inter subbands in the same scale. At the same time, our dictionary learning is hierarchical due to the multi-resolution characteristics of wavelets.
- The reweighted l_1 norm is used for better dictionary learning in each cluster, and l_0 norm is employed for the purpose of obtaining sparse codes for better reconstruction of the whole decomposition level.

4.3 Related Work

4.3.1 The Learned Simultaneous Sparse Coding

As an adaptive learning model, sparse coding has provided promising results compared to previous local filters (102). While the local filters can exploit the image self-similarity within neighborhoods, learned dictionaries by sparse coding usually utilize the sparsity from external training data. Assuming we need to denoise a patch $\mathbf{y}_i$ in $\mathbb{R}^m$ with a dictionary $\mathbf{D}$ in $\mathbb{R}^{m\times k}$, we can address the problem by solving the following optimization function:

$$\min_{\boldsymbol{\alpha}_i\in\mathbb{R}^k} \|\boldsymbol{\alpha}_i\|_p \quad s.t. \|\boldsymbol{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 \le \varepsilon, \tag{4.3}$$

where $\|\boldsymbol{\alpha}_i\|_p$ is the sparse-inducing regularization term, and $\mathbf{D}\boldsymbol{\alpha}_i$ is an estimate of the latent patch.

However, only exploiting the sparsity within a neighborhood for the current patch is not optimal for the denoising task (1). Mairal *et al.* (1) proposed learned simultaneous sparse coding to extend the local sparse coding model to nonlocal by grouping similar patches and forcing them to use the same dictionary atoms with different sparse codes. Concretely, T_i can represent the set of similar patches for patch $\boldsymbol{y}_i$:

$$T_i \triangleq \{j = 1, ..., n \quad s.t. \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \leq \xi\} \tag{4.4}$$

where ξ is the threshold, n is the size of the image. In the Simultaneous Sparse Coding model, the approximation of the clean patch for the current noisy patch $\boldsymbol{y}_i$ can be solved by a grouped-sparsity regularizer (1):

$$\begin{aligned} &\min_{(\mathbf{A}_i)_{i=1}^n, \mathbf{D} \in \mathcal{C}} \sum_{i=1}^{n} \frac{\|\mathbf{A}_i\|_{p,q}}{|T_i|^p} \\ &s.t. \forall i \sum_{j \in T_i} \|\boldsymbol{y}_j - \mathbf{D}\boldsymbol{\alpha}_{ij}\|_2^2 \leq \varepsilon_i, \end{aligned} \tag{4.5}$$

where $\mathbf{A}_i = [\boldsymbol{\alpha}_{ij}]_{j \in T_i} \in \mathbb{R}^{k \times |T_i|}$, $\mathcal{C}$ is the set of matrices in $\mathbb{R}^{m \times k}$ with unit l_2 norm columns, and ε_i is chosen according to the size of T_i: $\varepsilon_i = \sigma^2 F^{-1}_{m|T_i|}(\tau)$. F_m^{-1} is the inverse of F_m which is the cumulative distribution function of the χ_m^2 distribution. In LSSC, it is suggested that when $\tau = 0.9$, ε has acceptable values.

As claimed in (1), this method can exploit most sparsity within a matched group. For instance, if each patch $\mathbf{y}_i$ in a group takes p_i sparse codes to estimate the clean patch, it takes $|T_i| \times p_i$ sparse codes to estimate the clean group in the method of (4.3). Nevertheless, in the LSSC framework, it only takes $|T_i| \times p$ ($p \leq p_i$) sparse codes to represent the group.

Though LSSC achieves one of the best denoising results, it still has some drawbacks: 1) the regularizer is a standard way to solve data fitting problem but not specialized in denoising. All the thresholds ε_i are preset in most optimization processes empirically (1). The image contents are not involved in the above regularizer. Intuitively, for different patterns, the regularizer should have adaptive thresholds. Even though K-means clustering in this scheme has already considered the structured information, it is still a rough approximation which can result in artifacts. 2) The dictionary size in this scheme is empirically decided. Therefore, for different input natural images, the pre-decided size of a dictionary atom might not fit the new details. For instance, if the size of the dictionary atom is larger than the image details, then the denoised image would have oversmoothing artifacts.

4.3.2 Block Matching 3D (BM3D)

As a milestone in the research of image denoising, BM3D achieves remarkable results because it fully exploits the sparsity within a single image. The pre-processing step is similar to LSSC in that it groups similar patches within an image into 3D patch cubes. Then the wavelet transform is applied on those cubes. In the wavelet domain, the wavelet coefficients are filtered by thresholding and Wiener filtering in two successive steps.

This method works well on images with abundant repetitive patterns. However, for images with unique patches (which have few similar patches in the image), BM3D produces sub-optimal results. Sparse coding methods can deal with this issue by incorporating a prior on images via an initial dictionary trained on high quality datasets. This exploits the sparsity in a trained dictionary.

4.4 Proposed Formulation

Suppose the input noisy image $\boldsymbol{y}$ is from a clean image $\boldsymbol{x}$ contaminated by additive noise with zero-mean. First, we transform $\boldsymbol{y}$ into the wavelet domain which contains several decomposition levels. Within each level, the wavelet coefficients are divided into overlapping patches of a fixed size and each patch is modeled as a vector variable. Then, we apply k-means clustering to the vectors. Afterwards, in each cluster, a sub-dictionary is trained through a reweighted l_1 norm regularization process. For each decomposition level, all the trained sub-dictionaries need to be combined as the over-complete dictionary for this scale. The l_0 norm regularization process is then performed on the trained dictionaries and the wavelet coefficients are then denoised by the sparse coding process. Finally, the wavelets will be transformed back to the spatial domain, which results in the estimated denoised image $\hat{\boldsymbol{x}}$. In the following sections, we describe each of the steps in detail.

4.4.1 Nonlocal Hierarchical Dictionary Learning in Wavelet Domain

As suggested in Ophir *et al.* 's work (107), the wavelet based dictionary learning can further exploit the sparsity between the wavelet coefficients:

$$\arg\min_{\mathbf{D}^w,\mathbf{S}^w} \|\mathbf{WY}-\mathbf{D}^w\mathbf{S}^w\|_2^2 \quad s.t.\|s_i^w\|_0 < T, \forall i, \tag{4.6}$$

where $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2 ... \boldsymbol{y}_N]$ is the set of training samples, $\mathbf{W}$ is the wavelet analysis operator, $\mathbf{D}^w$ denotes the learned dictionary in the wavelet domain, and $\mathbf{S}^w$ is the sparse code matrix. In this chapter, we also exploit the sparsity in the wavelet domain but in a nonlocal way.

In the theory of compressive sensing (108), the sparse codes are usually not randomly located but clustered. This can be easily extended to the wavelet domain - the image self-similarity exists in the spatial domain, so there is a strong possibility that the nonlocal characteristic also exists in the wavelet domain for subbands from the same decomposition level. Therefore, for wavelet patches from the same scale, we propose to train different sub-dictionaries in different underlying clusters. In this way, it is then easy to obtain a sparse representation in the form that wavelet patches with similar structure will be represented by dictionary atoms from the same positions but the sparse codes are different (91)(67)(1). This would improve the coding efficiency and reduce artifacts. In this chapter, we propose to utilize clustering to obtain training samples for each sub-dictionary. For each decomposition level q, the dictionary $\mathbf{D}_{qj}^w \in \mathbb{R}^{M\times r_{qj}}$ for each cluster j is obtained from a regularizer with l_1 norm:

$$\begin{aligned} l(\mathbf{S}_{qj}^w, \mathbf{D}_{qj}^w) = \arg\min_{\mathbf{D}_{qj}^w,\mathbf{S}_{qj}^w} & \|\mathbf{Y}_{qj}^w - \mathbf{D}_{qj}^w\mathbf{S}_{qj}^w\|_2^2 + \lambda\|\mathbf{S}_{qj}^w\|_1 \\ & s.t.\|d_{qji}^w\|_2^2 \le b, i = 1,2,...,r_{qj} \\ & j = 1,2,...,K, q = 1,2,...,L \end{aligned} \tag{4.7}$$

where matrix $\mathbf{Y}_{qj}^w = [\mathbf{y}_{qj_i}^w, ..., \mathbf{y}_{qjN_q}^w] \in \mathbb{R}^{M\times N_q}$ represents the input wavelet coefficient patches in the cluster j of decomposition level q, $\boldsymbol{S}_{qj}^w = [\boldsymbol{s}_{qj1}^w, ..., \boldsymbol{s}_{qjN_q}^w] \in \mathbb{R}^{r_q\times N_q}$, and d_{qji}^w denotes the i-th column of the dictionary. In terms of patch denoising after the dictionary is learned, two approaches are discussed in this chapter. One simple method is patch reconstruction in each cluster using the trained sub-dictionary (we define this as NHDW1). However, due to the hierarchical structure of the wavelet coefficients, the lower subbands usually have far less training samples compared to the higher subbands, and therefore the sub-dictionaries are usually under-complete. The other method is to concatenate all the sub-dictionaries in the same scale (NHDW), which addresses the problem of insufficient training samples. The intuition behind this is that the performance of the dictionary learning system improves with the dictionary redundancy (106). Both NHDW1 and NHDW are compared in the experiment section, and we show that the latter is better for our proposed denoising framework.

After the individual training within each cluster, the scale dictionary for each decomposition level q can be obtained by merging the above small dictionaries into an over-complete dictionary $\mathbf{D}_q^w \in \mathbb{R}^{M \times r_q}$:

$$\mathbf{D}_q^w = [\mathbf{D}_{q1}^w, ..., \mathbf{D}_{qj}^w, ..., \mathbf{D}_{qK}^w] \tag{4.8}$$

In the above proposed learning framework, wavelet coefficients coming from the same decomposition level have been considered. However, along the same orientation, the lower subbands and higher subbands are correlated (Fig. 4.1). If a coefficient is small in the lower subband, its descendants in the higher subbands tend to be small accordingly. This motivates us to apply our proposed system to a new direction of wavelet coefficients.

In the wavelet domain, there are three orientations: LH, HL, HH. Along each orientation, patches with the size $m \times m$ are extracted from all the subbands in all different scales with maximum overlapping (similar to the grouping that has been suggested in (107)). The following dictionary learning and image reconstruction can be accomplished through the same steps as shown in Sec. 4.4.3. The implementation and parameter details are explained in Sec. 4.5.

One advantage of this variation is that the number of the training samples in a single orientation is larger than that in a decomposition level, especially for lower subbands. However, the nonlocal grouping between patches from different scales cannot guarantee the best way to exploit the correlations between different subbands in different scales.

4.4.2 Iterative Reweighted Regularizer

For the regularization term, it is suggested in (1) that l_1 norm can perform better in terms of dictionary training, and the l_0 norm can reconstruct the denoised image better.

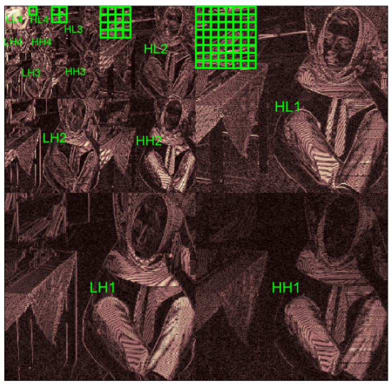


Figure 4.1: The whole image has shown the Wavelet subbands for the input noisy image. The green highlighted tree structure is for describing a coefficient in subband HL4 and its subsequent higher subbands in HL3, HL2, HL1.

In this chapter, we adopt this strategy.

The accuracy of the clustering stage in our method largely depends on how noisy the input wavelet coefficients are. From the experience of previous work (1), (14), one can see that better grouping can be achieved with iterative schemes or pre-denoising before classification. Therefore, we propose to integrate the l_1 reweighted regularizer with our dictionary learning process in an iterative scheme.

The regularization parameter λ (Eq. (4.7)) controls the balance between the fidelity term and the sparsity term. Assuming that $\mathbf{X}_{qj}^{w(t)} = [\mathbf{x}_{qj_i}^{w(t)}, ..., \mathbf{x}_{qj_{N_q}}^{w(t)}] \in \mathbb{R}^{M \times N_q}$ has already been denoised (M is the size of the patch vector and N_q is the number of the cluster members in decomposition level q), in the following iterations the regularization parameter between the fidelity term and the sparsity term should be altered with a more adaptive value $\boldsymbol{\theta}$. This can be expressed as:

$$\begin{aligned} \min_{\boldsymbol{s}_{qj}^{w} \in \mathbb{R}^{r_q}} \frac{1}{2} \|\boldsymbol{x}_{qj}^{w(t)} - \boldsymbol{D}_{qj}^{w(t)} \boldsymbol{s}_{qj}^{w}\|_2^2 + \gamma \|\boldsymbol{\theta}^{(t)} \boldsymbol{s}_{qj}^{w}\|_1 \\ s.t. j = 1, 2, ..., K \\ q = 1, 2, ..., J \\ t = 1, 2, ..., P \end{aligned} \tag{4.9}$$

where $\boldsymbol{D}_{qj}^{w(t)} \in \mathbb{R}^{M \times r_q}$, $\boldsymbol{\theta}^{(t)}$ is the diagonal matrix with $\theta_1^{(t)}, ..., \theta_{N_q}^{(t)}$ on the diagonal and zeros elsewhere. The above reweighting strategy has been proposed in the compressive sensing field (109) that "the new parameter should be inversely proportional to the underlying signal magnitude". In our algorithm, we adopt this as "the new parameter should be inversely proportional to the underlying sparse code":

$$\theta_i^{(t+1)} = \frac{1}{|s_{qj_i}^{w(t)}| + \varepsilon} \qquad i = 1, ..., r_q \tag{4.10}$$

where ε is a very small constant. The intuition for this reweighted process is that a sparse code that has a small value after t iterations but is not exactly zero will have a large reweighting factor $\theta_i^{(t+1)}$ in the next iteration. This would result in a sparser representation.

4.4.3 Denoising Algorithm

In the wavelet domain, most important information of the image is concentrated at the lowest subband (LL), and it is robust to noise. So simple soft-thresholding (109) is applied on the LL coefficients. The rest of the subbands are processed by our Nonlocal Hierarchical Dictionary learning using Wavelets (NHDW) algorithm, as shown in Fig. 4.2.

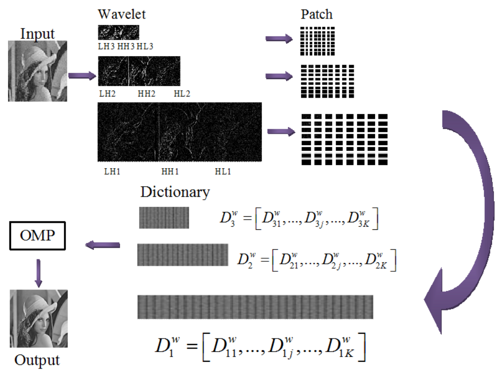

Wavelet
Patch
Input
LH3 HH3 HL3
LH2 HH2 HL2
LH1 HH1 HL1
Dictionary
$D_3^w = [D_{31}^w, ..., D_{3j}^w, ..., D_{3K}^w]$
OMP
$D_2^w = [D_{21}^w, ..., D_{2j}^w, ..., D_{2K}^w]$
$D_1^w = [D_{11}^w, ..., D_{1j}^w, ..., D_{1K}^w]$
Output

Figure 4.2: Flowchart of our proposed method. From the input to wavelet coefficients, it has been done according to Eq. (4.11). Patches are extracted from wavelet coefficients. Afterwards, the patches have been clustered following Eq. (4.12). Within each cluster, the dictionary is learned through Eq. (4.13). After all the cluster dictionaries are trained, the scale dictionary is combined as $D_i^w, i = 1, 2, 3....$ In the reconstruction stage, patches within the same scale are restored by Eq. (4.14) (4.15).

4.4.3.1 Input

The noisy image $\boldsymbol{y}$, standard deviation of the Gaussian noise σ (we use Gaussian noise here as an example).

4.4.3.2 Parameters

- The iteration times for the dictionary learning process P;
- The patch size of the wavelet coefficients $M = m \times m$;
- The type of the wavelet transform;
- The number of dictionary atoms per dictionary for each decomposition level r_l;
- The K for k-means clustering
- The initial γ for the $l_1$regularization and the initial $\boldsymbol{\theta}$

4.4.3.3 Initialization

The initial dictionaries can be learned from the online process in (110), or fixed basis transform matrices like Discrete Cosine Transform (DCT). Apparently, the pre-learned dictionary (learned from the clean images) is a good initial dictionary for the denoising process (1). However, in this chapter, we use DCT as the initial dictionary, planning to demonstrate that our method can exploit the image self-similarity well without the help of external dictionary elements from image datasets.

4.4.3.4 Wavelet decomposition

The 2-D wavelet transform is applied to the noisy input image $\mathbf{y}$:

$$\boldsymbol{y}^w = \mathbf{W}\boldsymbol{y} = \boldsymbol{x}^w + \boldsymbol{\eta}^w \tag{4.11}$$

For each decomposed level $q, (q = 1, 2, ...J)$, patches with the size $m \times m$ are extracted from $\boldsymbol{y}^w$ with maximum overlapping (patch stride size 1).

4.4.3.5 Dictionary Learning

In this learning process, for each wavelet decomposition level, repeat P times:

- K-means clustering: The wavelet coefficients patch matrix $\boldsymbol{y}_q^w$ will be clustered by k-means clustering into K clusters $\boldsymbol{y}_{qj}^w, (q = 1, 2, ...J, j = 1, 2, ..., K)$.

$$\arg\min_{Hm} \sum_{j=1}^{K} \sum_{H(\boldsymbol{y}_{qj_i}^w) \in Hm_j, i=1,2,...,N_j} |H(\boldsymbol{y}_{qji}^w) - \mu_j|^2 \tag{4.12}$$

  where $\boldsymbol{y}_{qji}^w$ represents a noisy patch i belonging to cluster j decomposition level q, and μ_j is the mean vector for the jth cluster Hm_j. Then, we can obtain K clusters Hm_1, Hm_2, Hm_j, ..., Hm_K. Each cluster, Hm_j is composed of N_j vectors.

- Sub-dictionary training:

  Within each cluster $Hm_j, (j = 1, 2, ..., K)$, the optimization of the dictionary is between the dictionary and the sparse codes alternatively:

  - Initialization: the sub-dictionary is initialized with DCT coefficients.
  - repeat P times:
    * Sparse Coding Stage - fix the dictionary, update the sparse codes: use the reweighted least angle regression (LARS) (111) to compute the sparse codes for the patches in each cluster.
    * Dictionary Update Stage - fix the sparse codes, update the dictionary: compute dictionary $D_{qj}^{w(t)}$ using $D_{qj}^{w(t-1)}$ as the initial value. The learning process is the same as the algorithm 2 in (110).

$$\begin{aligned} \mathbf{D}_{qj}^{w(t)} = \arg\min_{\mathbf{D}_{qj}^w} \sum_{i=1}^{t} \frac{1}{2} \|\boldsymbol{x}_{qj}^{w(i)} - \mathbf{D}_{qj}^w \boldsymbol{s}_{qj}^{w(i)}\|_2^2 + \\ \gamma \|\boldsymbol{\theta}^{(i)} \boldsymbol{s}_{qj}^{w(i)}\|_1 \\ s.t. j = 1, 2, ..., K \end{aligned} \tag{4.13}$$

  - Return the learned dictionary

Within each cluster $Hm_j, (j = 1, 2, ..., K)$, the sub-dictionary $\mathbf{D}_{qj}^w \in \mathbb{R}^{M\times r_q}$ is trained by the iterative method shown in (9) and (13). Afterwards, all the sub-dictionaries will be concatenated at each decomposition level as shown in (4.8). Now, we have over-complete dictionaries with different sizes for different decomposition levels $\mathbf{D}_q^w \in \mathbb{R}^{M\times r_l}$.

4.4.3.6 Image reconstruction

- Sparse coding: For the reconstruction stage, the dictionary for each decomposition level is known. Our goal is to find the sparse representation $\hat{s}_{ij}$ for each location and the overall output wavelet coefficients. The Orthogonal Matching Pursuit (OMP) (17) is used for obtaining sparse codes:

$$\begin{aligned} \hat{s}_{ij}^w &= \arg\min_{s_{ij}^w} \|s_{ij}^w\|_0 \\ s.t. \|\mathbf{D}_q^w s_{ij}^w &- R_{ij}\boldsymbol{x}_q^w\|_2^2 \le C\sigma \end{aligned} \tag{4.14}$$

where s_{ij}^w indicates the sparse codes of wavelet coefficient located at (i, j), C is a constant, σ is the standard deviation of the noise, R_{ij} is the binary matrix that extracts the (i, j) patch from the wavelet coefficient matrix, and $\mathbf{D}_q^w$ is the combined dictionary for decomposition level q.

- Reconstruction in the wavelet domain:

The final reconstructed wavelet coefficients matrix $\hat{\boldsymbol{x}}_q^w$ can be estimated by:

$$\hat{\boldsymbol{x}}_q^w = \arg\min_{\boldsymbol{x}_q^w} \lambda_1 \|\boldsymbol{x}_q^w - \boldsymbol{y}_q^w\|_2^2 + \sum_{ij} \|\mathbf{D}_q^w \hat{s}_{ij}^w - R_{ij}\boldsymbol{x}_q^w\|_2^2 \tag{4.15}$$

where $q = 1, 2, ..., J$.

- Image reconstruction:

$$\hat{\boldsymbol{x}} = W_s \hat{\boldsymbol{x}}^w; \tag{4.16}$$

where W_s is the inverse wavelet transform. Once we obtain the denoised patches in spatial domain $\hat{\boldsymbol{x}}$, we average the estimates of each pixel to reconstruct the final image.

4.5 Experimental Validation

In this section, our proposed algorithm is evaluated and compared with many state-of-the-art methods. Extensive experiments are conducted on noisy images which were produced by adding two types of synthetic noise to four standard grayscale images. [1] The methods we compared with are BLS-GSM [2], K-SVD [3], BM3D [4], and LSSC [5]. We used the codes provided by the authors with their optimal parameters in all these algorithms to enable a fair comparison.

