UNIVERSITY OF CALIFORNIA
SANTA CRUZ

GLOBAL IMAGE FILTERING

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Hossein Talebi

September 2015

The Dissertation of Hossein Talebi
is approved:

Professor Peyman Milanfar, Chair

Professor James Davis

Professor Alex Pang

Dean Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Global Image Filtering

by

Hossein Talebi

The state-of-the-art digital photography has made great progress over the past decades; however, imaging still suffers from distortions such as noise and blur. The result is an increased demand for more efficient and effective computational photography algorithms. In this dissertation a new data-dependent image filtering scheme is proposed. More specifically, various image enhancement applications from denoising to image editing are thoroughly explained. The proposed filters exploit the existing self-similarity of images to introduce a new set of basis functions capable of efficiently describing image components.

First, by focusing on the local similarities of images, measured by pixel affinities, a spatially adapted filtering strategy capable of improving performance of the existing local filters is introduced. The filter's strength is tuned by estimating the local signal-to-noise ratio (SNR), such that high SNR image patches are filtered more aggressively and low SNR patches are treated conservatively.

Second, we explore the global similarity of images and introduce a new image filtering scheme based on the spectrum of global affinities. The global filter is derived from a fully connected graph representing the image, and can be approximated using the Nyström extension. Using this, we drive an approximation to the spectral (principal) components of the global filter, which can be implemented efficiently by sampling a fairly small percentage of the pixels in the image. These orthonormal eigenfunctions

are highly expressive of the coarse and fine details in the underlying image, where each eigenvector can be interpreted as one scale of a data-dependent multiscale image decomposition. In this filtering scheme, each eigenvalue can boost or suppress the corresponding signal component in each scale. Experiments illustrate that the mapping of the eigenvalues by an appropriate polynomial function endows the filter with a number of important capabilities, such as edge-aware sharpening, denoising, tone manipulation and abstraction.

Lastly, asymptotic performance of the global denoising filter is analyzed to show that its performance always improves as a function of image size, regardless of image content. The rate of this improvement is estimated as an upper bound on the mean-squared-error (MSE).

This dissertation is dedicated to my family.

Acknowledgments

I wish to express my sincere appreciation to those who have contributed to this Ph.D dissertation and supported me throughout this journey.

First of all, I am extremely grateful to my advisor, Professor Peyman Milanfar, for his continuous support, patience, motivation, and immense knowledge. Peyman's guidance and deep insight helped me at various stages of my five years of research. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank the rest of my dissertation committee, Professor James Davis and Professor Alex Pang, for their insightful comments and encouragement.

I thank my former and current fellow labmates in MDSP lab, Hiroyuki Takeda, Hae-Jong Seo, Priyam Chatterjee, Xiang Zhu, Sujoy Biswas, Amin Kheradmand, M. Hossein Daraei, and Robert Sumner. It has been fun working with you all and thanks for offering your help whenever needed. They are excellent researchers and great friends.

Lastly, I would like to thank my family for all their love and encouragement. I am thankful to my parents, who taught me the value of hard work, self-respect, persistence and about how to be independent. My parents are great role models of resilience, strength and character, and I am grateful for them. I am thankful to my sisters, Elahe and Elham, and my brother, Behrooz who always believed in me and encouraged me to follow my dreams. Most of all, I am grateful to my loving, supportive and patient wife Jesica, whose faithful support throughout and during the final stages of this Ph.D is appreciated.

Santa Cruz, California

July 26, 2015

Hossein Talebi

Chapter 1

Data-dependent Filtering

Abstract – The problem of image filtering is introduced and discussed in this chapter. We also discuss the various sources of image perturbations and the existing image filtering approaches. The chapter concludes with our proposed non-parametric restoration framework.

1.1 Introduction

Digital cameras and camera phones have made photography an integral part of our lives. Millions of photos are taken and shared on the internet on a daily basis. This has led to increasing demand for computational photography algorithms. Images taken by existing commercial cameras are corrupted by distortions, such as noise, blur, etc. For a better understanding of the sources of these distortions, basic characteristics of the camera pipeline are described in the following.

A typical camera imaging pipeline [11] is shown in Fig. 1.1. The scene radiance is transmitted through the camera lens and converted to electrons in CCD or CMOS