4.5.1 Determination of Parameters

In our experiments, the "db1" wavelet basis was chosen with the decomposition level $J = 3$ for the wavelet decomposition (The type of the wavelets has been evaluated through our experiments. For instance, the wavelets mentioned in (107) were all tested, including "dmey", "sym8", "db8", etc.). We have empirically tested the K for k-means clustering from 5 to 20, and the best results appear at $K = 10$. So 10 clusters have been used in the following experiments.

In our implementation, the size of the dictionary atoms is fixed, which has been suggested in (107). We experimented to find the appropriate dictionary size by comparing the denoising results. The four images have been tested with noise level of $\sigma = 20$. Average results of 10 executions are presented in Fig. 4.3 and Fig. 4.4.

In each decomposition level, the number of the atoms in each sub-dictionary varies but the size of dictionary atoms is $m \times m$ ($m = 8$) as suggested in Ophir *et al.* 's work (107). Regarding the size of the dictionary, we based our experiments on the assumption that the sizes of the dictionaries in different scales are independent. We assume the

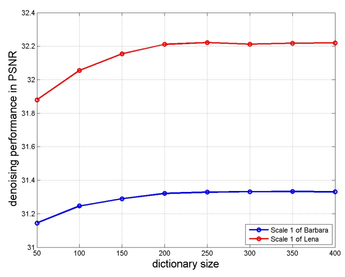
[1]The four images are: "Barbara", "Lena", "Monarch", "Straw". For each denoising experiment, the input of the denoising process is the noisy image while no extra training images are used.

[2]Using the code provided at http://decsai.ugr.es/ javier/denoise

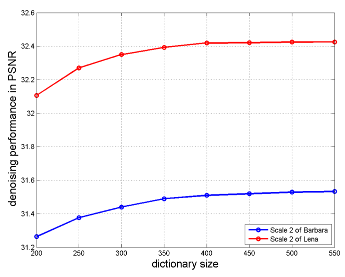
[3]Using the code provided at http://www.cs.technion.ac.il/ elad/software

[4]Using the code provided at http://www.cs.tut.fi/ foi/GCF-BM3D

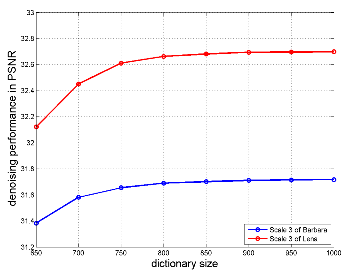
[5]http://www.di.ens.fr/ mairal/software.php



(a) Scale 1

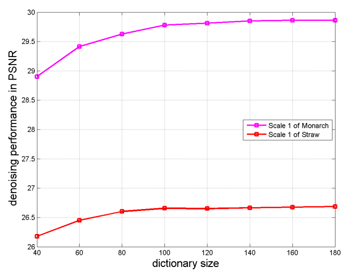


(b) Scale 2

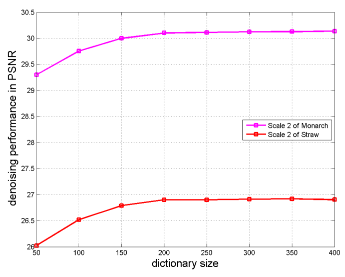


(c) Scale 3

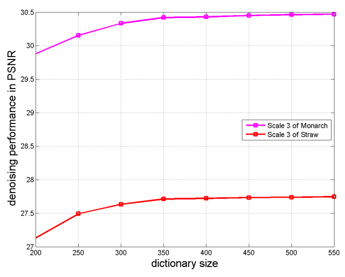
Figure 4.3: Average denoising results varying with the size of the dictionary in each decomposition level for the test images Barbara and Lena (512 × 512) at noise level $\sigma = 20$ (Gaussian noise).



(a) Scale 1



(b) Scale 2



(c) Scale 3

Figure 4.4: Average denoising results varying with the size of the dictionary in each decomposition level for the test images Monarch and Straw (256 × 256) at noise level $\sigma = 20$ (Gaussian noise).

estimated optimal value is 256 for each scale, which is the case in (112). For each scale, experiments have been done with different dictionary sizes varying from 50 to 2000 with the interval 50 when the other two dictionaries are fixed with size 256. In general, the increase of the number of the dictionary elements generally improves the results, but when the size is too large for the training samples overfitting is not avoidable. On the other hand, if the dictionary size is too small, underfitting will cause poor performance. Therefore, for each scale, there is a steady region of the optimal size, which is shown in Fig. 4.4,4.3. (In these two figures, the curves of the performance change in the 'bell' shape. When the dictionary size is too small, underfitting cause very low performance. When the dictionary size is too large overfitting cause the performance drop too. There is a steady region for optimum value. In out figures, we zoom in the figures to show the steady regions.) As can be seen, in these steady regions the performance's variation is within 0.12 dB. Considering the computational complexity, we pick the smallest dictionary size in these steady regions. For instance, if the input image is in size 512×512, we empirically choose the number of atoms of sub-dictionary for each cluster as 20, 40, 80. Therefore, the highest level has the dictionary with the size of 64×200, the middle level dictionary has the size of 64×400, and the lowest level has a dictionary with the size of 64×800.

The initial γ in (4.10) is set to be 0.15 and the θ is initialized with $\theta^{(0)} = I$. The number of iterations $P = 9$. In (4.15), at higher levels of noise ($\sigma \geq 50$) the noise gain was $C = 1.05$ (17) and at lower noise levels ($\sigma < 50$) it was $C = 1.15$ in our experiments.

In our experiments, two other variations are compared with our proposed method. The first one uses sub-dictionaries for the patch reconstructions (NHDW1), for which the sparse codes and sub-dictionaries are generated in framework as shown in Sec.4.4.3.5. The parameters are the same with the previous description but there is no OMP stage in Sec. 4.4.3.6.

The second one is training orientation dictionaries instead of scale dictionaries (NHDW2). According to the empirical testing, all the parameters in NHDW2 are still the same as the settings in NHDW except for the dictionary size. We use the same method to decide the dictionary size and the best number of atoms is 300 for each orientation dictionary.

4.5.2 Results

As shown in Table 4.1, BM3D and LSSC produce better results than K-SVD and BLS-GSM because of their nonlocal merits. For the averaging filter in the denoising process, if there are more similar patches, especially those located in the image itself, the averaging results will be better off because the noise will be cancelled out during the weighted averaging. In the proposed scheme, the nonlocal weighted averaging is conducted on wavelet coefficients, which benefits from both the nonlocal property and the wavelet multi-scale advantage. As shown in Table 4.1, our method is competitive compared to BM3D and LSSC in middle or low noise levels and outperforms both of them under high levels of noise. Compared to the two variations mentioned before, NHDW is better than both of them at all noise levels. NHDW1 performs worse than NHDW because the under-complete dictionary slightly causes the reconstruction artifacts. NHDW2 generates worse results because the size differences between features in different scales result in the clustering difficulty.

From Fig. 4.5, one can see that our method has a very similar visual result compared to LSSC and BM3D when the standard deviation of the noise is 20. However, when the sigma is 50, it is easy to spot in Fig. 4.6 that the textures denoised using our method are sharper than that denoised using the other methods listed. In Fig. 4.7, the edges are well preserved and least artifacts exist in the result of our proposed method. It is easy to see that BM3D does fail to produce good results when the noise level is high because of the erroneous grouping. LSSC still recovers some spurious structures in the smooth regions as 'image structure', which appear to be artifacts in low-frequency regions. In our method, this deficiency is overcome by the multi-scale structure of the wavelets. On the other hand, NHDW1 generates less artifacts compared to NHDW2, which shows that constructing dictionaries along the scale direction is better than the orientation directions. Our proposed NHDW has better visual results than NHDW1 due to the combined contribution of the over-complete dictionary, the l_1 norm for dictionary learning, and the l_0 norm for image reconstruction.

Table 4.1: Comparison of state-of-the-art image denoising methods on various images corrupted by Gaussian noise. The results shown are the average PSNR values obtained by the methods over ten independent noise simulations for each standard deviation σ.

σ	10	20	25	50	75	10	20	25	50	75
	Barbara (512×512)					Monarch (256×256)				
BLS-GSM	33.12	29.08	27.80	27.02	22.95	33.79	29.77	28.55	25.94	22.82
K-SVD	34.82	31.11	29.8	26.93	23.20	33.74	30.00	28.91	25.34	22.81
BM3D	34.98	31.78	**30.71**	27.22	25.12	34.12	30.35	29.25	25.82	23.91
LSSC	**35.36**	**31.82**	30.66	27.06	25.14	**34.49**	30.71	**29.52**	26.54	24.76
NHDW1	35.01	31.66	30.52	27.17	25.06	34.25	30.45	29.36	26.47	24.44
NHDW2	35.02	31.53	30.49	27.01	24.98	34.08	30.30	29.23	25.99	24.02
NHDW	35.34	31.79	30.70	**27.31**	**25.22**	34.48	**30.72**	29.49	**26.72**	**24.99**
	Lena (512×512)					Straw (256×256)				
BLS-GSM	35.24	32.24	31.26	28.19	26.45	30.51	26.26	24.99	21.53	19.99
K-SVD	35.63	32.67	31.67	28.61	26.85	30.99	26.95	25.70	21.52	19.45
BM3D	**35.93**	**33.05**	**32.07**	29.05	27.25	30.92	27.08	25.89	22.41	20.72
LSSC	35.83	32.91	31.88	28.87	27.16	31.51	27.50	26.21	23.05	21.62
NHDW1	35.63	32.72	31.92	28.87	27.10	31.45	27.48	26.33	22.88	21.55
NHDW2	35.57	32.21	31.74	28.53	26.91	31.32	27.39	26.18	22.68	21.29
NHDW	35.89	32.99	32.02	**29.06**	**27.39**	**31.70**	**27.72**	**26.68**	**23.38**	**21.91**

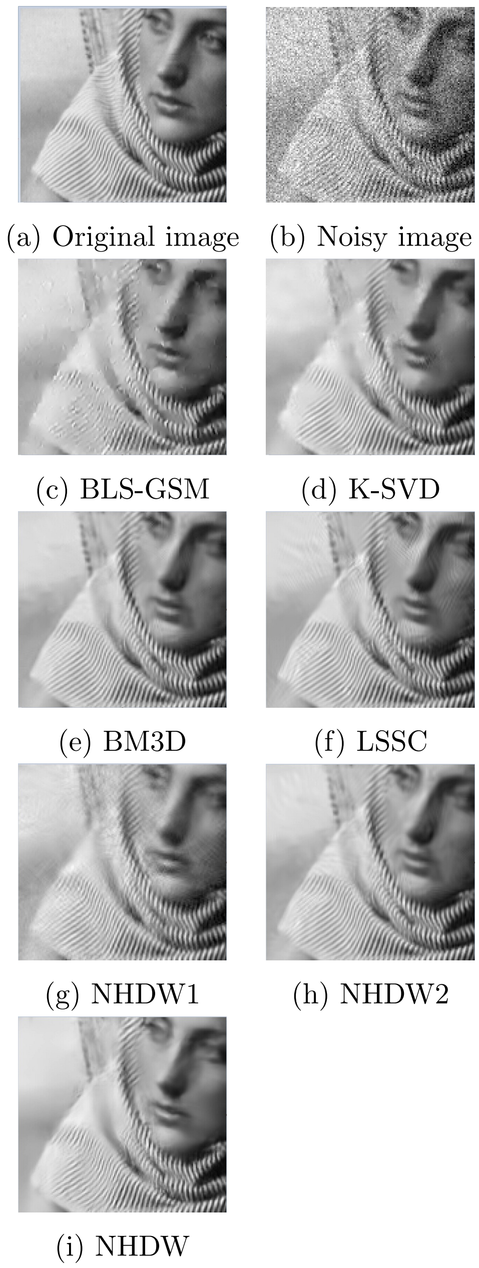
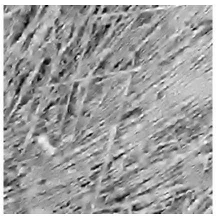
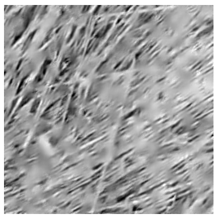


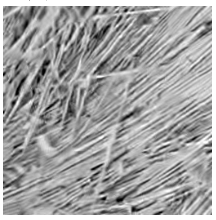
Figure 4.5: Denoising results for the image “Barbara” with $\sigma = 25$ (Gaussian noise).

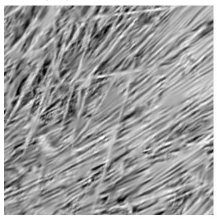
(a) Original image

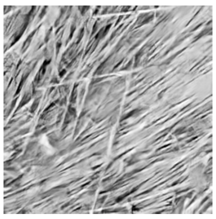
(b) Noisy image

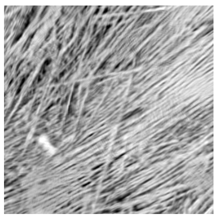

(c) BLS-GSM


(d) K-SVD


(e) BM3D


(f) LSSC


(g) NHDW1


(h)NHDW2

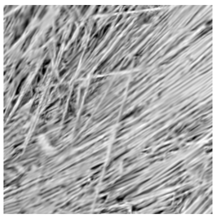
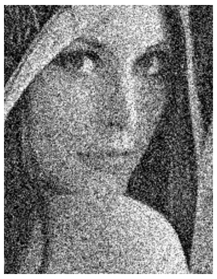
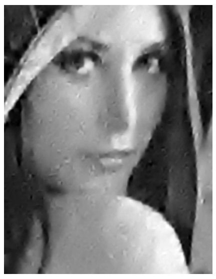
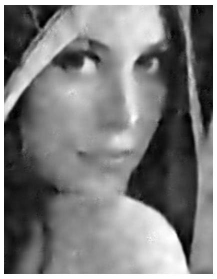

(i) NHDW

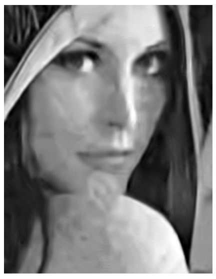
Figure 4.6: Denoising results for the image “Straw” with $\sigma = 50$ (Gaussian noise).

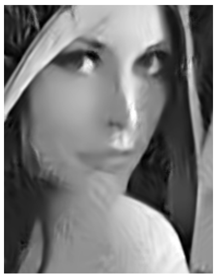
(a) Original image

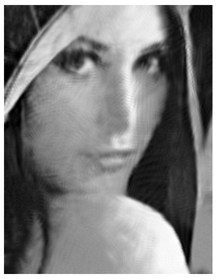

(b) Noisy image

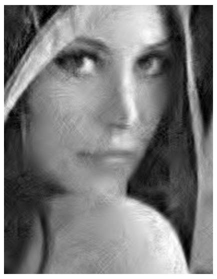

(c) BLS-GSM


(d) K-SVD


(e) BM3D


(f) LSSC


(g) NHDW1


(h) NHDW2

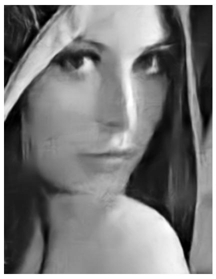

(i) NHDW

Figure 4.7: Denoising results for the image "Lena" with $\sigma = 50$ (Gaussian noise).

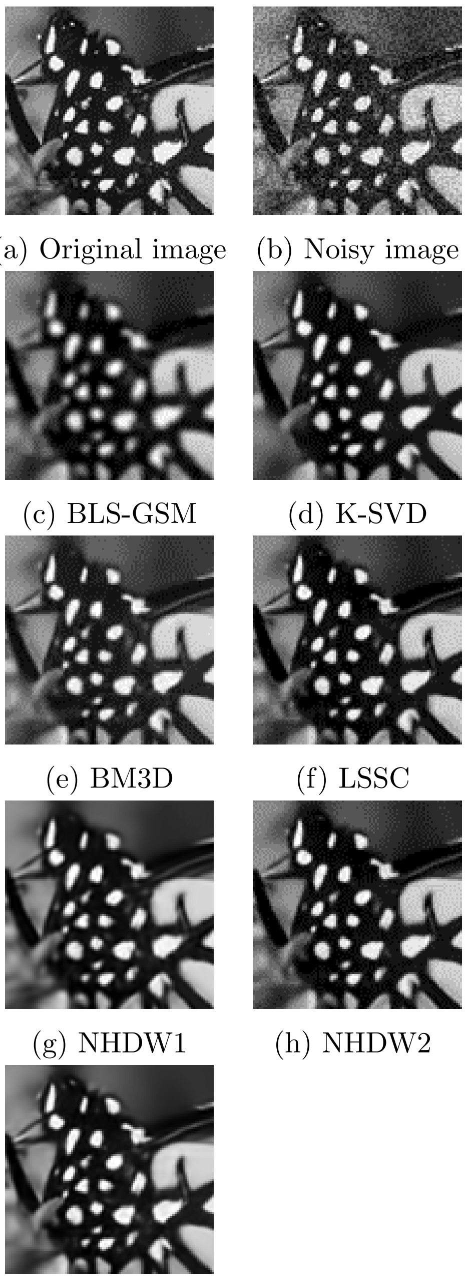
In our experiments, we also evaluated the proposed NHDW method using the uniform noise model, which can be defined as:

$$p(z) = \begin{cases} \frac{1}{2a}, & \text{if} - a \leq z \leq a, \\ 0. & \text{otherwise.} \end{cases} \tag{4.17}$$

for $a > 0$. Its variance is $\sigma^2 = (a^2)/3$. As shown in Table 4.2 and Fig. 4.8, our proposed NHDW performs better than the state-of-the-art denoising methods in the presence of uniform noise.

Table 4.2: Comparison of State-of-the-art Image Denoising Methods on Various Images Corrupted by Uniform Noise. The results shown are the average PSNR values obtained by the methods over ten independent noise simulations for each a.

a	±10	±20	±30	±40	±10	±20	±30	±40
	Barbara (512 × 512)				Monarch (256 × 256)			
BLS-GSM	34.10	27.99	24.34	23.29	33.15	29.32	25.34	24.69
K-SVD	35.13	29.89	26.64	24.47	34.50	29.62	27.21	25.63
BM3D	36.43	32.98	31.13	29.82	35.82	31.51	29.56	28.32
LSSC	**36.72**	32.53	29.85	29.77	35.81	30.99	29.68	27.98
NHDW1	36.08	32.75	31.02	29.48	35.22	31.15	29.16	27.98
NHDW2	35.21	32.05	30.38	28.55	34.76	30.59	28.50	27.63
NHDW	36.69	**33.49**	**31.73**	**30.33**	**35.94**	**31.82**	**29.91**	**28.66**
	Lena (512 × 512)				Straw (256 × 256)			
BLS-GSM	34.03	30.61	28.95	27.33	32.22	26.85	23.21	22.08
K-SVD	35.39	31.76	29.41	27.72	31.89	26.12	22.78	19.96
BM3D	**36.84**	34.10	32.52	31.44	33.46	28.07	25.71	24.30
LSSC	36.41	32.88	31.27	29.05	33.23	27.37	24.55	23.38
NHDW1	35.79	34.21	32.17	31.01	33.07	27.82	25.49	23.99
NHDW2	35.15	33.21	31.66	30.24	32.24	27.11	24.95	23.14
NHDW	36.53	**34.99**	**32.88**	**31.75**	**33.51**	**28.44**	**26.23**	**24.75**



(a) Original image (b) Noisy image

(c) BLS-GSM (d) K-SVD

(e) BM3D (f) LSSC

(g) NHDW1 (h) NHDW2

(i) NHDW

Figure 4.8: Denoising results for the image "Monarch" with $a = 30$ (uniform noise).

4.6 Conclusion

This work takes advantages of the sparse coding framework, nonlocal grouping and wavelet transform, leading to the state-of-the-art denoising performance. The proposed method builds a nonlocal hierarchical sparse dictionary on the wavelet coefficients of a noisy image. Within each layer of the multi-scale dictionary, the k-means clustering serves as a tool to exploit sparsity not just within a subband but also between subbands of the same scale. The reweighted l_1 norm is used in the regularization in each cluster to find the best sub-dictionary. Once the dictionaries are trained, they are combined as a whole to represent the entire decomposition level. Our main contributions are: 1) a hierarchical sparse dictionary in the wavelet domain; 2) the reweighted training strategy to improve the performance.

5

Image Denoising using Genetic Programming

5.1 Chapter Abstract

The coefficients in previous local filters are mostly heuristically optimized, which leads to artifacts in the denoised image when the optimization is not adaptive enough to the image content. Compared to parametric filters, learning-based denoising methods are more capable of tackling the conflicts between noise reduction and artifact suppression. In this chapter, a patch-based Evolved Local Adaptive (ELA) filter is proposed for natural image denoising. In the training process, a patch clustering is used and the Genetic Programming (GP) is applied afterwards for determining the optimal filter (linear or nonlinear in a tree structure) for each cluster. In the testing stage, the optimal filter trained beforehand by GP will be retrieved and employed on the input noisy patch. In addition, this adaptive scheme can be used for different noise models. Extensive experiments verify that our method can compete with and sometimes outperform the state-of-the-art local denoising methods in the presence of Gaussian or salt-and-pepper noise. Additionally, the computational efficiency has been improved significantly because of the separation of the offline training and the online testing processes.

This chapter is based on the following work:

R. Yan, L. Shao, L. Liu, and Y. Liu, "Natural image denoising using evolved local adaptive filters", *Signal Processing*, vol. 103, pp. 36-44, Oct. 2014.

5.2 Introduction

Image denoising is the process of recovering the underlying clean image from an observation that has been corrupted by various noises. Due to the fact that image quality is critical for later high-level applications (e.g., object detection), denoising is a very popular topic in the image processing field (9, 54).

The existing image denoising techniques can be divided into heuristically optimized and non-parametric methods. In the first category, there are linear and nonlinear filters. Linear filters are widely applied because of their low cost. However, they tend to be ineffective in the presence of non-Gaussian noise. On the other hand, nonlinear filters are used to overcome the limitations of linear filters (113), for instance, nonlinear filters have better edge-preserving ability. However, most filters, either linear or nonlinear, are optimized through tedious tuning and testing (114, 115). Since the Gaussian filter was applied to image denoising, many local filters have been proposed to improve it. The anisotropic filter (38) was proposed to smooth the image only in the direction which is orthogonal to the gradient direction in order to reduce the blur effect of the Gaussian filter. The method in (68) utilizes the total variation minimization technique to smooth the homogenous regions of the image but not its edges. Similarly, the bilateral filter (37) can average pixels in the local neighborhood, which are from the same range as the central pixel, for improving the edge-preserving ability of the Gaussian filter. Nevertheless, the denoising performance of the bilateral filter depends highly on the parameter optimization.

The recent local filters which produce impressive results are mostly nonparametric. Similar to the idea of early local filters (33, 38, 69), a weighted averaging scheme is adopted to perform image denoising in the trained filter (36) with the difference that the weights are obtained from off-line training on a large number of images. Portilla *et al.* (16) proposed a Gaussian scale mixture model based on a multiscale wavelet decomposition for local image statistics (BLS-GSM). Instead of fixed basis local representations, K-clustering with Singular Value Decomposition (K-SVD) (17) achieves good denoising results by adaptive learning of a dictionary from the noisy image. Each patch can be represented by elements from the dictionary.

In real applications, the noise model is varied, such as impulse, uniform, Gaussian and mixed noise(41). Therefore, learning-based methods are desirable because they can be adaptive based on the training data (7, 64). However, most of them can only provide a linear solution while most real degradations are not necessarily linear (17) (1). Accordingly, we address this problem by developing a local adaptive learning-based denoising method which can generate both linear and nonlinear estimations.

Genetic Programming (GP) is a branch of Evolutionary Computation (EC) that stochastically transforms populations of programs into a new, better population of programs. As a random process, the results of GP can never be guaranteed, however, unexpected solutions can be generated which is beyond the human expert's consideration. GP has already been successfully used in image denoising (116, 117). Petrovic *et al.* (118) have developed a successful GP-based denoising method, which involves noise detector and noise remover for the impulse noise. However, their method was not designed for other noise models, for instance, Gaussian noise.

In this chapter, we propose a patch-based image denoising method that is learned from training data by genetic programming. In the training stage, a patch clustering is applied first to classify the image content before the GP process. The filter evolved by GP is more adaptive to the local image content in a linear or nonlinear form. Different from the existing GP-based denoising method (118), our function set is composed of local filters (e.g., Gaussian and bilateral filters) and arithmetic operators (e.g., addition, substraction, multiplication, division), which are more adaptive to various image contents because the arithmetic operators enable the random combinations within the local filter candidate set. In addition, though the offline training process is not very efficient, the online testing phase is faster than most of the state-of-the-art local methods. Results on additive noise reduction are comparable with the state-of-the-art. Furthermore, our proposed scheme can be extended to other noise models (e.g., salt-and-pepper noise) and other image enhancement tasks, which makes our method more versatile compared to previous similar work (e.g., (118)).

5.3 Related Work

5.3.1 Bilateral Filter

Bilateral filter is a nonlinear filter which removes the noise from images and preserves the edges (37). It can be formulated by the following equation:

$$\hat{\mathbf{x}} = \frac{1}{w(\tau)} \int\int_{\xi \in D} g(\xi)c(\xi,\tau)s(f(\xi), f(\tau))d\xi \tag{5.1}$$

where $\hat{\mathbf{x}}$ is the restored image, $s(f(\xi), f(\tau))$ is the similarity weight between pixel ξ and pixel τ. $c(\xi,\tau)$ is the weight based Euclidean distance between those two pixels. Actually, bilateral filter is the combination of a domain filter and a range filter, which can be expressed by the following equations separately:

$$c(\xi,\tau) = \exp^{-\frac{\|\xi-\tau\|^2}{2\sigma_d^2}} \tag{5.2}$$

$$s(g(\xi), g(\tau)) = \exp^{-\frac{\|f(\xi)-f(\tau)\|^2}{2\sigma_r^2}} \tag{5.3}$$

where σ_r and σ_d are the standard deviations of the range filter and domain filter respectively. For the domain filter, pixels which are spatially close to the current one are given high weights. For the range filter, pixels which are similar to the reference pixel have higher weights. In this way, the weighted averaging process is done mostly along the edge and greatly reduced in the gradient direction.

5.3.2 A brief introduction to genetic programming

"Genetic programming is an evolutionary method which has been extensively used to evolve programs or sequences of operations" (119). The basic workflow of a genetic programming algorithm is illustrated in Fig. 5.1. GP can determine whether a program is good by running it and evaluate the fitness function. By comparing the fitness of different individual programs, GP can select the best program from the current population. It is also able to create new computer programs by mutation and crossover. The iteration process of selection/crossover/mutation is carried on or a certain number of times or it will stop until some target is met.

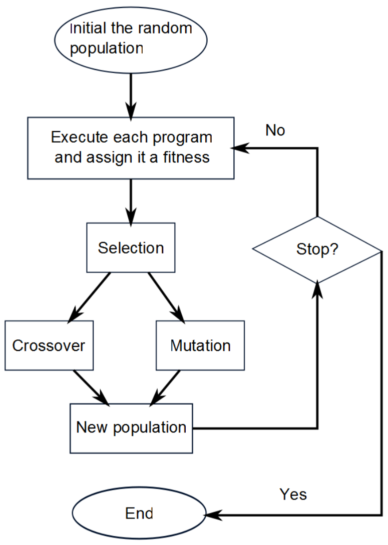

Initial the random population
Execute each program and assign it a fitness
Selection
Crossover
Mutation
New population
Stop?
No
Yes
End

Figure 5.1: The GP work flow

5.4 Methodology

In this chapter, our objective is to use genetic programming to generate a novel local adaptive filter for various image contents. The outline of our algorithm is illustrated in Fig. 5.2.

The proposed ELA algorithm is a supervised learning process that includes classification based off-line training and the online denoising procedure. The original images are first corrupted with additive Gaussian noise. Patch clustering is applied to these corrupted images to group the patches with similar local structures. Within each class, an optimal filter is evolved by GP. The filter itself is an individual in the form of a tree structure, illustrated in Fig. 5.3.

Suppose x_i is a patch of the original high quality image centered at pixel i and y_i is the corresponding patch in the corrupted image for a particular class k. Then the filtered patch $\hat{x}_i$ can be obtained by the desired optimal filter for class k as follows:

$$\hat{x}_i = F_k(y_i) \tag{5.4}$$

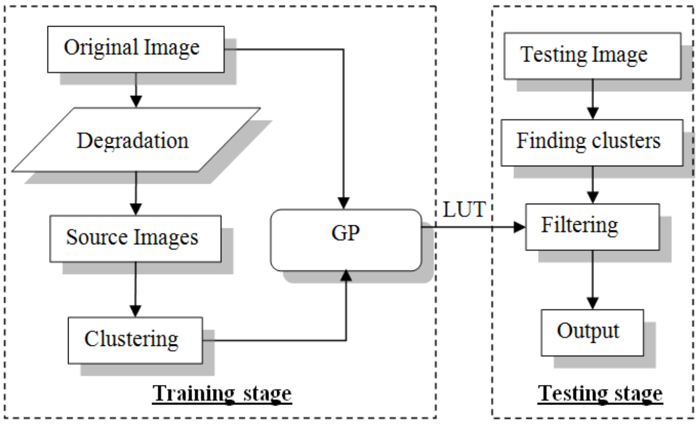


Figure 5.2: The ELA-filter framework

where F_k represents the desired filter for class k generated by GP. Then the filter estimation is to minimize the reconstruction error, which is shown in eq.(5.8) (5.9).

Fig. 5.2 shows the filtering process after the GP filter is trained. For a given testing noisy image, patches are extracted from the image. After calculating the distances between the SVD feature of the current patch and the cluster centers established in the offline process, the corresponding coefficients are retrieved for filtering.

5.4.1 Patch clustering

In this stage, our algorithm attempts to perform clustering to group patches with similar textures, such as strong edges or smooth regions. Previous clustering methods exploit different features from images, such as, pixel intensities, gradients, or a combination of these (91). However, in the case of image denoising, the disturbance from the noise requires the structural features to be robust. On the other hand, considering that the filter elements we use in the function set are not sensitive to the orientation of the edge, a clustering method that is invariant to rotation can be adopted. Singular Value Decomposition (SVD) (45) is proposed to extract features from input noisy images, and then the K-means clustering (K-means) technique is proposed for patch clustering based on those features. The intuition for this clustering is: the magnitude of the singular value in the dominant direction of each patch is used for the classification because the Gaussian noise is non-directional. And, the magnitude of the singular value indicates

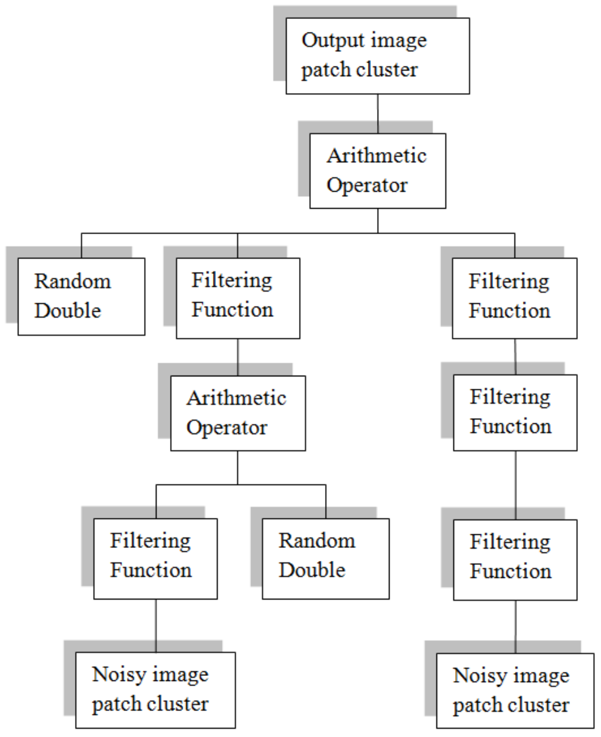


Figure 5.3: The GP tree structure

whether this patch is in the smooth region or an edge/texture region. This is what we need in our clustering.

For a patch centered at i with the size $n \times n = N$, we can express its gradient values with a matrix G_i and accordingly the SVD can be calculated as:

$$G_i = [\nabla y_i(1)^T \nabla y_i(2)^T \dots \nabla y_i(N)^T]^T, G_i = U_i S_i V_i^T \tag{5.5}$$

where

$$\nabla y_i(j) = [\frac{\partial y_i(j)}{\partial \alpha} \frac{\partial y_i(j)}{\partial \beta}]^T \tag{5.6}$$

is the gradient of patch y_i at point $j, j = 1, 2, ..., N$. The gradient of image $\mathbf{y}$ at point i (G_i) can be decomposed into three matrices: U_i is an $N \times N$ orthogonal matrix, V_i is a 2×2 orthogonal matrix which gives the dominant orientation of the gradient field, and S_i is an $N \times 2$ matrix that contains singular values. Therefore, the patches within a dataset can be clustered as (45):

$$\arg\min_{c} \sum_{k=1}^{K} \sum_{w(y_{ki}) \in Wm_k i=1,2,...,N_k} |w(y_{ki}) - \mu_k|^2, \tag{5.7}$$

where y_{ki} represents a noisy patch centered at pixel i belonging to cluster k, $w(y_{ki})$ is the magnitude of the dominant orientation of the local gradient, and μ_k is the mean vector for the kth cluster Wm_k (Wm_k is the set of the magnitude of the dominant orientation of the local gradient). Then, we can obtain K clusters Wm_1, Wm_2, Wm_k, ..., Wm_K. Each cluster, Wm_k, is composed of N_k vectors Wm_{kq} ($k = 1, ..., K, q = 1, 2, ..., N_k$).

5.4.2 Training GP-based local filters for image denoising

In the training stage, the idea is to design an adaptive filter for any individual class to denoise patches with certain patterns. In the previous work by Shao et al. (36), a least squares optimization process is used for estimating the linear parameter model. However, in real applications, only using a linear filter for image denoising is not enough. For instance, linear filters are effective for removing additive Gaussian noise but at the same time tend to blur sharp edges and fail to effectively reduce the signal-dependent noise. On the other hand, nonlinear filters are capable of dealing with non-uniform smoothing that can be locally adapted to data features, such as removing impulsive noise and heavy tailed noise.

Our intuition for the proposed method is: after the input image has been classified by local structures, GP is used to determine whether a linear or nonlinear filter should be used for pixels with certain textures. More precisely, all the patches belonging to a class Wm_k are trained through the GP process. For each generation, they are filtered by the current best individual and compared with clean underlying patches using the fitness function (Eq. (5.8)(5.9)).

5.4.2.1 The function set

The function set of the GP process depends on the nature of the problem domain. Usually, the relation between the functions should be considered; the efficiency should be a concern because the evolving process of any individual tree is time-consuming.

Table 5.1: The functon set of the GP process.

Functions	Parameters	Description
Add	Image, Image	Adds the pixel values
Sub	Image, Image	Subtracts the pixel values
AbsSub	Image, Image	Absolute value of the difference
Abs	Image	Absolute value of an image
Med	Image, Window	Median value of the window
Bil1	Image, Window	The sigmas are [1.5 0.2]
Bil2	Image, Window	The sigmas are [1.6 0.29]
Bil3	Image, Window	The sigmas are [1.7 0.196]
Bil4	Image, Window	The sigmas are [1.8 0.186]
Bil5	Image, Window	The sigmas are [1.9 0.265]
Bil6	Image, Window	The sigmas are [2.1 0.176]
Ave	Image, Window	Average value of the window
Gau	Image, Window	The sigma is 0.09

In our implementation, the function set for the GP procedure includes several bilateral filters, the median filter, the average filter, the Gaussian filter, and the basic arithmetic functions. Most of the arithmetic functions are applied to two images, but the filters are used on single images (patches). Some of the functions are applied to two images and some are applied to one image as shown in Table 5.1. All of them return a single image as the output.

As suggested in (120), the optimal σ_d value in the bilateral filter (the standard deviation of the space Gaussian filter) is relatively insensitive to noise variance compared to the optimal σ_r value (the standard deviation of the range Gaussian filter). The best range of σ_d is $[1.5, 2.1]$. The six σ_r have been proved to be within the optimum settings for the bilateral filters (121). In our experiments, the values for σ_r and σ_d are empirically chosen for various image contents based on the suggested values from (120)(121).

These filtering functions operate on the raw image data. The reason why we choose the bilateral filter is that it has good edge-preserving ability and at the same time the complexity is not too high for GP evolution (37).

Meanwhile, the arithmetic functions are used for building better individuals, which has been shown in our experimental results and also in previous work (122).

5.4.2.2 The fitness function

The way we evaluate the filters is using a fitness function with the inverse of the average Peak Signal to Noise Ratio (PSNR) for the whole cluster:

$$fitness = \frac{N_k}{\sum_{l=1}^{N_k} PSNR_l}. \tag{5.8}$$

The individual PSNR for each patch is defined as:

$$PSNR_k = 10\log_{10}\left(\frac{NL^2}{\sum_{j=1}^{N}[x_i(j) - F_k(y_i(j))]^2}\right), \tag{5.9}$$

where L is the dynamic range of the pixel values, j represents the pixel location, N is the total number of pixels involved, $y_i(j)$ and $x_i(j)$ are the pixel values at point j. The best individual selected should have the minimal fitness value.

5.5 Experimental Results and Discussions

The whole GP process is implemented using the sample code from the Genetic Programming toolbox GPLAB [1] on Matlab 2011a. We used a six-core processor and 32 GB of RAM running Linux. The computationally intensive part in our proposed method is the bilateral filter, so we compiled the C++ code of the bilateral filter from OpenCV into a mex function, which speeds up the whole program by a factor of 30 times.

5.5.1 Dataset

Training set: The Berkeley segmentation dataset (123), which is composed of 200 images. All the images have been downsampled into 256×256 by bicubic interpolation and converted from color images to gray-scale images. It is worth mentioning that downsampling in our experiments would affect the experimental results. However, the reason for us to use it is that: some images in the dataset are already noisy or blurry, downsampling is supposed to help with improving the image quality (124). In our experiments, we have not used images with much higher resolutions (e.g. 1920 by 1080) because all our candidate functions in the function set have already been tested to have the best parameter range for images with patch size 9 by 9. If various

[1] http://gplab.sourceforge.net/download.html

resolutions are considered in our experiments, we need to update the parameters for those candidate functions.

Test data: 1) Standard test images: We used 8 images which are standard testing images for most state-of-the-art denoising algorithms, including "Barbara", "Boat", "Man", "Couple", "Hill", "House", "Lena", "Peppers". 2) Pascal VOC 2009: 100 images from this dataset have been selected.

5.5.2 Determination of Parameters

5.5.2.1 Parameters for the patch clustering

The patches we extracted from the dataset are all in size 9×9. Through experiments, we found out that the denoising results are almost the same when we use a patch offset of 1 or 3. So we choose the latter for the computational efficiency.

The number of clusters (K) can be tuned for our clustering process. Through our experiments, the best performance can be achieved by choosing 5 to 20 clusters for the 5 images we choose (For each experiment, we randomly choose the images from the training set.). Considering the performance and efficiency, the K for clustering is 15 in our implementation.

5.5.2.2 Parameters for the GP process

a) The training process would normally have 50 generations and 500 populations, which enables a random selection of GP for better adaptive filters. The initial population is generated by the ramped half-and-half method (125). b) In the literature, the mutation rates in other studies (e.g. that involves image processing) are also quite low, which is based on previous experiments where higher mutation rates were not as effective. So we choose mutation rate as 0.05. c) The selection method we used is "tournament", which draws a number of individuals from the population and selects only the best of them. The reason we choose this method is that our solution tree does not overfit easily given the large population. So the bloat controlling methods are not necessary in our case. On the contrary, they might cause the evolution diversity drop significantly in the very early generations. d) The "keepbest" is applied as our survival method, so the best individual from both parents and children is to be kept in the new population. e)

Table 5.2: The denoising results when images are corrupted by Gaussian noise with $\sigma = 25$.

image	BF		MBF		LARK		BLS-GSM		KSVD		Proposed	
	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
Barbara	25.86	0.851	26.03	0.855	26.22	0.841	27.80	0.899	29.79	0.915	**30.54**	**0.944**
Boat	27.16	0.845	27.72	0.845	28.43	0.847	29.26	0.890	29.28	0.880	**30.01**	**0.916**
Man	26.98	0.861	27.24	0.845	28.25	0.868	28.55	0.884	28.52	0.875	**29.25**	**0.902**
Couple	26.86	0.847	27.06	0.830	28.11	0.857	28.92	0.889	28.83	0.877	**29.48**	**0.937**
Hill	27.69	0.845	28.07	0.822	28.70	0.849	29.23	0.869	29.16	0.855	**29.97**	**0.911**
House	28.22	0.671	30.07	0.813	29.88	0.743	31.56	0.835	32.22	0.847	**32.83**	**0.898**
Lena	28.48	0.850	30.01	0.898	30.12	0.839	31.26	0.918	31.32	0.912	**32.02**	**0.966**

Table 5.3: The average denoising results (average of the whole testing images over 5 times of experiments) when images are corrupted by Gaussian noise with various σ.

Noise Variance	Noisy Image		BF		MBF		LARK		BLS-GSM		KSVD		Proposed	
	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
10	28.11	0.793	31.63	0.923	32.39	0.925	32.41	0.929	33.16	0.951	33.66	0.954	**34.05**	**0.978**
15	24.59	0.678	29.13	0.881	29.29	0.873	30.29	0.892	30.90	0.924	31.48	0.929	**32.71**	**0.981**
20	22.09	0.584	27.46	0.831	27.77	0.828	28.63	0.854	29.34	0.896	29.95	0.904	**30.74**	**0.955**
25	20.15	0.507	26.20	0.782	26.66	0.787	27.36	0.818	28.17	0.869	28.82	0.882	**29.22**	**0.932**
50	14.13	0.278	22.37	0.591	23.62	0.648	23.75	0.672	24.81	0.749	25.03	0.747	**25.11**	**0.753**
75	10.61	0.172	20.03	0.466	22.09	0.567	22.02	0.582	**23.10**	**0.662**	22.89	0.635	22.76	0.642
100	8.11	0.115	18.24	0.378	21.07	0.507	20.99	0.525	**21.98**	**0.596**	21.51	0.559	21.40	0.560

The termination condition for training is that the fitness is smaller than 1×10^{-6} or the maximum number of generations has been reached.

5.5.3 Results

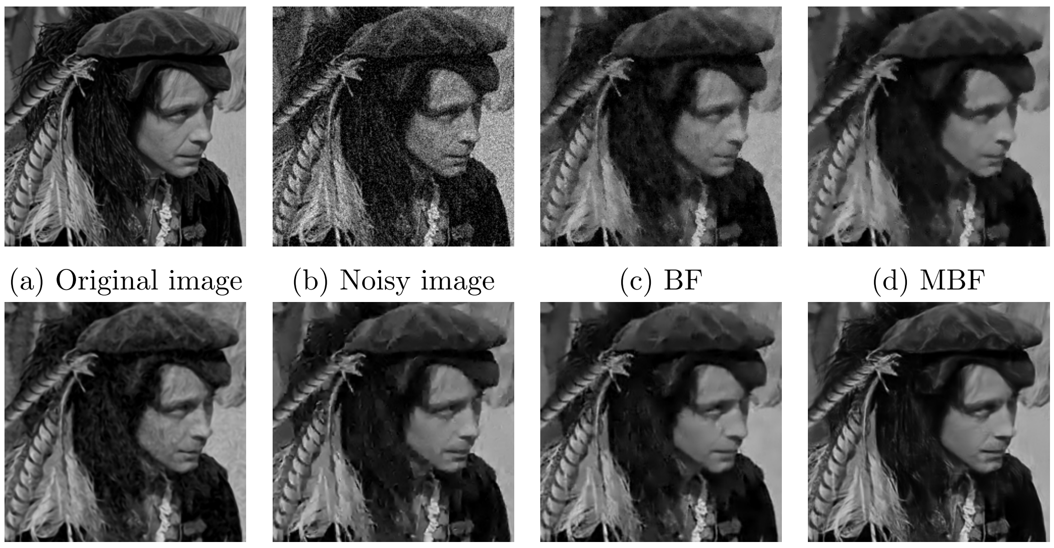
5.5.3.1 Objective performance comparison

Objective performance is evaluated by PSNR and mean structural similarity (MSSIM) (93). The PSNR is the same with eq.(5.9) but calculated on the whole image. The MSSIM is defined as (93):

$$MSSIM(O, E) = \frac{\sum_{k=1}^{B} SSIM(o_k, e_k)}{B} \tag{5.10}$$

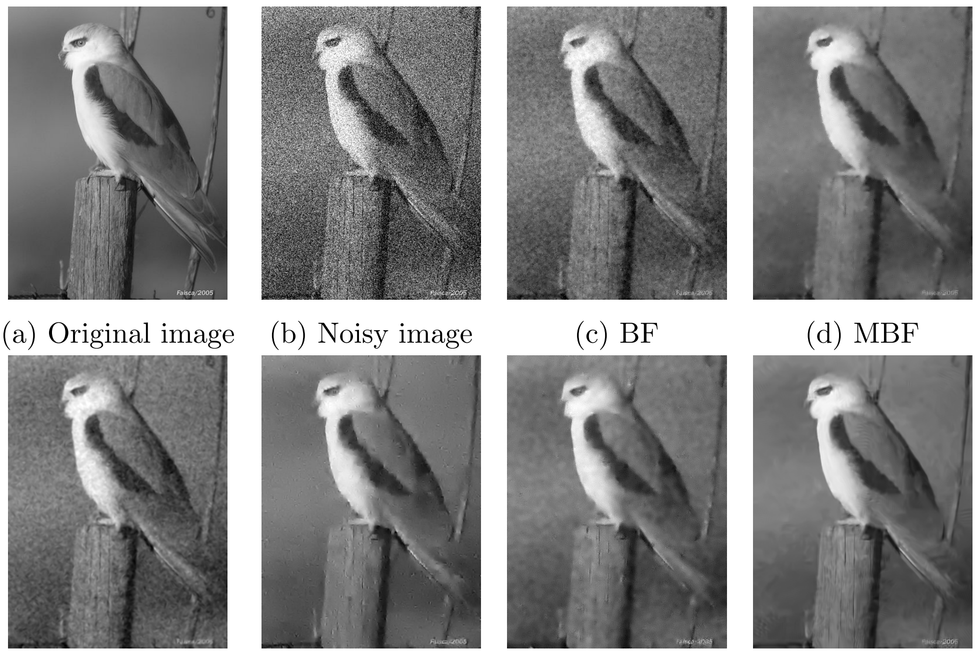
$$SSIM(o_k, e_k) = \frac{(2\mu_{o_k}\mu_{e_k} + C_1)(2\sigma_{o_k e_k} + C_2)}{(\mu_{o_k}^2 + \mu_{e_k}^2 + C_1)(\sigma_{o_k}^2 + \sigma_{e_k}^2 + C_2)} \tag{5.11}$$

where O and E are the original image and the denoised image; o_k and e_k denote k-th patches from original image and denoised image; B is the total number of local windows


(a) Original image (b) Noisy image (c) BF (d) MBF

(e) LARK (f) BLS-GSM (g) K-SVD (h) Proposed

Figure 5.4: The visual result comparison. The noise level is $\sigma = 25$.


(a) Original image (b) Noisy image (c) BF (d) MBF

(e) LARK (f) BLS-GSM (g) K-SVD (h) Proposed

Figure 5.5: The visual result comparison. The noise level is $\sigma = 50$.

in the image; μ_{o_k} and μ_{e_k} are the mean intensity of o_k and e_k; $\sigma_{o_k e_k}$ is the covariance between o_k and e_k; C_1 and C_2 are small constants to stabilize SSIM (93).

Gaussian noise: The proposed method was trained on image patches which are contaminated by additive Gaussian noise with the standard deviation of 10, 15, 20, 25, 50, 75 and 100, respectively. All the parameters of these methods are set as what have been suggested to be the optimal one in the original paper. The quantitative results of our algorithm are compared with the state-of-the-art local learning-based methods: K-SVD, among which K-SVD is superior to others. As shown in Tables 5.2 and 5.3, the proposed ELA filter provides a significant improvement over K-SVD at low or medium noise levels. This is in line with the characteristics of the bilateral filter, in which at those noise levels the filter can manage to avoid blurring edges while smoothing the homogeneous regions. Our GP framework enables adaptive parameter tuning within each local patch, which boosts the performance of the original bilateral filter. On the contrary, K-SVD has the issue of using a global dictionary to generate representations for local patches, which leads to artifacts due to the inaccurate sparse representations. At high noise levels, the ELA filter performs similarly as K-SVD in terms of PSNR, but the MSSIM of our method is higher. When the noise corrupts most of the structures in an image, it brings difficulties for our clustering and the individual bilateral filters selected by GP. Under such circumstances, K-SVD suffers from the same problem and is unable to provide very precise sparse representations.

Meanwhile, some other representative local denoising methods are listed for comparison, including Bilateral Filter (BF), the improved BF (Multiresolution Bilateral Filtering (MBF) (120)) (the best improvement for Bilateral Filter), Kernel regression for image denosing (LARK) (39) (which is an extension of BF by applying more adaptive local filters via learning.), and an overcomplete local image model in the wavelet domain (BLS-GSM) (which is considered to be one of the state-of-the-art local filters). The ELA filter performs much better than most of them, however, when the noise level is high, our method generates worse results than BLS-GSM. The reason is that the candidate filters in the function set all have difficulties in dealing with high noise levels and our GP process hasn't managed to 'create' a breakthrough on this by using many generations' evolution. The implementations by the respective authors are used for all experiments. However, as all the codes are written in MATLAB and run very slowly, we revised the main denoising functions into C and compiled them into Mex functions. This helps the later time complexity comparison in the following section.

Salt-and-pepper noise: The proposed ELA filter has also been compared with state-of-the-art methods for removing salt-and-pepper noise (NAFSM (126)). As shown in Fig. 5.8, our method outperforms these methods when the noise level is not very high (less than 60%). When the noise level is high, our method is slightly worse than NAFSM due to the similar reasons as described for the Gaussian noise. Median filter and AKA (41) have also been employed for comparison within the local filter category. Obviously, our method outperforms these methods significantly.

5.5.3.2 Visual performance comparison

Gaussian noise: As shown in Fig. 5.4 and Fig. 5.5, our proposed ELA method produces better visual results than the other methods at low or medium noise levels. For greater amount of noise, our method generates similar results as K-SVD. This is due to the fact that our function set is mostly composed of bilateral filters, which are good at preserving edges when the noise levels are not high. One can see that the ELA filter provides the best overall visual performance among these methods.

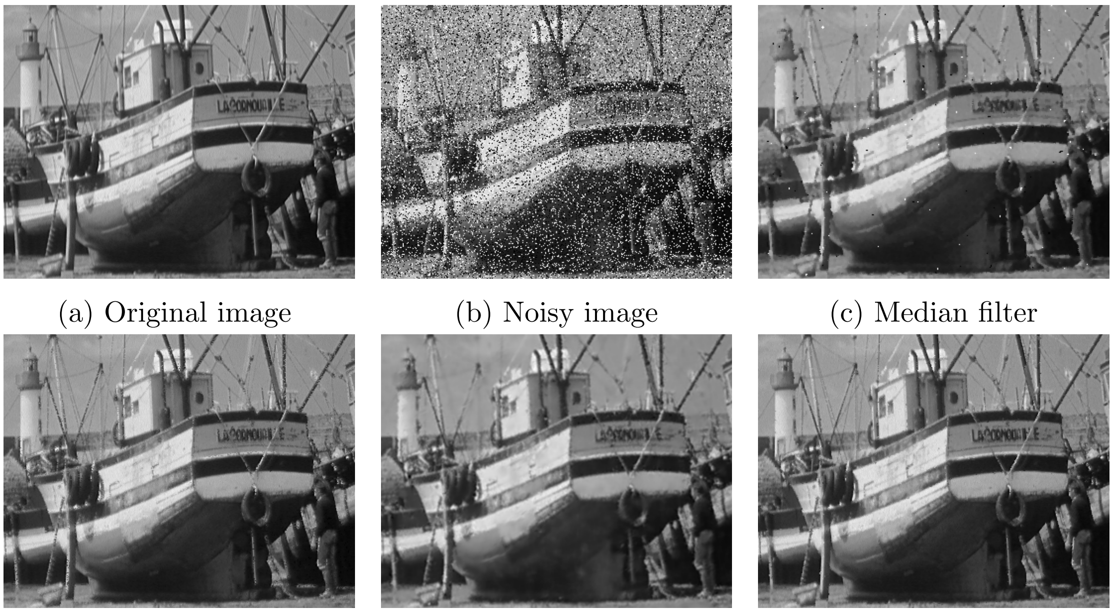
Salt-and-pepper noise: In Fig. 5.6, the edges are well preserved by our proposed ELA filter and NAFSM. However, median filter and AKA fail to maintain edges very well. Between NAFSM and our proposed method, we can observe that around strong edges there are visible noise residues in the NAFSM result but not in the ELA result.

In Fig. 5.7, the amount of noise has been increased. Though our method has less advantage in this case (quantitative results), one can still observe good visual image quality. On the contrary, large amount of noise still remains in the result of the median filter. AKA produces severe artifacts, which is consistent with the quantitative results.

5.5.3.3 Analysis of a sample program

Fig. 5.9 shows a high-performing tree (in LISP syntax) generated by the proposed ELA algorithm. The fitness of the program on the test data set has the average value of 0.0324, which means that GP has managed to find a good filter for this class.

The tree shown in Fig. 5.9 is not like any conventional filtering method. It is a combination of bilateral filters and Gaussian filter, which indicates that GP managed



(a) Original image (b) Noisy image (c) Median filter

(d) NAFSM (e) AKA (f) Proposed

Figure 5.6: Denoising results of image "boat" corrupted with 20% salt-and-pepper noise. For the median filter, the iteration times are chosen for this noise level to achieve its best performance.

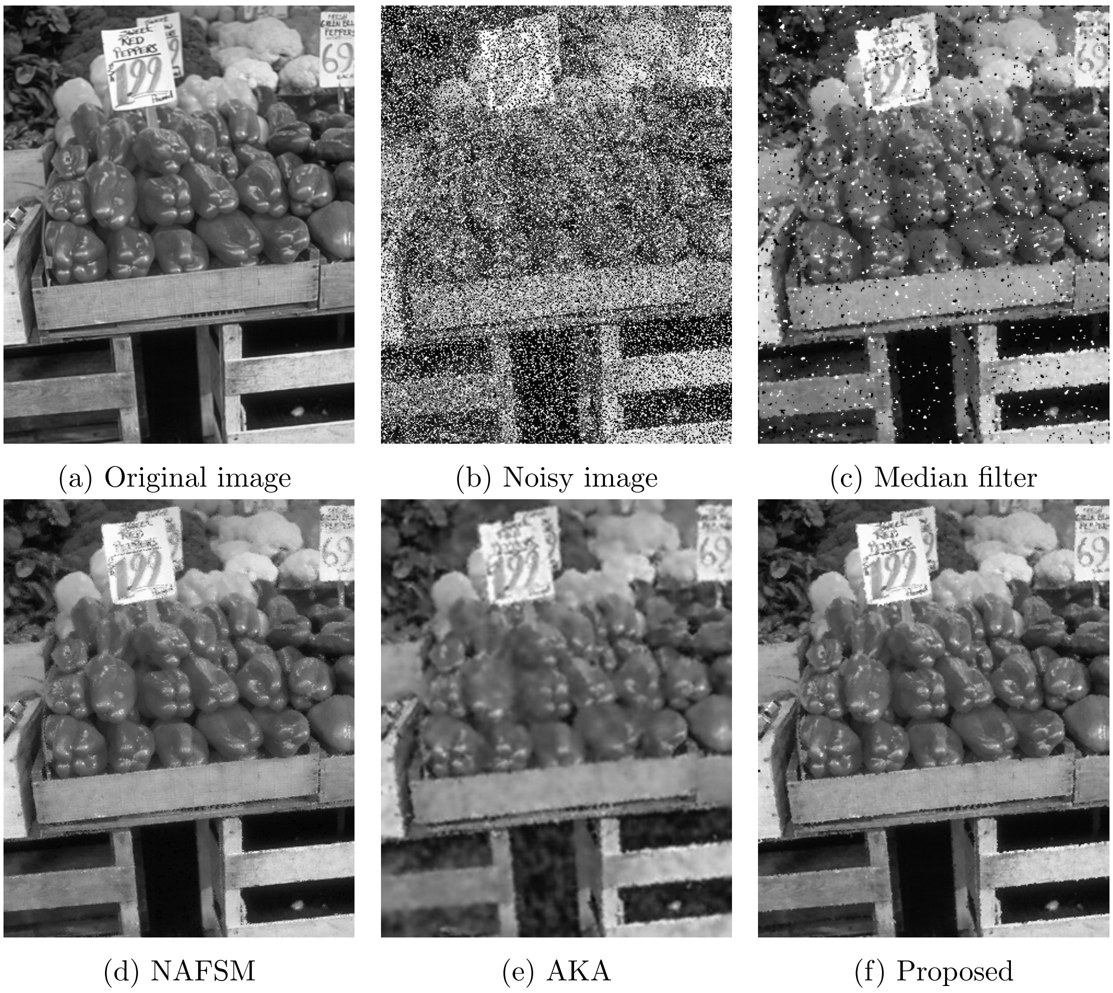


Figure 5.7: Denoising results of image "vegetables" corrupted with 40% salt-and-pepper noise. For the median filter, the iteration times are chosen for this noise level to achieve its best performance.

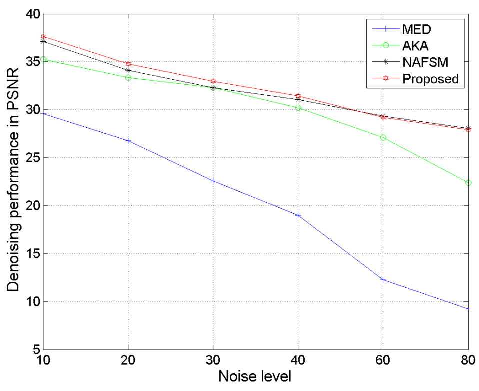


Figure 5.8: The denoising performance for salt and pepper noise. For the median filter, the iteration times are chosen for different noise levels to achieve its best performance.

to consider both edge preserving and smoothing the homogeneous regions for various image patch content.

```
Add(AbsSub(Bil2(Sub(Bil4(X1),0.243)),Bil6(X1)),
Bil2(Sub(Bil4(X1),Add(AbsSub(Bil3(Sub(Gau(X1),-0.172)),
Bil2(X1)),Bil1(Sub(X1,0.699))))))
```

Figure 5.9: The trained GP filter when the noise model is Gaussian and the noise level is 20.

5.5.4 Computational efficiency

The GP process is computationally expensive in the off-line training. For each class, there are 50 generations and a population of 500 individuals within each generation. The typical training time for a single cluster with 2000 elements is 8 hours. However, the online testing stage is very efficient compared to the state-of-the-art, which only takes 0.27 sec for an image of the size 256×256. The MBF takes about 6.21 sec. The BLS-GSM model with parameter setting ”1” requires 51.87 sec, and K-SVD (10 iterations) takes 657.14 sec.

5.6 Conclusions

We have presented an image denoising method using genetic programming. By exploiting the randomness of GP, the generated optimal filter for each class is effectively more adaptive. The experimental results demonstrate that the proposed method achieves comparable and better results for removing both Gaussian noise and salt-and-pepper noise compared to the state-of-the-art local methods. In future work, our proposed method could be extended to other image enhancement tasks (for instance, coding artifact removal) by changing the patch clustering methods, the function set, and the fitness of GP.

In our extra experiments, we tried to apply the sparse coding algorithm (chapter 4) to the problem of coding artifact removal. However, the dictionary trained by sparse coding is not adaptive to the special textures in the images corrupted by coding artifacts. Usually, the special textures are disconnected edges which cannot be easily reconstructed from trained dictionary simply because the sparse coding objective function has this limitation that the residue between the target and reconstructed images should have Gaussian distribution or distributions similar to Gaussian. Since the distribution of coding artifacts is very irregular, the sparse coding model fails. In this chapter, we exploited GP as a tool for developing solutions for irregular problems. For instance, we tried to use GP for both the Gaussian noise and Salt-and-pepper noise reduction. In future, we can consider using different function set which contains suitable functions for coding artifact removal, which might solve the problem of irregular distortion caused by the image codec.

6

Image restoration using Deep Learning

6.1 Image Blur identification using Deep Neural Networks

6.1.1 Abstract

Image blur kernel classification and parameter estimation are critical for blind image deblurring. In this chapter, our work focuses on two parts: the identification of the blur type and then the parameter classification. Current dominant approaches use handcrafted blur features that are optimized for a certain type of blur, which is not applicable in real blind deconvolution application when the Point Spread Function (PSF) of the blur is unknown. In this chapter, a Two-stage system using Deep Belief Networks (TDBN) is proposed to first classify the blur type and then identify its parameters. To the best of our knowledge, this is the first time that Deep Belief Network (DBN) has been applied to the problem of blur analysis. The reason we choose DBN is that it can learn a good representation from the input features by the double-direction inferences. In the blur type classification, our method attempts to identify the blur type from mixed input of various blurs with different parameters, rather than blur estimation based on the assumption of a single blur type in current methodology. To this aim, a semi-supervised DBN is trained to project the input samples into a discriminate feature space, and then classify those features. Moreover, in the parameter identification, the proposed edge detection on logarithm spectrum helps DBN to identify the blur parameters with very high accuracy. In this proposed chapter, the training process is entirely

offline, which is different from the previous chapter 4. Experiments demonstrate the effectiveness of the proposed methods with better results compared to the state-of-the-art on the Berkeley segmentation dataset and the Pascal VOC 2007 dataset.

This chapter is based on the following work:

R. Yan, L. Shao, "Image blur classification and parameter identification using two-stage deep belief networks", in Proc. *British Machine Vision Conference*, Bristol, UK, 2013.

R. Yan, L. Shao, "Blur kernel classification and estimation from a single image", submitted to *IEEE Transactions on Image Processing*, 2013.

6.1.2 Introduction

Image blur is a major source of image degradation, although sometimes it is required for artistic purposes. Various reasons can cause image blur, such as the atmospheric turbulence (Gaussian blur), camera relative motion during exposure (motion blur), and lens aberrations (out-of-focus blur) (127).

The restoration of blurred photographs, image deblurring, is the process of inferring latent sharp images with inadequate information of the degradation model. It can be categorized into *blind* and *non-blind*. Non-blind deblurring requires the prior knowledge of the blur kernel and its parameters, while in blind deblurring we assume that the blurring operator is unknown. In most situations of practical interest the Point Spread Function (PSF) is not acquired, so the application range of non-blind deblurring is much narrower than the blind deblurring (128). In blind image deblurring, there are two main classes: *multi-image* (32, 129, 130) and *single-image* deblurring. In the real application, a single blurred image is usually the case we need to deal with. For instance, Baysian network has been used for single-channel blind deconvolution in (131). Wavelet transform is applied to tune the parameter for Gaussian blur in (132). Restoring images degraded by motion blur is discussed in (133). Similarly, one popular approach is the application of radon transform, which can estimate the blur kernel by analyzing the edges in the image (134). Other methods have also been tried for motion blur, such as cepstral method, and steerable filters (135).

While most previous work focuses on image deblurring, not as much research has been done on *blur classification*, which is more practical because the type of blurs is usually unknown in photographs. Based on the descriptor of blurs, there are a few blur classification methods without image deblurring. One of the state-of-the-art method is based on a Bayes Classifier using blur features, for instance, local autocorrelation congruency (5). Another similar method, based on the alpha channel feature, has been proposed by Su *et al.* (6), which has different circularity of the blur extension. Though both of them managed to detect local blurs in the realistic image, their methods are based on handcrafted features.

Although previous blur classification methods can perform well with handcrafted features, their performance is still limited due to the diversity of natural images. Recently, many researchers have moved their attention from the heuristic prior to the learned deep architecture. The deep hierarchical neural network has similar structure as human visual cortex, which has been applied in many vision tasks, such as object recognition, image classification, and even image analysis. In Jain *et al.* 's denoising work (136), they have shown the potential of using Convolutional Neural Network (CNN) for denoising images corrupted by Gaussian noise. In such an architecture, the learned weights and biases in the deep convolutional neural network are obtained through the training on sufficient amount of natural images. For testing stage, these parameters in the network act like "prior" information for the degraded images, which end up with better results compared to the top local denoising approaches. Another example is the blur extent metric developed by the multifeature classifier based on Neural Networks (NN) (137). It has proved that the combined learned feature works better than individual handcrafted feature mostly.

Inspired by the practical blur type classification in (5, 6) and the merits of learned descriptors in (136, 137), we intend to design another patch-based blur type classification and parameters identification method to better solve the realistic blur analysis problem. Deep Belief Network (DBN) is chosen for accomplishing the feature extraction and final classification in this system. A two-stage framework is proposed: first, for the input image patches with different blurs, the DBN is used for identifying the blur type; second, different samples with the same blur type will be sent to the corresponding DBN blocks for further parameter estimation. The DBN is trained in a semi-supervised way: the unsupervised training of the DBN is done by a greedy layer-wise pre-training before the supervised backpropagation for the fine-tuning. The unsupervised process helps the feature learning, and the backpropagation helps to construct the discriminative information.

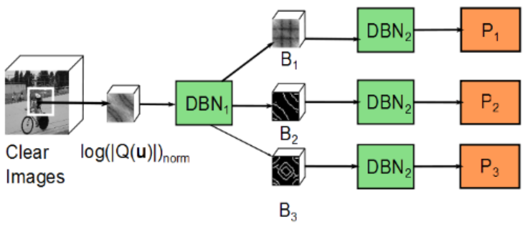


Figure 6.1: The TDBN architecture: DBN_1 is the first stage for blur type classification, which has 3 output labels. DBN_2 is the blur PSF parameter identification, which has different output labels for each blur type. P_1, P_2, and P_3 are the estimated parameter labels, which can be seen in Sec. 6.1.7.3.

In a word, our contributions are threefold:

- To the best of our knowledge, this is the first time that deep belief network has been applied to the problem of blur analysis.
- A discriminative feature, derived from edge extraction on Fourier Transform coefficients, has been proposed to preprocess blurred images before they are fed into the DBN.
- A two-stage framework is proposed to estimate the blur type and parameters for any given image degraded by spatially invariant blur of an unknown type.

6.1.3 Problem Formulation

The image blurring can be modeled as the following degradation process from the high exposed image to the observed image (138):

$$g(\mathbf{x}) = q(\mathbf{x}) * f(\mathbf{x}) + n(\mathbf{x}) \tag{6.1}$$

where $\mathbf{x} = \{x_1, x_2\}$ denotes the coordinates of an image pixel, g represents the blurred image, f is the intensity of the original high quality image, q denotes the PSF of a certain blur type, $*$ indicates the convolution, and n is the additive noise.

In blind image deconvolution, it is very difficult to recover the PSF from a single blurred image due to the loss of information during blurring (139). Our goal is to classify the blurred patches into their corresponding degradation functions and parameters. Several blurring functions are considered in this chapter.

In many applications, such as satellite imaging, Gaussian blur can be used to model the PSF of the atmospheric turbulence:

$$q(\mathbf{x},\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{x_1^2+x_2^2}{2\sigma^2}), \quad \mathbf{x} \in R \tag{6.2}$$

where σ is the blur radius to be estimated, and R is the region of support. R is usually set as $[-3\sigma, 3\sigma]$, because it contains 99.7% of the energy in a Gaussian function (140).

Another blur is caused by linear motion of the camera, which is called motion blur (141):

$$q(\mathbf{x}) = \begin{cases} \frac{1}{M}, & (x_1, x_2)\begin{pmatrix} sin(\omega) \\ cos(\omega) \end{pmatrix} = 0 \quad and \quad x_1^2 + x_2^2 \leq M^2/4 \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

where M describes the length of motion in pixels and ω is the motion angle according to the horizontal x axis. These two parameters are what we need to estimate in our system.

The third blur is the out-of-focus blur, which can be modeled as a cylinder function:

$$q(\mathbf{x}) = \begin{cases} \frac{1}{\pi R^2}, & \sqrt{x_1^2 + x_2^2} \leq R \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

where R is the blur radius.

In the blur classification method of (5), a motion blur descriptor, local autocorrelation congruency, is used as a feature for the Bayes classifier to discriminate motion blur from defocus blur because the descriptor is strongly related to the shape and value of the PSF. Later, Su *et al.* (6) have presented better handcrafted features for blur classification, which gives better results without any training. Though both methods generate good results on identifying motion blur and out-of-focus blur, the features they used are both limited to a single or several blur kernels. In this chapter, we attempt to find a general feature extractor for common blur kernels with various parameters, which is closer to realistic application scenarios. Therefore, enlightened by the previous success of applying deep belief networks to discriminative learning (26), we consider to use the DBN as our feature extractor.

When designing the DBN, it is natural to use observed blurred patches as training and testing samples. However, their characteristics are not as obvious as their frequency coefficients (142). Hence, the logarithmic power spectra are adopted as input features for the DBN, since the PSF in the frequency domain manifests different characteristics for different blur kernels. Bengio *et al.* (143) have pointed out that the scaling continuous-valued input to $(0, 1)$ worked well for pixel gray levels, but it is not necessarily appropriate for other kinds of input data. From Eq. 6.1 one can see that the noise might interfere the inference in the DBN (143), so preprocessing steps are necessary for preparing our training samples. In this chapter, we use an edge detector to obtain binary input values for the DBN training, which has been proved to benefit the blur analysis task.

We propose a two-stage classification system to both classify the blur kernel and identify the blur parameters. These two stages have a similar network architecture but different input layers. The first stage is an initial classification of the blur type, and the second stage is to further identify the blur parameters within samples with the same label from the results of the first stage. Since the variation between blur parameters of the same blur type is not as great as that between different blur types, more discriminative features have been designed for the second stage, which yields much better results than combining the two stages into one in our experiments.

6.1.4 Blur Features

If we apply the Fourier Transform (FT) to both sides of Eq. (6.1), we can obtain:

$$G(\mathbf{u}) = Q(\mathbf{u})F(\mathbf{u}) + N(\mathbf{u}) \tag{6.5}$$

where $\mathbf{u} = \{u_1, u_2\}$. For the out-of-focus blur, $Q(\mathbf{u}) = \frac{J_1(\pi R r)}{\pi R r}, \quad r = \sqrt{u_1^2 + u_2^2}$.

J_1 is the first-order Bessel function of the first kind, R is the radius of the amplitude of the periodic function (144). [1]

For the motion blur, the FT of the PSF is a *sinc* function: $Q(\mathbf{u}) = \frac{sin(\pi M \omega)}{\pi M \omega}$, $\omega = \pm\frac{1}{M}, \pm\frac{2}{M}, ...$

[1]http://www.clear.rice.edu/elec431/projects95/lords/elec431.html

In order to know the PSF $Q(\mathbf{u})$, we attempt to identify type and parameters of Q from the observation image $G(\mathbf{u})$. Therefore, the normalized logarithm of G can be used in our implementation:

$$\log(|G(\mathbf{u})|)_{norm} = \frac{\log(|G(\mathbf{u})|) - \log(|G_{min}|)}{\log(|G_{max}|) - \log(|G_{min}|)} \tag{6.6}$$

where G represents $G(\mathbf{u})$, $G_{max} = max_{\mathbf{u}}(G(\mathbf{u}))$, and $Gmin = min_{\mathbf{u}}(G(\mathbf{u}))$.

As shown in Fig. 6.2, the patterns in these images ($\log(|G(\mathbf{u})|)_{norm}$) can represent the motion blur or the defocus blur intuitively. Hence, no extra preprocessing needs to be done for the blur type classification. However, defocus blurs with different radii are easy to be confused, which also has been proved in our experiments. Therefore, for blur parameter identification, an edge detection step is proposed here.

If a classic edge detector is applied directly, redundant edges would interfere with the pattern we need for the DBN learning. Many improved edge detectors have been explored to solve this issue, however, most of them do not apply to the logarithmic power spectra data, which cause even worse performance (145, 146). For instance, Bao *et al.* (145) proposed to improve the Canny detector by the scale multiplication, which indeed enhances the localization of the Canny detector. However, this method does not generate good edges on our input data for the DBN.

We solve this issue by applying the Canny detector first, and then using a heuristic method to refine the detected edges. Due to the fact that the useful edges are isolated near zero-crossings, we need to refine the detection results from the logarithmic power spectrum. The Canny edge detector is applied to form an initial edge map. Then, we design several steps to select the most useful edges: 1) For both of the blur types, we select isolated edges. Assuming the isolated region has the radius d, those edges, in the orthogonal direction of the current edge within radius d, will be discarded (134). 2) For the motion blur, we abandon short and very curvy edges. We consider the orientations $\theta = [0, \pi]$ of the candidate edges within radius d are considered. The criterion proposed by Watson (147) is utilized for estimating their alignment (134).

For the Gaussian blur, the Fourier transform of the PSF is still a Gaussian function, and there is no significant pattern change in the frequency domain. From Eq. (6.2), we can see that the Gaussian kernel serves as a low pass filter. When the sigma of this filter is larger, more “high frequency information” will pass through. However, from our observation, when the σ is larger than 2, the pattern on the logarithmic spectrum image barely changes. In the experiment section, we show that edge detection is not suitable in this case.

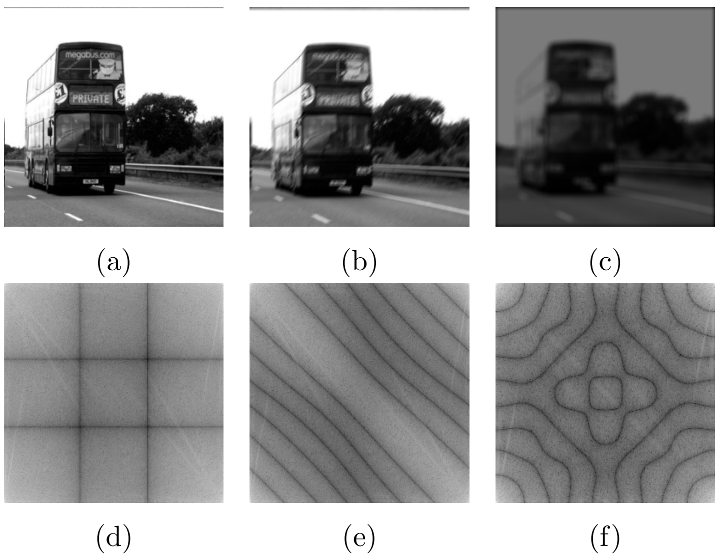


Figure 6.2: The blur images and their logarithmic spectra. (a) Image with Gaussian blur. (b) Image with motion blur. (c) Image with out-of-focus blur. (d) Logarithmic spectrum of Gaussian blur ($\sigma = 2$). (e) Logarithmic spectrum of motion blur ($M = 9, \omega = 45$). (f) Logarithmic spectrum of out-of-focus blur($R = 30$).

6.1.5 The Training Process of Deep Belief Networks

Deep belief nets are used as a generative model for feature learning in a lot of previous work (26). In this chapter, we first construct the DBN by unsupervised greedy layer-wise training to extract features in the form of hidden layers and then apply a fine-tuning for discriminative weights in a supervised way.

The training process of an individual DBN is as follows:

1. The input layer is trained in the first Restricted Boltzmann Machine (RBM) as the visible layer. Then, a representation of the input blurred sample is obtained for further hidden layers. This representation is chosen to be the mean activations in our experiments as $p(h^{(k+1)} = 1|h^{(k)}), k = 0, 1, ...P$, where P is the number of all the hidden layers.

2. The next layer is trained as an RBM by greedy layer-wise information reconstruction. The training process of RBM is to update weights between two adjacent layers and the biases of each layer. In our scheme, Contrastive Divergence (CD) (148) is applied.

3. Repeat the first and second steps until the parameters in all the layers (visible and all hidden layers) are learned.

4. In the supervised learning part, the labels are used for training the DBN to have discriminant ability using backpropagation. Then, the goal for the optimization process is to minimize the backpropagation error derivatives: $\phi^* = \arg\min_{\phi}[-\sum_p \mathbf{y}_p \log \hat{\mathbf{y}}_p]$, where $\mathbf{y}_p$ and $\hat{\mathbf{y}}_p$ are the estimated label and the correct label. The conjugate gradient descent is used for this optimization.

6.1.6 Forming the TDBN

The TDBN is formed by two-stage DBN learning (Fig. 6.1). First, the identification of blur patterns is carried out in the first stage by using the logarithmic spectra of the input blurred patches. The output of this stage is 3 labels: the Gaussian blur, the motion blur and the defocus blur. With the label information, the classified blur vectors will be used in the second stage for blur parameter identification. At this stage, motion blur and defocus blur will be further preprocessed by the edge detector (Sec. 6.1.4) before the training but Gaussian blur vectors remain the same. The output of this stage is various labels for individual DBNs as shown in Sec. 6.1.7.3.

6.1.7 Experiments

6.1.7.1 Experimental setup

Training datasets: *The Oxford image classification dataset* [1], and *the Caltech 101 dataset* are chosen to be our training sets. We randomly selected 4000 images from each of them.

The size of the training samples ranges from 32×32 to 128×128 pixels, which are cropped from the original image. By empirical evaluation, the best results occur when the patch size is 32×32. The size of the training set is 12000 (randomly selected from

[1] http://www.robots.ox.ac.uk/ vgg/share/practical-image-classification.htm

those cropped images). In those 12000 training samples, 4000 of them are degraded by Gaussian PSF, 4000 of them are degraded by the PSF of motion blur, and the rest are degraded by the defocus PSF.

Testing datasets: *Berkeley segmentation dataset* (200 images) has been used for our testing stage, which has been applied to the denoising algorithms (123, 149). *Pascal VOC 2007*: 500 images are randomly selected from this dataset (150).

2000 testing samples are chosen from each of them according to the same procedure as the training set. The numbers of the three types of blurred patches are random in the testing set.

Blur features: The Canny detector is applied to the logarithmic power spectrum of image patches with automatic low and high thresholds. Afterwards, the isolated edges are selected with the radius of 3 pixels according to the suggestions from (134).

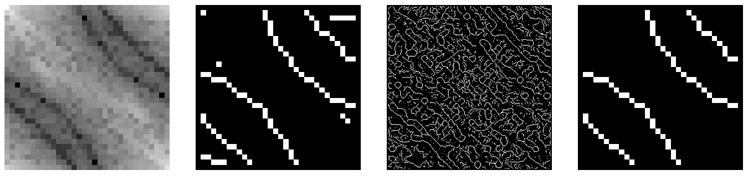


Figure 6.3: Comparison of the three edge detection methods applied to a training sample. From left to right: (1) the logarithmic power spectrum of a patch; (2) the edge detected by Canny detector (automatic thresholds); (3) the edge detected by the improved Canny detector using scale multiplication; (4) the edge detected by our method

DBN Training: For parameters of the DBN learning process, the basic learning rate and momentum in the model are set according to the previous work (143). In the unsupervised greedy learning stage, the number of epochs is fixed at 50 and the learning rate is 0.1. The initial momentum is 0.5, and it changes to 0.9 after five epochs. Our supervised fine-tuning process always converges earlier than epoch 30.

6.1.7.2 Image blur type classification

In our implementation, the input visible layer has 1024 nodes, and the output layer has 3 labels (Gaussian kernel, motion kernel, and defocus kernel). Therefore the whole architecture is: 1024 $\longrightarrow$ 500 $\longrightarrow$ 30 $\longrightarrow$ 10 $\longrightarrow$ 3. These node numbers in each hidden layer are selected empirically.

Method	Features	CR1	CR2
Liu *et al.* 's method (5)		79.3%	80.5%
Su *et al.* 's method (6)	Handcrafted	81.6%	83.1%
SVM on our features (153)		78.2%	80.8%
NN (151)		92.5%	91.7%
CNN (152)	Learned	95.3%	96.8%
Proposed		**99.7**%	**98.2**%

Table 6.1: Comparison of obtained average results on the two testing datasets with the state-of-the-art. CR1 is the Berkeley dataset, and CR2 is the Pascal dataset.

On the one hand, we compare our method with the previous blur type classification methods based on handcrafted features: (5) and (6). Their original frameworks contain a blur detection stage, and the blur type classification is applied afterwards. However, in our algorithm, the image blurs are simulated by convolving the high quality patches with various PSFs. In our comparison, (5) has been trained and tested with the same datasets we used, while (6) has been tested with the same testing set we used.

On the other hand, NN (151), CNN (152) and Support Vector Machine (SVM) have been chosen for the classifier comparison. The same blur feature vectors are used for NN and CNN. The SVM-based classifier was implemented following the usual technique: several binary SVM classifiers are combined to the multi-classifier (153).

The classification rate is used for evaluating the performance:

$$CR = 100\frac{N_c}{N_a}(\%) \tag{6.7}$$

where N_c is the number of the correct classified samples, and N_a is the number of the total samples.

We can observe from Table 6.1 that algorithms based on learned features perform better than those based on handcrafted features, which suggests that learning-based feature extractor is less restricted to the type of the blur we consider. Meanwhile, our method performs best among all the algorithms using automatically learned features. Even though SVM is the most commonly used classifier for many computer vision tasks, it requires the handcrafted features to be distinctive enough for good classification results. While in our experiments for comparison, the same handcrafted features are

Method	CR11	CR12	CR13	CR21	CR22	CR23
SVM (153)	96.5%	97.2%	96.9%	95.1%	95.7%	94.9%
NN (151)	90.1%	92.6%	92.2%	90.9%	91.5%	90.6%
CNN (152)	97.9%	98.9%	98.5%	97.3%	98.1%	98.2%
DBN	97.8%	98.1%	97.9%	97.7%	97.8%	97.5%
TDBN1	**99.5**%	98.8%	98.4%	**99.3**%	98.5%	98.2%
TDBN2	99.2%	**99.9**%	**99.4**%	99.1%	**99.7**%	**99.2**%

Table 6.2: Comparison of obtained results on the two testing datasets with the state-of-the-art. In CRxx the first x refers to the dataset type (1 for Berkeley and 2 for Pascal) and the second x refers to the blur type (the Gaussian blur, the motion blur, and the defocus blur). DBN is the case that the mixed blur patches are classified by a single DBN. TDBN1 is the case when we use the logarithm spectrum for stage 1 and stage 2. TDBN2 is the case when we use the logarithmic spectrum for stage 1 and edge detection for stage 2.

used as input for both the categories. The learning based techniques all have this function of updating features and classifiers at the same time, which outperforms SVM.

6.1.7.3 Blur kernel parameter identification

In this experiment, the parameters of the blur kernels are identified. For different blur kernels, different parameters are estimated as explained in Sec. 6.1.3. The parameters are set as: 1) Gaussian blur has 8 labels: $\sigma = \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4\}$; 2) Motion blur has 8 labels: $M = \{3, 9\}$ $\omega = \{0, 45, 90, 135\}$; 3) Out-of-focus blur has 8 labels: $R = \{2, 5, 8, 11, 14, 17, 20, 23\}$. The architecture in each DBN is the same except for the output layer: $1024 \longrightarrow 576 \longrightarrow 36 \longrightarrow 25 \longrightarrow$ number of labels for a certain blur type.

Our method is compared to the NN, CNN, and SVM with the same input layer of the blur features. As shown in the following Table 6.2, our method achieves the best results among all, especially for the motion and defocus blur due to the obvious patterns they have in their logarithmic power spectra. Besides, for the Gaussian blur, we can observe that the edge detector has not benefited them, which is consistent with our previous analysis. Moreover, our proposed two-stage strategy works better than a single DBN as shown in Table 6.2.

6.1.8 Conclusion

In this section, a two-stage deep belief network has been proposed for the blur type classification and parameter identification. Our training samples are generated by patches from abundant datasets, after the Fourier transform and our designed edge detection. In the training stage, deep belief networks have been applied in a semi-supervised way. That is, the whole network is trained in an unsupervised manner and afterwards the backpropagation fine-tunes the weights. In this way, a discriminative classifier has been trained. The experimental results have demonstrated the superiority of our TDBN compared to the state-of-the-art methods.

6.2 Image Noise Level Identification using DBN

6.2.1 Introduction

In previous chapter, most of the image denoising methods we proposed are based on the fact that the noise level has already been acquired. However, in real applications, for instance, object recognition, image segmentation, and blind image denoising, noise levels are usually unknown for most circumstances while it does largely affect the performance of other steps. Under the assumption of Gaussian additive model for the input noisy image, it is a common difficulty for us to estimate noise level because it is not easy to tell whether the high frequency information is noise or detailed textures.

The current noise level estimation methods (154, 155, 156, 157, 158, 159) can be divided into: filter-based, and patch-based methods. In the filter-based method (154, 159), the noisy image will be filtered first and then the method noise (10) will be calculated. Ideally, the method noise should be random Gaussian noise if the filtering process performs well. In this way, the noise level can be estimated using the method noise. However, these algorithms have their limitation that they require very advanced blind denoising filter and the image itself should not have too many detailed textures.

The second type of noise level estimation methods are patch-based (155, 157, 160). According to our previous analysis of the noise level estimation problem, we could select patches which have homogeneous regions for noise estimation simply because in these patches there are less interferences from high frequency details. Liu *et al.* (160) have proposed to select low-rank patches based on the gradients and other statistics. After the selection, principal component analysis is used for noise estimation on the selected patches. This method works well on most images but it can potentially fail for images with few low-rank patches. The patch selection performance is not stable because of the input noise level. When the noise level is too high, many statistics features do not work very well as expected.

In this section, we are aiming at designing a noise level estimation method which is not restricted to image contents. The proposed method is filter-based. However, the filter we choose has been proved to perform well at most noise levels ranged $[0, 100]$. It is the trained Convolutional Neural Network that can do blind denoising. After the filtering process, we use deep belief network to classify the noise level according to the edge in the denoised image. The DBN acts as both edge detector and classifier for the input. For images corrupted by severe noise, the edge is not well preserved. However, for mildly corrupted noisy image, the edge can still be sharp and continuous. DBN can classify the patches according to this edge information.

6.2.2 Related Work

The noise model of the image patch can be modeled as the following equation:

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \boldsymbol{n}_i \tag{6.8}$$

where $i, i = 1, ..., N$ is the index of the pixel, N is the number of the pixels in an image. In most applications, the Gaussian noise model is considered.

In Jain *et al.* 's work (136), the Convolutional Neural Network (CNN) has been used for training the blind denoising filter. Assume that m hidden layers are used for the CNN training, the relationship between input layer and the hidden layer afterwards can be described as:

$$I_{m,a} = f(\sum_b w_{m,ab} * I_{m-1,b} - \beta_{m,a}) \tag{6.9}$$

where $I_{m-1,b}$ are feature maps serving as the input for $I_{m,a}$, and $*$ is the convolution. The function f is: $f(x) = 1/(1 + e^{-x})$ and $\beta_{m,a}$ is the bias parameter. In this image denoising application, the output of the mapping function should preferably have the range [0, 1], therefore, the sigmoid function is chosen (136).

In the blind denoising process, for the target image $\boldsymbol{x}_i$, input image $\boldsymbol{y}_i$ has been corrupted by random noise with standard deviation ranging from [0, 100]. The training of the CNN is to optimize the parameters for the network to minimize the reconstruction error:

$$\min_{\phi} \sum_{i} (\boldsymbol{x}_i - F_{\phi}(\boldsymbol{y}_i))^2 \tag{6.10}$$

The experimental results in this chapter have shown that compared to density estimation based methods CNN can manage to learn a more representative model. Therefore, this CNN is proposed to act as the pre-filter for our first step in the framework.

6.2.3 Noise Level Identification based on Deep Belief Network

In this section, a noise level estimation method is proposed based on the Deep Belief Network. For an input noisy image, the CNN blind denoising method should be applied first and then the denoised image will be sent to a simple edge detector, the result of which can be used as the input of the DBN. The DBN used in this section is similar to the one in Sec. 6.1. It has the first few layers as the hidden structures and the last layer as the classifier.

6.2.3.1 Noise features

There are various noise features used for noise level estimation. Usually, statistical values can be a good indicator for noise levels. For instance, in Zoran *et al.* 's work (161), the change in kurtosis values can be used for evaluating the actual noise level.

In our proposed method, the denoised images are used for the input of the deep belief network. The input noisy patches are first denoised by the blind CNN denoising filter. A six-layer CNN is trained for the blind denoising process. The input patches are corrupted by random sigma value from the range $[0, 100]$.

6.2.4 Experimental Validation

6.2.4.1 Experimental setup

In this experiments, the training datasets, testing datasets, and the parameters are set according to the previous section. The comparison we want to make in this section are twofold: 1) using different features as input for deep belief network; 2) using the proposed noise feature as input for different classifiers. In our experiments, three noise levels $\sigma = 10, 50, 100$ are added to each random patch we extract from the datasets, which could prove that deep belief network is suitable for classifying the denoised images for noise level estimation.

6.2.4.2 The spatial pooling for comparison

In our experiments, the spatial pooling method (162) is applied for optimizing the parameters in the pooling layer and classifier at the same time. Different from the traditional pooling (e.g. max pooling, sum pooling), the spatial pooling in (162) has used a parameterized version of the pooling operator:

$$\Theta_w(U) := \rho_{j=1}^{M}(w_j \circ u_j) \tag{6.11}$$

where $w_j \circ u_j$ is the element-wise multiplication, and ρ is a pooling function. ρ can be either max or sum. This pooling weight can be used for the Spatial Pyramid Matching (SPM) architectures in that the pooling regions can be shaped and the weights for the areas can be learned as well as the classifier. Assume that the input code words are u_j, the pooling units can be denoted by a_l in the following equation (162):

$$a_l := \rho_{j=1}^{M}(w_j^l \circ u_j) = \Theta_{w^l}(U) \tag{6.12}$$

The regression process from the input units to the output layer can be expressed as:

$$J(\Theta) := -\frac{1}{D}\sum_{i=1}^{D}\sum_{j=1}^{C} 1y^{(i)} = j \log p(y^{(i)} = j|a^{(i)}; \Theta) \tag{6.13}$$

where D denotes the number of all images, C is the number of all classes, $y^{(i)}$ is a label assigned to the i-th input image, and $a^{(i)}$ are responses from the 'stacked' pooling units for the i-th image.

6.2.4.3 Experimental Results

In our experiments, the input for the pooling SPM is the noise feature proposed in this section. At the same time, a support vector machine (SVM) using lib-svm with a

Methods	CR1	CR2
SVM (163)	81.22%	82.57%
NN (151)	93.15%	93.84%
SPM+Pooling (162)	98.89%	98.93%
Ours	99.99%	99.95%

Table 6.3: Comparison of obtained classification rate on the two testing datasets with the learning-based classifiers. CR1 is the Berkeley dataset, and CR2 is the Pascal dataset.

radial basis function kernel (163) on the same input noise features are also tested. The results of the comparison are shown in Table 6.3.

We can observe from Table 6.3 that both the methods involving optimizing parameters and classifiers at the same time could end up with better results compared to the traditional classifier. Similar to the discussion from the previous section Sec. 5.4, SVM is very limited in this experiment because it can only provide good classification results while deep learning based methods can capture the image structure information better. Among the learning based methods, our proposed method performs the best simply because this DBN was originally proposed for binary images and in this scenario the input of our problem are edges extracted from the input images, which is binary too. From this perspective, we consider using deep learning rather than a regressor (e.g. Neural Network) is more beneficial to the image analysis problems considering that the image pixels within a patch is not independent from each other.

6.2.5 Conclusion

In this section, we follow the idea from Sec. 6.1 to design a DBN based classifier for noise level identification, which is potentially useful for blind image denoising. Similar to the previous filter-based method, in this algorithm, a blind denoising filter is applied to the input noisy image first, and then an edge detector is exploited for useful structure information. Finally, this extracted edge information will be sent into the deep belief network for further classification. Keeping the same structure as what is explained in Sec. 6.1, this DBN works well on the denoised image features, which has been shown in our experimental results. This section has proved that DBN consistently works better than simple neural network on image analysis problem. The reason for that is deep belief network is the general model which is aiming at finding the actual pattern or structure in the input rather than ignoring this information but trying hard to 'catch up the target'. Therefore, we have reasons to believe that DBN can work well on other image analysis problems too, for instance, image blur analysis. In the next section, we are going to have extended experiments on the problem of blur estimation rather than just blur identification.

6.3 Blur Classification and Parameter Estimation from a Single Image

6.3.1 Abstract

In this section, a learning-based method using a pre-trained Deep Neural Network (DNN) and a General Regression Neural Network (GRNN) is proposed to first classify the blur type and then estimate its parameters. To the best of our knowledge, this is the first time that pre-trained DNN and GRNN have been applied to the problem of blur analysis. Firstly, our method identifies the blur type from a mixed input of image patches corrupted by various blurs with different parameters, rather than making the assumption of a single blur type as in current practice. To this aim, a supervised DNN is trained to project the input samples into a discriminative feature space, in which the blur type can be easily classified. Then, for each blur type, the proposed GRNN estimates the blur parameters with very high accuracy. Experiments demonstrate the effectiveness of the proposed method in several tasks with better or competitive results compared to the state-of-the-art on two standard image datasets, i.e., the Berkeley segmentation dataset and the Pascal VOC 2007 dataset. In addition, blur region segmentation on a number of real photographs shows that our method outperforms previous techniques even for locally blurred images.

6.3.2 Introduction

The restoration of blurred photographs, i.e., image deblurring, is the process of inferring latent sharp images with inadequate information of the degradation model. There are different approaches to solve this problem. On the one hand, according to whether the blur kernel is known, deblurring methods can be categorized into *blind* and *non-blind*

((164), (165), (166), (167)). Non-blind deblurring requires the prior knowledge of the blur kernel and its parameters, while in blind deblurring we assume that the blurring operator is unknown. In most situations of practical interest the Point Spread Function (PSF) is not acquired, so the application range of the blind deblurring (128) is much wider than that of the non-blind deblurring. In real applications, a single blurred image is usually the only input we have to deal with. Classical blind deconvolution methods involve improving image priors in the maximum a-posteriori (MAP) estimation. In terms of image priors, sparsity priors ((165), (168), (169), (83), (170)) and edge priors ((166), (134)) are commonly considered in the literature. Although image prior based methods can successfully estimate the kernels as well as the latent sharp images, there are flaws which restrict their applications. The major flaw of sparsity priors is that they can only represent very small neighborhoods (171). The edge prior methods, largely depending on the image content, will easily fail when the image content is homogeneous. In this chapter, a “learned prior” based on the Fourier transform is proposed for the blur kernel estimation. The frequency domain feature and deep architectures solve the issue of no edges in some of the natural image patches. Though the input is patch-based, our framework can handle larger image patches compared to sparsity priors.

On the other hand, a blurred image can be either *locally* or *globally* blurred. In real applications, locally blurred images are more common, for instance, due to multiple moving objects or different depths of field. As most previous methods focus on image deblurring or blur kernel estimation for a single type of blur, significant attention should be paid to *blur type classification*, because the type of blur is usually unknown in photographs or various regions within a single picture. Despite its importance, only a limited number of methods have been proposed for blur type classification. One typical example is applying a Bayes Classifier using handcrafted blur features, for instance, local autocorrelation congruency (5). Another similar method has been proposed by Su *et al.* (6) based on the alpha channel feature, which has different circularity of the blur extension. Though both of them managed to detect local blurs in real images, their methods are based on handcrafted features.

Although the methods based on handcrafted features can perform well in the cases shown in (5) (6), their applicability is still limited due to the diversity of natural images (7). Based on the success from our previous work, a novel blur parameter estimation method is proposed in this section to extend the parameter identification to estimation under the same two-step framework. Targeting realistic blur estimation, we attempt to handle two difficulties in this section. One of them is blind blur parameter estimation from a single (either locally or globally) blurred image without doing any deblurring. A two-stage framework is proposed: first, a pre-trained DNN is chosen for accomplishing the feature extraction and classification to determine the blur type; second, different samples with the same blur type will be sent to the corresponding GRNN blocks for the parameter estimation. A deep belief network is trained only for weight initialization in an unsupervised way. The DNN uses the weights and the backpropagation to ensure more effective training in a supervised way. The other challenge is the pixel-based blur segmentation using classified blur types. Similar to the first step in the above method, the proposed pre-trained DNN is applied for identifying blur types of all the patches within the same image.

This section makes five contributions:

- To our knowledge, this is the first time that pre-trained DNN has been applied to the problem of blur analysis.
- A two-stage framework is proposed to estimate the blur type and parameter for any given image patch degraded by spatially invariant blur of an unknown type.
- GRNN is first explored in this chapter as a regression tool for blur parameter estimation after the blur type is determined.
- The proposed framework is also applied to real images for local blur classification.

6.3.3 Restricted Boltzmann Machines

An Restricted Boltzmann Machine (RBM) is a type of undirected graphical model which contains undirected, symmetric connections between the input layer (observations) and the hidden layer (representing features). There are no connections between the nodes within the same layer. Suppose that the input layer is $\mathbf{h}^{k-1}$, and the hidden layer is $\mathbf{h}^{k}$, $k = 2, 3, 4....$ The probabilities in the representation model are determined by the energy of the joint configuration of the input layer and the output layer, which can be expressed as:

$$E(\mathbf{h}^{k-1}, \mathbf{h}^k; \theta) = \\ -\sum_{i=1}^{H_{k-1}} \sum_{j=1}^{H_k} w_{ij} h_i^{k-1} h_j^k - \sum_{i=1}^{H_{k-1}} b_i h_i^{k-1} - \sum_{j=1}^{H_k} c_j h_j^k \tag{6.14}$$

where $\theta = (\mathbf{w}, \mathbf{b}, \mathbf{c})$ denotes the model parameters, w_{ij} represents the symmetric interaction term between unit i in the layer $\mathbf{h}^{k-1}$ and unit j in the layer $\mathbf{h}^k$. b_i and c_j are the bias terms of the nodes i and j, respectively.

In an RBM, the output units are conditionally independent given the input states. So an unbiased sample from the posterior distribution can be obtained when an input data-vector is given, which can be expressed as:

$$P(\mathbf{h}|\mathbf{v}) = \prod_i P(h_i|\mathbf{v}) \tag{6.15}$$

Since $h_i \in 0, 1$, the conditional distributions are given as:

$$p(h_j^k = 1|\mathbf{h}^{k-1}; \theta) = \sigma(\sum_{i=1}^{H^{k-1}} w_{ij} h_i^{k-1} + c_j) \tag{6.16}$$

$$p(h_i^{k-1} = 1|\mathbf{h}^k; \theta) = \sigma(\sum_{j=1}^{H^k} w_{ij} h_j^k + b_i) \tag{6.17}$$

where $\sigma(t) = (1 + e^{-t})^{-1}$.

As shown in the above equation, weights between two layers and the biases of each layer decide the energy of the joint configuration. The training process of the RBM is to update $\theta = (\mathbf{w}, \mathbf{b}, \mathbf{c})$ by Contrastive Divergence (CD) (148).

The intuition for CD is: the training vector on the input layer is used for the inference of the output layer, so the units of the output layer have been updated as well as the weights connected between layers. Afterwards, another inference goes from the output layer to the input layer with more updates of the weights and input biases. This process is carried out repeatedly until the representation model is built.

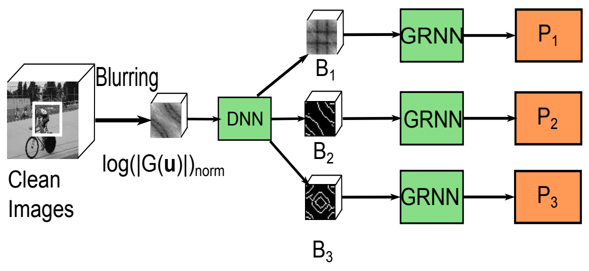


Figure 6.4: The proposed architecture: DNN is the first stage for blur type classification, which has 3 output labels. GRNN is the blur PSF parameter estimation, which has different output labels for each blur type. P_1, P_2, and P_3 are the estimated parameters, which can be seen in Sec. 6.3.8.3. B_1, B_2, and B_3 are the features for Gaussian, motion, and defocus blur, respectively.

6.3.4 Methodology

In this section, we describe the proposed two-stage framework (Fig. 6.4) for blur classification and parameter estimation. We explain the problem formulation, the proposed blur features, the training of DNN, and the structure of the GRNN in Sec. 6.1.3, Sec. 6.1.4, Sec. 6.3.5, and Sec. 6.3.6, respectively.

6.3.5 The Training Process of Deep Neural Networks

Deep belief nets are used as a generative model for feature learning in a lot of previous work (26). In this chapter, we first construct the DBN by unsupervised greedy layer-wise training to extract features in the form of hidden layers. Then the weights in these hidden layers serve as the initial values for a neural network. In this process, the neural network is trained in a supervised way.

6.3.5.1 Regularization Terms

Given that

$$E(\mathbf{h}^{k-1}, \mathbf{h}^k; \theta) = -\log P(\mathbf{h}^{k-1}, \mathbf{h}^k) \tag{6.18}$$

Assume the training set is $\mathbf{h}_1^{k-1}, ..., \mathbf{h}_m{}^{k-1}$, the following regularization term is proposed for reducing the chance of overfitting:

$$\min_{\{w_{ij}, b_i\, c_j\}} -\sum_{p=1}^{m} \log \sum_{h} P(\mathbf{h}_p^{k-1}, \mathbf{h}_p^{k}) \tag{6.19}$$

$$+\lambda \sum_{j=1}^{n} |t - \frac{1}{m} \sum_{p=1}^{m} E[h_j^{(pk)} | h^{(p(k-1))}]|^2 \tag{6.20}$$

where $E[\cdot]$ is the conditional expectation given the data, t is the constant controlling the sparseness of the hidden units h_j^k, and λ is a constant for the regularization. In this way, the hidden units are restricted to have a mean value closing to t.

6.3.5.2 The pretrained deep neural network

- The input layer is trained in the first RBM as the visible layer. Then, a representation of the input blurred sample is obtained for further hidden layers.
- The next layer is trained as an RBM by greedy layer-wise information reconstruction. The training process of RBM is to update weights between two adjacent layers and the biases of each layer.
- Repeat the first and second steps until the parameters in all layers (visible and all hidden layers) are learned.
- In the supervised learning part, the above trained parameter W, b, a are used for initializing the weights in the deep neural network.

$$\hat{y}_k = \sigma(\sum_{j=0}^{M} w_{kj}^{(l+1)} h(\sum_{i=0}^{d} w_{ji}^{(l)} x_i))$$
$$l = 1, 2, ..., N-1$$
$$k = 1, 2, ..., K$$

The goal for the optimization process is to minimize the backpropagation error derivatives:

$$\phi^* = \arg\min_{\phi} [-\sum_{p} \mathbf{y}_p \log \hat{\mathbf{y}}_p] \tag{6.21}$$

Evaluate the error signals for each output and hidden unit using back-propagation of error (172).

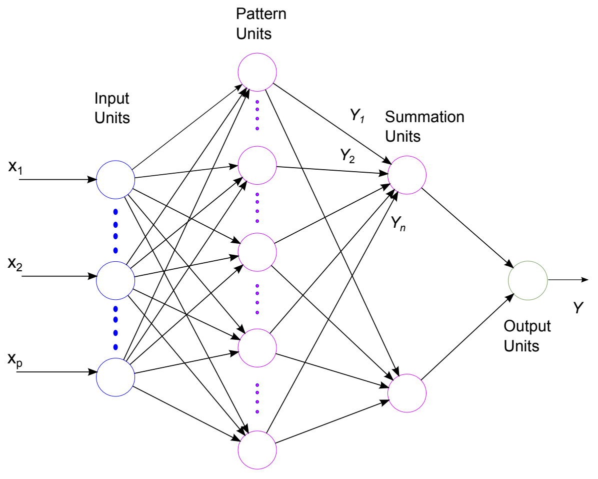


Figure 6.5: The diagram of GRNN.

6.3.6 General Regression Neural Network

The general regression neural network is considered to be a generalization of both Probabilistic Neural Networks (PNN) and Radial Basis Function Networks (RBFN) (173). It outperforms RBFN and backpropagation neural networks in terms of the results of prediction (174). The main goal of a GRNN is to estimate a joint probability density function of the input independent variables and the output.

As shown in Fig. 6.5, "GRNN (174) is composed of an input layer, a hidden layer, "unnormalized" output units, a summation unit, and normalized outputs. GRNN is trained using a one-pass learning algorithm without any iterations." Intuitively, in the training process, the target values for the training vectors help to define cluster centroids, which act as part of the weights for the summation units.

Assume that the training vectors can be represented as X and the training targets are Y. In the pattern layer, each hidden unit is corresponding to an input sample. From the pattern layer to the summation layer, each weight is the target for the input sample. The summation units can be denoted as:

$$\hat{Y} = \frac{\sum_{i=1}^{n} Y_i \exp(-D_i^2/2\sigma^2)}{\sum_{i=1}^{n} \exp(-D_i^2/2\sigma^2)} \tag{6.22}$$

where $D_i^2 = (X - X_i)^T(X - X_i)$, σ is the spread parameter.

In the testing stage, for any input T, the Euclidean distance between this input and the hidden units are calculated. In the summation layer, the weighted average of the possible 'target' is calculated for each hidden node and then averaged by the normalization process.

6.3.7 Forming the Two-phase Structure

The proposed method is formed by two-stage learning (Fig. 6.4). First, the identification of blur patterns is carried out by using the logarithmic spectra of the input blurred patches. The output of this stage is 3 labels: the Gaussian blur, the motion blur and the defocus blur. With the label information, the classified blur vectors will be used in the second stage for blur parameter estimation. At this stage, motion blur and defocus blur will be further preprocessed by the edge detector (Sec. 6.1.4) before the training but Gaussian blur vectors remain the same (As shown in our previous experiments (7), the appropriate feature for Gaussian blur is the logarithmic spectra without edge detection). This stage outputs various estimated parameters for individual GRNN as shown in Sec. 6.3.8.3.

6.3.8 Experiments

6.3.8.1 Experimental setup

The training datasets, testing datasets, the parameters for our Pre-training part of the DNN (DBN training) are the same as the ones in our previous work (7). In this section, the biggest difference is the parameter estimation stage, which is the GRNN training.

GRNN Training For parameters of the GRNN training, there is a smoothness-of-fit parameter σ that needs to be tuned. A range of values [0.02, 1] with the intervals of 0.1 has been used for determining the parameter, which is shown in Fig. 6.6. The value $\sigma = 0.2$ is selected for our implementation.

6.3.8.2 Image blur type classification

In this experiment, the parameter setting for our DNN remains the same.

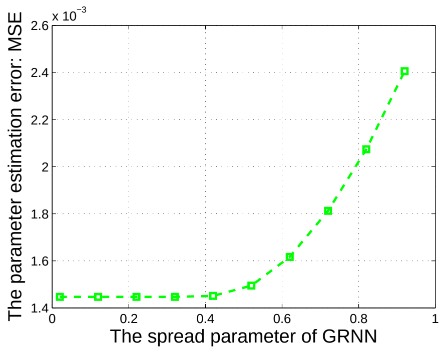


Figure 6.6: The estimation error changes with the spread parameter of GRNN. The parameter testing was done on the data which are corrupted with Gaussian blur with various kernels.

Table 6.4: Comparison of obtained average results on the two testing datasets with the state-of-the-art. CR1 is the Berkeley dataset, and CR2 is the Pascal dataset.

Method	Features	CR1	CR2
(5)		78.1%	79.4%
(6)	Handcrafted	80.7%	81.5%
(153)		76.9%	78.8%
NN (151)		89.7%	90.2%
CNN (152)	Learned	92.2%	93.9%
Proposed		**94.5%**	**95.2%**

We can observe from Table 6.4 that algorithms based on learned features perform better than those based on handcrafted features, which suggests that learning-based feature extractor is less restricted to the type of the blur we consider. Meanwhile, our method performs best among all the algorithms using automatically learned features.

6.3.8.3 Blur kernel parameter estimation

In this experiment, the parameters of the blur kernels are estimated through GRNN. For different blur kernels, different parameters are estimated as explained in Sec. 6.1.3. The parameters are set as: 1) Gaussian blur has a range: $\sigma = [1, 5]$; 2) Motion blur has $\omega = [30, 180]$; 3) defocus blur: $R = [2, 23]$. The architectures in each GRNN are the same.

The first comparison is between our previous method (7) and the method proposed in this chapter, through which we would like to see the improvement by using the regression rather than the classification. Table 6.5 has shown the performance of the image deblurring using the estimated parameters. One can see that apart from the Gaussian blur, both results of the other two types have been improved significantly by using parameter estimation instead of classification. Visual results of this experiment are also shown in Fig. 6.10.

Table 6.5: Quantitative comparison of the proposed method and the previous method (7). The results shown are the average values obtained on the synthetic test set.

	Gaussian blur		Motion blur		Out-of-focus blur	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Input	25.11	0.6624	24.72	0.6413	22.33	0.6157
(7)	28.82	0.8669	26.73	0.8221	26.41	0.8008
Ours	28.96	0.8786	27.94	0.8415	27.67	0.7991

The other type of comparisons are made between our methods and other regression methods. Specifically, our method is compared to the back-propagation Neural Network, Support Vector Regressor (SVR) (175), and pre-trained DNN plus linear regressor (the same input layer of the blur features but continuous targets instead of discrete labels). As shown in Fig. 6.7, our GRNN method achieves the best results among all,

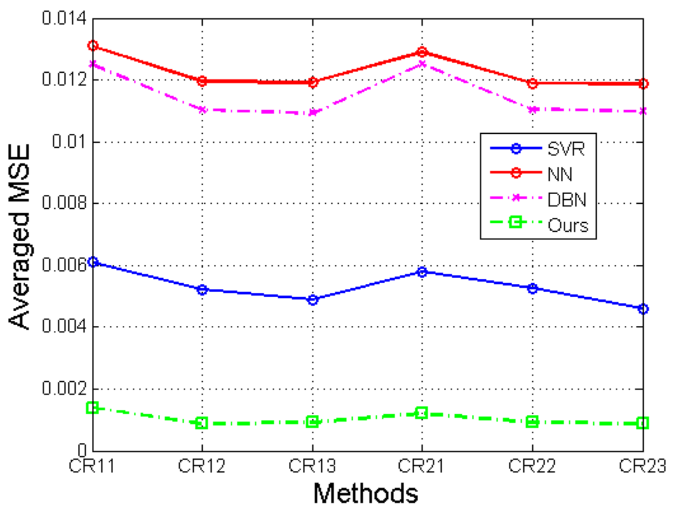


Figure 6.7: The parameter estimation was done on the data which are corrupted with different blur kernels with various sizes. In CRxx the first x refers to the dataset type (1 for Berkeley and 2 for Pascal) and the second x refers to the blur type (the Gaussian blur, the motion blur, and the defocus blur).

which demonstrate the fact mentioned in (174)(176) that GRNN yields better results compared to back-propagation neural network. As can be seen from the figure, SVR performs much better than neural networks with our input data, which also proves that determining prediction results directly from the training data seems to be a better scheme for our problem compared to the weight tuning in the back-propagation frameworks. Moreover, our proposed GRNN works better than the pre-trained DNN with a linear regressor as shown in Fig. 6.7, which shows that GRNN is a better regressor for the blur analysis.

6.3.9 Deblurring synthetic test images using the estimated values

Once the blur type and the parameter of the blur kernel are estimated, it is easier to use non-blind image reconstruction method EPLL (167) to restore the latent image. The restored images are compared with the results of several popular blind image deblurring methods in the case of motion blur (easier for fair comparisons).

The quantitative reconstruction results are presented by the cumulative histogram (177) of the deconvolution error ratio across test datasets in Fig. 6.8. The error ratio in

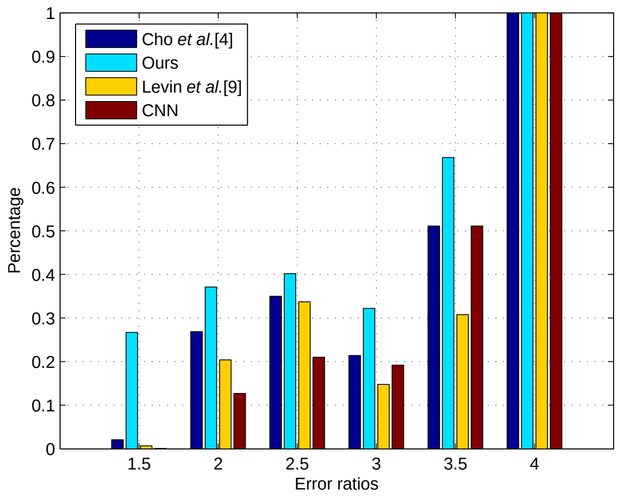


Figure 6.8: Cumulative histogram of the deconvolution error ratios.

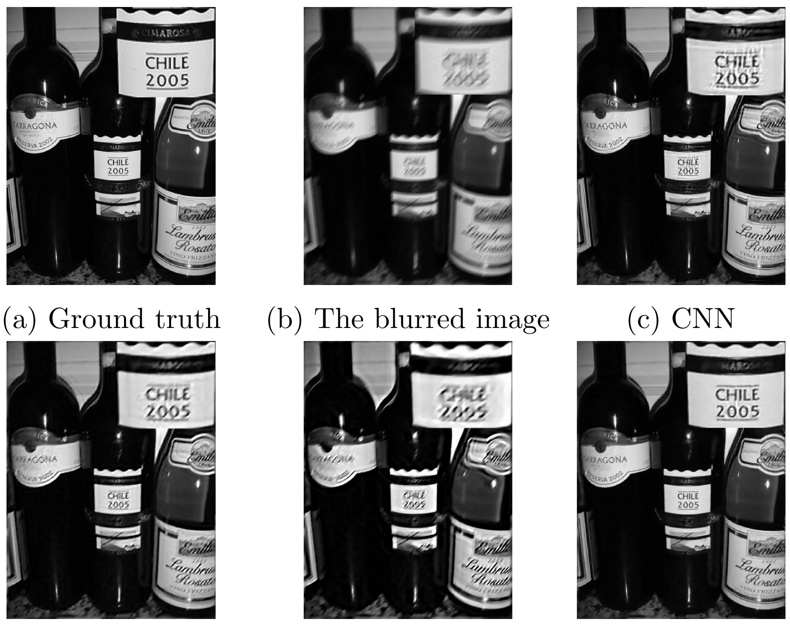
this figure is calculated by comparing the two types of SSD error between reconstructed images and the ground truth images. One of them is the restored results using estimated kernel and the other one is with the truth kernel.

The deconvolved images are shown in the following Fig. 6.9. Contrary to the quantitative results, it is obvious that our deblurred images have very competitive visual quality. Our method outperforms CNN a lot due to the fact that our GRNN step can provide much more precise parameter estimation.

6.3.10 Blur region segmentation on the real photographs

In this experiment, our DNN structure is trained on real photographs, from which blurred training patches are extracted. The blur types of the patches are manually labeled. 200 partially blurred images are selected from Flickr.com. Half of these images are used for training and the other half are used for testing according to what has been described in paper (5). The size of each patch is still the same compared to previous experiments (32 by 32). Using the blur type classification results by our proposed method, we also consider the spatial similarity of blur types in the same region mentioned by Liu *et al.* 's (5).

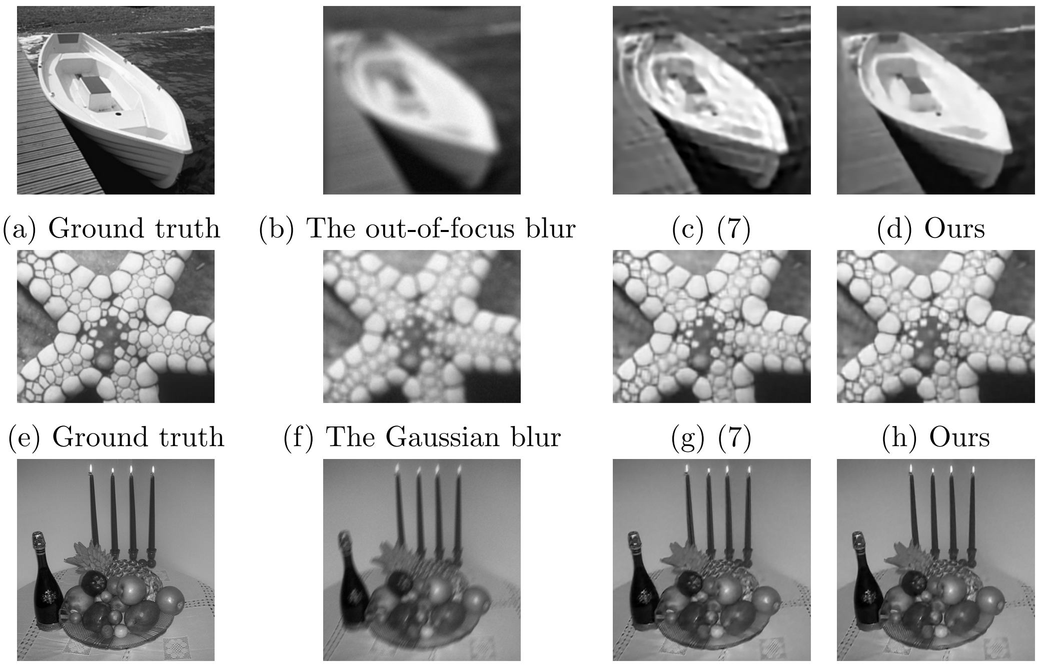
The segmentation result of our method is compared with (5) and (6) in Fig. 6.11.



(a) Ground truth (b) The blurred image (c) CNN

(d) Levin *et al.* (83) (e) Cho *et al.* (166) (f) Ours

Figure 6.9: Comparison of the deblurred results of images corrupted by motion blur with length 10 and angle 45.



(a) Ground truth (b) The out-of-focus blur (c) (7) (d) Ours

(e) Ground truth (f) The Gaussian blur (g) (7) (h) Ours

(i) Ground truth (j) The motion blur (k) (7) (l) Ours

Figure 6.10: Comparison of the deblurred results of different images corrupted by various blur kernels.

As can be seen from these subjective results, our classification is more solid even when the motion is significant. This is useful for real deblurring applications.

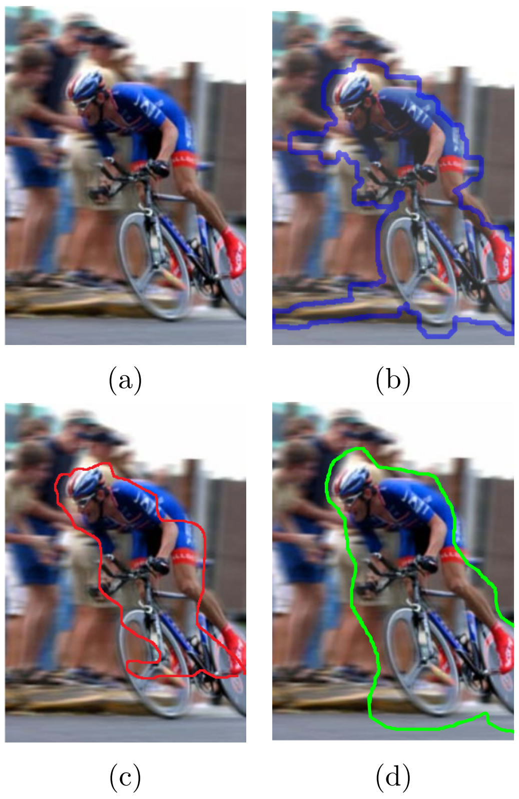


Figure 6.11: Comparison of the blur segmentation results for partially blurred images.(a) input blurred image; (b) blur segmentation result in (5); (c) blur segmentation result in (6); (d) our result.

6.4 Conclusions

In this section, a learning-based blur estimation method has been proposed for blind blur analysis. Our training samples are generated by patches from abundant datasets, after the Fourier transform and our designed edge detection. In the training stage, a pre-trained DNN has been applied in a supervised way. That is, the whole network is trained in an unsupervised manner by using DBN and afterwards the backpropagation fine-tunes the weights. In this way, a discriminative classifier can be trained. In the parameter estimation stage, a strong regressor GRNN is proposed to deal with our problem of blind parameter estimation. The experimental results have demonstrated the superiority of our proposed method compared to the state-of-the-art methods for applications such as blind image deblurring and blur region segmentation for real blurry images.

7

Conclusions and Future Work

7.1 Conclusions

In this thesis, our goal is to improve the representation models for image restoration. Based on the traditional categories of image denoising algorithms, non-local models, sparse coding models, genetic programming, and deep learning models are mainly exploited for the application of image restoration.

First, in the nonlocal model, despite the fact that this model has successfully utilized image self-similarity in most circumstances, there are deficiencies worth mentioning. For instance, lack of candidates for certain noisy patches, or high noise levels interfering with the similarity terms between patches. Therefore, we have proposed a solution to these problems by using a pre-classification before the nonlocal process, which is a clustering based on moment invariants. Also, Rotationally Invariant Block Matching (RIBM) is proposed to improve block matching for the actual weighted averaging. It has been proved through many experiments that this approach has improved the original nonlocal means by a significant amount.

In our second line of research, in order to fully exploit the sparsity of image models to handle high noise levels or noise models rather than Gaussian, the major sparse models are combined in this work, which are image self-similarity, pre-learned and fixed representations. The multi-resolution structure and sparsity of wavelets are employed by nonlocal dictionary learning in each decomposition level of the wavelets. We have demonstrated experimentally that the proposed method outperforms two of the state-of-the-art denoising algorithms on higher noise levels and removing uniform noise.

The third work presented in this thesis is related to the machine learning theory Genetic Programming (GP). Previous local spatial filter tried to interpret the local patch model as a polynomial expression, which works well in some occasions. Inspired by the success of sparse coding, we intended to design a learning-based filter which could take advantages of local filters, such as Gaussian filter, bilateral filter, etc. In this work, a patch clustering is used and GP is applied afterwards for determining the optimal filter for each cluster. In the testing stage, the optimal filter trained by GP will be retrieved and employed on the input noisy patch. Extensive experiments verify that this method can compete and outperform the state-of-the-art denoising methods in the case of removing Gaussian or salt-and-pepper noise. At the same time, the computational efficiency has been improved significantly too.

The final main line of research is deep learning for image restoration. In this part, three major contributions are presented. First, a two-stage framework has been proposed for blur type classification and parameter estimation. Second, the deep belief network from the first work has been used in the noise level identification. Third, the pre-trained DBN is used for initializing the neural network, which is the blur type classification step. Following this, the general regression neural network is proposed to estimate the parameter in continuous values. Experiments show the effectiveness of the above several applications of deep learning with the classification rate of the blur type, the reconstructed blurred image, and the segmentation results of images with mixed blur types.

7.2 Future Work

From the research in this thesis, one can draw the conclusion that several properties of the image representation model are very critical for improving the results of image restoration, which are **clustering**, **hierarchical structures**, **sparsity**, and **deep architectures**.

Clustering was used in both my spatial domain chapter 3 and the sparse coding work chapter 4. Aiming at designing adaptive representations, clustering is a very useful tool for classifying image textures, which could provide the opportunities for image models to be adaptive for different types of image textures in the same scale. Though clustering can be very useful for adaptive image models, it can still be less accurate when images are very corrupted. For instance, when images are very noisy, the classical Euclidean distance is prone to fail. Under such circumstances, more distortion invariant features should be used for the clustering process. Or, the clustering could happen in the iterative ways. In our future work, making image representation models more structured is necessary. For instance, in the framework of convolutional neural networks, nodes in the small neighborhood share the same weights between the current layer and adjacent layers. However, more structures could be exploited through the process of the CNN learning. Either a pre-clustering could be done before the CNN process, or we could guide the CNN with structured architectures by different weight constraints.

Hierarchical structures was proven by wavelets that it mimics the human visual perception by providing image models in different resolution scales, which overcomes the artifacts in previous representation models (e.g. BLS-GSM, BM3D, etc.). This structure has contributed to our proposed sparse coding algorithm with a more adaptive model (our chapter 4), which allows the method to perform different levels of thresholding in different scales. However, the number and size of layers in the hierarchical structures remain uncertain for most algorithms. In our future work, whenever we build a representation model with multiple layers, we need to think about this issue. For instance, in deep neural network, there might be other ways of initializing the hierarchical structures rather than using DBN with heuristically decided number and size of layers. For instance, bilinear projection or other trained coefficients regarding certain data. Bilinear projection can provide more discriminative coefficients for classification problems, and trained coefficients from other methods, such as dictionary learning, can provide even more meaningful starting coefficients for the hierarchical structure.

Sparsity is an idea closely related to image compression. In this thesis, we have shown that wavelets and sparse coding can both exploit image sparsity, which ends up with more adaptive image representation models. When the image representation is very sparse, it means the current image model has found the intrinsic structure in the input images. In the setting of image restoration, that means the model can help with better reconstruction (removing distortions). For our future work, for other machine learning tasks, for instance, deep belief networks, we need to pay attention to sparsity too.

Deep architectures perform surprisingly well for tasks like object recognition and image classification because of its ability to extract semantic features from image data without human expert knowledge. However, these deep learning methods still perform not well enough in the problem like image denoising / deblurring. On the one hand, from the experiments and experience in this thesis, we can see that the quality of the input would affect the ability of deep architectures extracting semantic features. For instance, in the application of using DBN for image noise level classification, it is very difficult for DBN to extract any useful features from the input noisy image patches, especially when the noise level is too high. This worths studying that unless the noise is following a set pattern or structure, it will be difficult for DBN to learn anything useful from it. On the other hand, From a manifold perspective, natural image data forms “tangled manifolds” for the last layer of neural network to separate with. With functions like softmax, the performance could stay at a certain level without improving because of this. Therefore, the combination of deep architectures with traditional classifier like K-nearest neighbor could be useful [1]. In our future work, I think making this deep model better for image restoration is a challenging direction. It is interesting to explore the manifold of natural image patches to see whether the last layer of the deep architecture could be revised into a better layer for separating noise and image structures.

[1]http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Bibliography

[1] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Int. Conf. Computer Vision, 2009*, Kyoto, Japan, 2009. xi, 5, 6, 17, 27, 28, 29, 33, 68, 69, 70, 71, 72, 74, 75, 76, 78, 95

[2] J. Mairal, "Sparse coding for machine learning, image processing and computer vision," Ph.D. dissertation, Ecole Normale Superieure de Cachan, May 2010. xi, 4, 6, 8

[3] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Conf. Advances in Neural Information Processing Systems*, Lake Tahoe, Nevada, U.S., 2012. xi, 12, 13, 14

[4] G. Sven, Z. Sebastian, and W. Joachim, "Rotationally invariant similarity measures for nonlocal image denoising," *J. Visual Comm. and Image Representation*, vol. 22, no. 2, pp. 117–130, Feb. 2011. xiii, 17, 21, 48, 49, 50, 54, 55, 56, 57, 59, 60, 61, 62, 63, 64, 65, 66

[5] R. Liu, Z. Li, and J. Jia, "Image partial blur detection and classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Anchorage, AL, USA, 2008. xv, 115, 117, 123, 131, 138, 141, 144

[6] B. Su, S. Lu, and C. Tan, "Blurred image region detection and classification," in *Proc. ACM Multimedia*, Scottsdale, AZ, USA, 2011. xv, 115, 117, 123, 131, 138, 141, 144

[7] R. Yan and L. Shao, "Image blur classification and parameter identification using two-stage deep belief networks," in *Proc. Conf. British Machine Vision Conference*, Bristol, UK, 2013. xviii, 95, 131, 137, 139, 143

[8] B. Hunt, "Bayesian methods in nonlinear digital image restoration," *IEEE Trans. on Computers*, vol. 26, no. 3, pp. 219–229, Mar. 1977. 1

[9] L. Shao, R. Yan, X. Li, and Y. Liu, "From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithm," *IEEE Trans. Cybern.*, to be published, 2013. 2, 94

[10] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithm, with a new one," *Simul*, vol. 4, pp. 490–530, 2005. 2, 33, 35, 36, 125

[11] A. Efros and T. Leung, "Texture synthesis by nonparametric sampling," in *Proc. IEEE Int. Conf. Computer Vision*, Kerkyra, Greece, 1999. 2

[12] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. on Computers*, vol. 23, no. 1, pp. 90–93, Jan. 1974. 2, 68

[13] S. Mallat, *A wavelet tour of signal processing.* London: Academic Press, 2008. 2, 23, 27, 68

[14] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007. 2, 17, 24, 25, 33, 48, 69, 76

[15] W. T. Freeman and E. H. Adelson, "The design and the use of steerable filters," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, Sep. 1991. 2, 23, 68

[16] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003. 2, 17, 24, 33, 48, 68, 69, 94

[17] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006. 2, 17, 27, 28, 29, 31, 33, 40, 68, 69, 80, 83, 94, 95

[18] G. Peyre, "Sparse modeling of textures," *Journal of Mathematical Imaging and Vision*, vol. 34, no. 1, pp. 17–31, May 2009. 2

[19] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Transactions on Image Process.*, vol. 18, no. 1, pp. 27–36, Jan. 2009. 5

[20] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, Jan. 2009. 9

[21] Y. Bengio, A. Courville, and P. Vincent, "Representation learning a review and new perspective," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Jan. 2013. 10, 11

[22] A. Mohamed, T. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Proc. IEEE Int. Conf. Acoustics, Speeach and Signal Processing*, Prague, Czech Republic, 2011. 10

[23] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pretraining help deep learning ?" *Journal of Mach. Learning Research*, vol. 11, pp. 625–660, Jan. 2010. 10

[24] G. Huang, H. Lee, and E. Learned-Miller, "Learning hierarchical representations for face verification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012. 10

[25] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 19, no. 3, pp. 51–59, Jan. 1996. 10

[26] S. Zhong, Y. Liu, and Y. Liu, "Bilinear deep learning for image classification," in *Proc. ACM Multimedia*, Scottsdale, AZ, USA, 2011. 11, 117, 120, 134

[27] P. Vincent, H. Larochelle, Y. Bengio, and M. P., "Extracting and composing robust features with denoising autoencoders," in *Proc. IEEE Int. Conf. Machine Learning*, Helsinki, Finland, 2008. 11

[28] F. Agostinelli, M. Anderson, and H. Lee, "Adaptive multi-column deep neural network with application to robust image desnoising," in *Proc. Conf. Advances in Neural Information Processing Systems*, Harrahs and Harveys, Nevada, U.S., 2013. 12

[29] J. H. Hong, S. B. Cho, and U. K. Cho, "A novel evolutionary approach to image enhancement filter design: method and applications," *IEEE Trans. Syst. Man Cybern.B, bern.*, vol. 39, no. 6, pp. 1446–1457, Dec. 2009. 15

[30] J. H. Wang, W. J. Liu, and L. D. Lin, "Histogram-based fuzzy filter for image restoration," *IEEE Trans. Syst. Man Cybern. B, bern.*, vol. 32, no. 2, pp. 230–238, Apr. 2002. 15

[31] C. S. Lee, S. M. Guo, and C. Y. Hsu, "Genetic-based fuzzy image filter and its application to image processing," *IEEE Trans. Syst. Man Cybern. B, bern.*, vol. 35, no. 4, pp. 694–711, Aug. 2005. 15

[32] Y. Li, S. Kang, N. Joshi, S. S., and H. D., "Generating sharp panoramas from motion blurred videos," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010, pp. 2424–2431. 16, 114

[33] L. Shapiro and G. Stockman, *Computer Vision.* Prentice Hall, 2001. 16, 18, 19, 52, 94

[34] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* New York: Wiley, 1949. 16

[35] B. Widrow and S. Haykin, *Least-mean-square adaptive filters.* Wiley-IEEE, 2003. 16

[36] L. Shao, H. Zhang, and G. de Haan, "An overview and performance evaluation of classification-based least squares trained filters," *IEEE Trans. Image Process.*, vol. 17, no. 10, pp. 1772–1782, Oct. 2008. 16, 19, 33, 47, 94, 100

[37] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Computer Vision (ICCV), 1998*, Bombay, India, 1998, pp. 839–846. 16, 47, 49, 94, 96, 101

[38] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 12, pp. 629–639, Jul. 1990. 16, 18, 19, 94

[39] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007. 16, 17, 18, 106

[40] X. Zhu and P. Milanfar, "Automatic parameter selection for denoising algorithms using a no-reference measure of image content," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3116–3132, Dec. 2010. 16, 19, 20, 33

[41] P. Bouboulis, K. Slavakis, and S. Theodoridis, "Adaptive kernel-based image denoising employing semi-parametric regularization," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1465–1479, Jun. 2010. 16, 20, 33, 94, 107

[42] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. San Diego, CA, USA: IEEE Press, 2005, pp. 60–65. 16, 21, 22, 33, 48, 49, 59

[43] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005. 17, 21, 48, 50

[44] P. Coupe, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot, "An optimized blockwise nonlocal means denoising filter for 3-d magnetic resonance images," *IEEE Trans. Med. Imag.*, vol. 27, no. 4, pp. 425–441, Apr. 2008. 17, 21, 48, 50

[45] T. Thaipanich, O. Byung Tae, W. Ping-Hao, X. Daru, and C. C. J. Kuo, "Improved image denoising with adaptive nonlocal means (anl-means) algorithm," *IEEE Trans. Consum. Electron.*, vol. 56, no. 4, pp. 2623–2630, Nov. 2010. 17, 21, 25, 48, 50, 58, 59, 98, 100

[46] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, "Fast non-local algorithm for image denoising," in *Proc. IEEE Int. Conf. Image Process.*, Atlanta,GA,USA, 2006, pp. 1429–1432. 17, 21, 48, 52

[47] B. Goossens, H. Luong, A. Pizurica, and W. Philips, "An improved non-local denoising algorithm," in *Proc. Int. Workshop on Local and Non-local Approximation in Image Process.*, Tuusalu, Finland, 2008, pp. 143–156. 17, 21, 22, 33, 48

[48] P. Chao, O. C. Au, D. Jingjing, Y. Wen, and Z. Feng, "A fast nl-means method in image denoising based on the similarity of spatially sampled pixels," in *Proc. IEEE Int. Workshop on Multimedia Signal Processing*, Rio de Janeiro, Brazil, 2009, pp. 1–4. 17, 21, 48

[49] D. Tschumperle and L. Brun, "Non-local image smoothing by applying anisotropic diffusion pde's in the space of patches," in *Proc. IEEE Int. Conf. Image Process.*, Cairo, Egypt, 2009, pp. 2957–2960. 17, 21, 48

[50] J. Wei, "Lebesgue anisotropic image denoising," *Int. J. Imag. Syst. and Tech.*, vol. 15, no. 1, pp. 66–73, Jul. 2005. 17, 21

[51] C. Kervrann and J. Boulanger, "Local adaptivity to variable smoothness for exemplar-based image denoising and representation," INRIA, Tech. Rep., Jul. 2005. 17, 21, 22

[52] Y. Lou, P. Favaro, and S. Soatto, "Nonlocal similarity image filtering," UCLA Computational and Applied Mathematics, LA,CA,USA, Tech. Rep., Apr. 2008. 17, 21

[53] S. Zimmer, S. Didas, and J. Weickert, "A rotationally invariant block matching strategy improving image denoising with non-local means," in *Proc. Int. Workshop on Local and Non-local Approximation in Image Process.*, 2008, pp. 143–156. 17, 21

[54] R. Yan, L. Shao, S. D. Cvetkovic, and J. Klijn, "Improved nonlocal means based on pre-classification and invairant block matching," *IEEE /OSA J. Display Technol.*, vol. 8, no. 4, pp. 212–218, Apr. 2012. 17, 94

[55] J. C. Brailean, R. P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, and R. L. Lagendijk, "Noise reduction filters for dynamic image sequences: a review," *Proc. IEEE*, vol. 83, no. 9, pp. 1272–1292, Sep. 1995. 17

[56] E. P. Simoncelli and E. H. Adelson, "Noise removal via bayesian wavelet coring," in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, 1996, pp. 379–382. 17

[57] J. L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, Jun. 2002. 17

[58] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005. 17

[59] E. Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 423–438, Apr. 2005. 17, 23, 68

[60] F. Luisier, T. Blu, and M. Unser, "A new sure approach to image denoising: Interscale orthonormal wavelet thresholding," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 593–606, Mar. 2007. 17

[61] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognition*, vol. 43, no. 4, pp. 1531–1549, Apr. 2010. 17, 26, 33

[62] F. Luisier, T. Blu, and M. Unser, "Sure-let for orthonormal wavelet-domain video denoising," *IEEE Trans. Circuits Syst. deo Technol.*, vol. 20, no. 6, pp. 913–919, Jun. 2010. 17

[63] A. Pizurica, P. Aleksandra, and S. Wink, "A review of wavelet denoising in mri and ultrasound brain imaging," *Current Medical Imag. Reviews*, vol. 2, no. 2, pp. 247–260, May 2006. 17, 26

[64] R. Yan, L. Shao, and Y. Liu, "Nonlocal hierarchical dictionary learning using wavelets for image denoising," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4689–4698, Dec. 2013. 17, 95

[65] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *IEEE Conf. Computer Vision and Pattern Recognition*, Colorado, USA, 2011. 17, 28, 30, 31, 33

[66] G. Peyre, "A review of adaptive image representations," *IEEE J. Selected Topics Signal Process.*, vol. 5, no. 5, pp. 896–911, Sep. 2011. 17

[67] K. Vladimir, F. Alessandro, E. Karen, and A. Jaakko, "From local kernel to nonlocal multiple-model image denoising," *Int.J.Computer Vision*, vol. 86, no. 1, pp. 1–32, Jan. 2010. 17, 27, 70, 74

[68] L. Rudin and S. Osher, "Total variation based image restoration with free local constraints," in *Proc. IEEE Int. Conf. on Image Processing*, Austin, Texas, USA, 1994. 18, 94

[69] S. M. Smith and J. M. Brady, "Susan - a new approach to low level image processing," *Int. J. Computer Vision*, vol. 23, no. 1, pp. 45–78, May 1997. 18, 19, 94

[70] X. Liu, M. Tanaka, and M. Okutomi, "Noise level estimation using weak textured patches of a single noisy image," in *Proc. IEEE Int. Conf. Image Processing*, Florida, U.S.A., 2012, pp. 665–668. 20

[71] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 35, no. 4, pp. 849–862, April. 2013. 23

[72] D. Donoho, "Wedgelets: nearly- minimax estimation of edges," *Annals of Stat.*, vol. 27, no. 3, pp. 859–897, Jun. 1999. 23, 68

[73] E. Candes and D. L. Donoho, "Recovering edges in ill-posed inverse problems: optimality of curvelet frames," *Annals of Stat.*, vol. 30, no. 3, pp. 784–842, Jun. 2002. 23, 68

[74] ——, "New tight frames of curvelets and the problem of approximating piecewise c2 image with piecewise c2 edges," *Comm. Pure Appl. Math.*, vol. 57, pp. 219–266, Nov. 2003. 23, 68

[75] S. Mallat and E. L. Pennec., "Bandelet image approximation and compression," *SIAM Mul. Model. Simul.*, vol. 4, no. 3, pp. 992–1039, Mar. 2005. 23, 68

[76] M. Do and M. Vetterli, "Frame pyramids," *IEEE Trans. Signal Process.*, vol. 51, no. 9, pp. 2329–2342, Sep. 2003. 23, 68

[77] S. Mallat and G. Peyre, "Orthogonal bandlet bases for geometric images approximation," *Communication on Pure and Applied Mathematics*, vol. 61, no. 9, pp. 1173–1212, Feb. 2008. 23, 68

[78] I. Daubechies, *Ten lectures on wavelets.* Philadelphia, PA, USA: SIAM, 2006. 23, 27

[79] M. Vetterli and J. Kovacevic, *Wavelets and subband coding.* Prentice Hall, 1995. 23, 27

[80] D. L. Donoho, "Denoising by soft-thresholding," *IEEE Trans. Inform. Theory*, vol. 41, no. 3, pp. 613–627, May 1995. 23, 27

[81] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaption via wavelet shrinkage," *Biometrika*, vol. 81, pp. 425– 455, 1994. 23, 27

[82] J. Guerrero-Colon and J. Portilla, "Deblurring-by-denoising using spatially adaptive gaussian scale mixtures in overcomplete pyramids," in *Proc. IEEE Int. Conf. Image Processing*, Atlanta, GA, USA, 2006, pp. 625–628. 24

[83] A. Levin, Y. Weiss, F. Durand, and W. Freeman, "Efficient marginal likelihood optimization in blind deconvolution," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Colorado Springs, CO, USA, 2011. 25, 131, 142

[84] B. A. Olshausen and D. J. Field, "Emergence of simplecell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, Jun. 1996. 27

[85] ——, "Sparse coding with an overcomplete basis set: A strategy employed by v1 ?" *Vis. Res.*, vol. 37, no. 23, pp. 311–325, Dec. 1996. 27

[86] K. Engan, S. Aase, and J. Hakon-Husoy, "Method of optimal directions for frame design," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, 1999, pp. 2443– 2446. 27

[87] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T. Lee, and T. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neur. Comput.*, vol. 15, no. 2, pp. 349–396, Feb. 2003. 27

[88] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neur. Comput.*, vol. 12, no. 2, pp. 337–365, Feb. 2000. 27

[89] L. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decompositiion," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Philadelphia, PA, USA, 2005, pp. 2272–2279. 27

[90] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, Jan. 2008. 27, 31

[91] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1438–1451, Jul. 2009. 27, 31, 68, 69, 74, 98

[92] J. Mairal, F. Bach, and G. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. pp, no. 19, pp. 1–14, Aug 2011. 30

[93] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004. 33, 34, 57, 104, 106

[94] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 1772–1782, Feb. 2008. 33, 36

[95] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3440– 3451, Nov. 2008. 36

[96] S. Yaser and D. Psaltis, "Recognitie aspects of moment invariants," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 6, no. 12, pp. 698–706, Nov. 1984. 53, 56

[97] D. Mackay, *Information Theory, Inference and Learning Algorithms.* Cambridge University Press., 2003. 54

[98] J. Salmon, "On two parameters for denoising with nonlocal means," *IEEE Signal Process. Lett.*, vol. 17, no. 10, pp. 269–272, Mar. 2010. 58

[99] F. Luisier, T. Blu, and M. Unser, "Image denoising in mixed poisson-gaussian noise," *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 696–708, Mar. 2011. 68

[100] K. R. Castleman, *Digital image processing*, Prentice Hall, Toronto, 1996. 68

[101] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011. 68

[102] M. Elad, M. A. T. Figueiredo, and M. Y., "On the role of sparse and redundant representations in image processing," *IEEE Proc. Special Issue on Applications of Sparse Representations and Compressive Sensing*, vol. 98, no. 6, pp. 972–982, Apr. 2010. 69, 71

[103] E. Simoncelli and W. T. Freeman, *The steerable pyramid: A flexible architecture for multi-scale derivative computation*, Philadelphia, PA., USA, 1995. 69

[104] B. Olshausen, P. Sallee, and M. Lewicki, "Learning sparse image codes using a wavelet pyramid architecture," in *Proc. Conf. Advances in Neural Information Processing Systems*, Denver, CO, USA, 2000, pp. 887–893. 69

[105] P. Sallee and B. Olshausen, "Learning sparse multiscale image representations," in *Proc. Conf. Advances Neural Information Processing Systems*, vol. 15, Vancouver, Canada, 2002, pp. 1327–1334. 69

[106] M. Elad, *Sparse and redundant representations: From theory to applications in signal and image processing.* New York, NY 10013, USA: Springer Science Business Media, 2010. 69, 75

[107] B. Ophir, M. Lustig, and M. Elad, "Multi-scale dictionary learning using wavelets," *IEEE J. Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1014–1024, Sept. 2011. 70, 74, 75, 81

[108] J. Huang, X. Huang, and D. Metaxas, "Learning with dynamic group sparsity," in *Proc. IEEE Int. Conf. Computer Vision*, Kyoto, Japan, 2009, pp. 64–71. 74

[109] E. Candes, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted l1 minimization," *J. Fourier Analysis and Applications*, vol. 14, pp. 877–905, Nov. 2008. 77

[110] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *IEEE Signal Process. Lett.*, 2009, pp. 689–696. 78, 79

[111] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–409, 2004. 79

[112] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse represenions for image and video restoration," *Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008. 83

[113] S. J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Trans. Circuits And Systems*, vol. 38, no. 9, pp. 984–993, Sept. 1991. 94

[114] Y. Li, H. Wei, and S. Billings, "Identification of time-varying systems using multi-wavelet basis functions," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 656–663, May 2011. 94

[115] ——, “Time-varying linear and nonlinear parametric model for granger causality analysis,” *Physical Review E*, vol. 85, no. 4, 2012. 94

[116] N. Petrovic and V. Crnojevic, “Impulse noise detection based on robust statistics and genetic programming,” in *Proc. 7th Int. Conf. Advanced Comcepts for Intelligent Vision Systems (ACIVS), 2005*, Antwerp, Belgium, 2005, pp. 643–649. 95

[117] ——, “Evolutionary tree-structured filter for impulse noise removal,” in *Proc. 8th Int. Conf. Advanced Comcepts for Intelligent Vision Systems (ACIVS), 2006*, Antwerp, Belgium, 2006, pp. 103–113. 95

[118] N. I. Petrovic and V. Crnojevic, “Universal impulse noise filter based on genetic programming,” *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1109–1120, July 2008. 95

[119] J. R. Koza, *Automatic discovery of reusable programs.* Cambridge, MA, USA: MIT Press, 1994. 96

[120] B. Zhang and B. K. Gunturk, “Multiresolution bilateral filtering for image denoising,” *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2324–2333, 2008. 101, 106

[121] M. Zhang and B. Gunturk, “A new image denoising framework based on bilateral filter,” in *Proc. SPIE Visual Communications and Image Processing, 2008*, San Diego, CA, USA, 2008, pp. 6822–68 221B. 101

[122] D. Atkins, K. Neshatian, and M. Zhang, “A domain independent genetic programming approach to automatic feature extraction for image classification,” in *Proc. IEEE Congress on Evolutionary Computation (CEC), 2011*, Wellington, New Zealand, 2011, pp. 238–245. 101

[123] D. Martin, D. Fowlkes, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proc. IEEE Int. Conf. Computer Vision*, Vancouver, Canada, 2001. 102, 122

[124] H. Burger, C. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with bm3d?” in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, Providence, RI, US, 2012. 102

[125] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming.* Lulu Enterprises, UK, 2008. 103

[126] K. Toh and N. Isa, “Noise adaptive fuzzy switching median filter for salt-and-pepper noise reduction,” *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 281–284, Mar. 2010. 107

[127] R. Lagendijk and J. Biemond, *Basic methods for image restoration and identification.* London: Academic Press, 2000. 114

[128] M. Almeida and L. Almeida, “Blind and semi-blind deblurring of natural images,” *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 36–52, Aug. 2010. 114, 131

[129] J. Cai, H. Ji, C. Liu, and Z. Shen, “High-quality curvelet-based motion deblurring from an image pair.” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1566–1573. 114

[130] J. Chen, L. Yuan, C. Tang, and Q. L., “Robust dual motion deblurring,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1–8. 114

[131] A. Likas and N. Galatsanos, “A variational approach for bayesian blind image deconvolution,” *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2222–2233, Aug. 2004. 114

[132] F. Rooms, W. Philips, and J. Portilla, “Parametric psf estimation via sparseness maximizaion in the wavelet domain,” in *Proc. SPIE Wavelet Application in Industrial Processing II*, San Diego, CA, USA, 2004. 114

[133] I. Rekleitis, “Optical flow recognition from the power of spectrum of a single blurred image,” in *Proc. IEEE Int. Conf. on Image Processing*, Lausanne, Switzerland, 1996. 114

[134] S. Cho, S. Paris, B. Horn, and W. Freeman, “Blur kernel estimation using the radon transform,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Colorado Springs, CO, USA, 2011. 114, 119, 122, 131

[135] F. Krahmer, Y. Lin, B. McAdoo, K. Ott, J. Wang, D. Widemann, and B. Wohlberg, “Technical report, mathematical modeling in industry x workshop,” in *Technical Report*, Institute for Mathematics and its Applications, University of Minnesota, 2006. 114

[136] V. Jain and H. Seung, “Natural image denoising with convolutional networks,” in *Proc. Conf. Advances in Neural Information Processing Systems*, Vancovour, Canada, 2008. 115, 126

[137] A. Ciancio, A. Costa, E. Silva, A. Said, R. Samadani, and P. Obrador, “No-reference blur assessment of digital pictures based on multifeature classifiers,” *IEEE Trans. on Image Process.*, vol. 20, no. 1, pp. 64–75, Jan. 2011. 115

[138] R. Molina, J. Mateos, and A. Katsaggelos, “Blind deconvolution using a variational approach to parameter, image, and blur estimation,” *IEEE Trans. on Image Process.*, vol. 15, no. 12, pp. 3715–3727, Nov. 2006. 116

[139] W. Hu, J. Xue, and N. Zheng, “Psf estimation via gradient domain correlation,” *IEEE Trans. on Image Process.*, vol. 21, no. 1, pp. 386–392, Jan. 2012. 116

[140] F. Chen and J. Ma, “An empirical identification method of gaussian blur parameter for image deblurring,” *IEEE Trans. on Signal Process.*, vol. 57, no. 7, pp. 2467–2478, Mar. 2009. 117

[141] D. Kundur and D. Hatzinakos, “Blind image deconvolutions,” *IEEE Signal Processing Mag.*, pp. 43–63, 1996. 117

[142] M. Cannon, “Blind deconvolution of spatially invariant image blurs with phase,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 1, pp. 58–63, Jan. 1976. 118

[143] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proc. Conf. Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 2006. 118, 122

BIBLIOGRAPHY

[144] S. Wu, Z. Lu, and E. Ong, "Blind image blur identification in cepstrum domain," in *Proc. IEEE Int. Conf. Computer Communications and Networks*, Honolulu, Hawaii, U.S., 2007. 118

[145] P. Bao, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 27, no. 9, pp. 1485–1490, Sept. 2005. 119

[146] W. Mcllhagga, "The canny edge detector revisited," *International Journal of Computer Vision*, vol. 91, no. 3, pp. 251–261, Feb. 2011. 119

[147] G. Watson. Statistics on spheres, 1983. 119

[148] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002. 121, 133

[149] S. Roth and M. J. Black, "Field of experts: a framework for learning image priors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Diego, CA, USA, 2005. 122

[150] M. Everingham, V. G. L., C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2007 results," 2007. 122

[151] T. Mitchell, *Machine learning*, 1997. 123, 124, 129, 138

[152] R. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Master's thesis, Technical University of Denmark, DTU Informatics*, 2012. 123, 124, 138

[153] S. Duan, K.and Keerthi, "Which is the best multiclass svm method? an empirical study," *Lecture Notes in Computer Science*, vol. 3541, no. 8, 2005. 123, 124, 138

[154] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *IEE Proc. Vis., Image, Signal Process.*, vol. 146, no. 2, pp. 80–84, Aug. 1999. 125

[155] A. Amer, A. Mitiche, and E. Dubois, "Reliable and fast structure-oriented video noise estimation," in *Proc. IEEE Int. Conf. Image Process.*, Rochester, New York, 2002, pp. 840–843. 125

[156] B. Cornera, R. Narayanana, and S. Reichenbach, "Noise estimation in remote sensing imagery using data masking," *Int. J. Remote Sens.*, vol. 24, no. 4, pp. 689–702, Apr. 2003. 125

[157] D. Shin, R. Park, S. Yang, and J. Jung, "Block-based noise estimation using adaptive gaussian filtering," *IEEE Trans. Consumer Electron.*, vol. 51, no. 1, pp. 218–226, Feb. 2005. 125

[158] C. Liu, R. Szeliski, S. Kang, C. Zitnick, and W. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 299–314, Feb. 2008. 125

[159] S. Olsen, "Estimation of noise in images: An evaluation," *Graph. Models Image Process.*, vol. 55, no. 4, pp. 319–323, Feb. 1993. 125

[160] X. Liu, M. Tanaka, and M. Okutomi, "Sing-image noise level estimation for blind denoising," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5226–5237, Dec 2013. 125

[161] Z. Daniel and W. Yair, "Scale invariance and noise in natural images," in *Proc. IEEE Conf. Comput. Vis.*, Kyoto, Japan, 2009. 127

[162] M. Malinowski and M. Fritz, "Learning smooth pooling regions for visual recognition," in *Proc. British Machine Vision Conference*, Bristol, UK, 2013. 128, 129

[163] C. Chang and C. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27–27, May 2011. 129

[164] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Proc. Conf. Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 2009. 131

[165] R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. Freeman, "Removing camera shake from a single photograph," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 787–794, Jul. 2006. 131

[166] S. Cho and S. Lee, "Fast motion deblurring," in *Proc. ACM SIGGRAPH Asia*, Pacifico Yokobama, Japan, 2009. 131, 142

[167] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Computer Vision*, Barcelona, Spain, 2011. 131, 140

[168] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 721–730, Aug. 2008. 131

[169] N. Joshi, R. Szeliski, and D. Kriegman, "Psf estimation using sharp edge prediction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Anchorage, AL, USA, 2008. 131

[170] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalizaed sparsity measure," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Colorado Springs, CO, USA, 2011. 131

[171] L. Sun, S. Cho, J. Wang, and J. Hays, "Edge-based blur kernel estimation using patch priors," in *Proc. IEEE Int. Conf. Computational Photography*, Cambridge, MA, USA, Apr. 2013. 131

[172] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989. 135

[173] M. Amin, H. Davande, A. Sadeghian, and S. Chartier, "Feedback associative memory based on a new hybrid model of generalized regression and self-feedback neural networks," *Neural Networks*, vol. 23, no. 7, pp. 892–904, Sept. 2010. 136

[174] D. Tomandl and A. Schober, "A modified general regression neural network with new, efficient training algorithms as a robust black box tool for data analysis," *Neural Netw.*, vol. 14, no. 8, pp. 1023–1034, Oct. 2001. 136, 140

[175] S. Gunn, "Support vector mahines for classification and regression," in *Technical Report*, School of Electronics and Computer Science, University of Southampton, 1998. 139

[176] Q. Li, Q. Meng, J. Cai, H. Yoshino, and A. Mochida, "Predictiing hourly cooling load in the building: A comparison of support vector machine and different artificial neural networks," *Ener. Conv. Manage.*, vol. 50, no. 1, pp. 90–96, Jan. 2009. 140

[177] A. Levin, Y. Weiss, F. Durand, and W. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami Beach, FL, USA, 2009. 140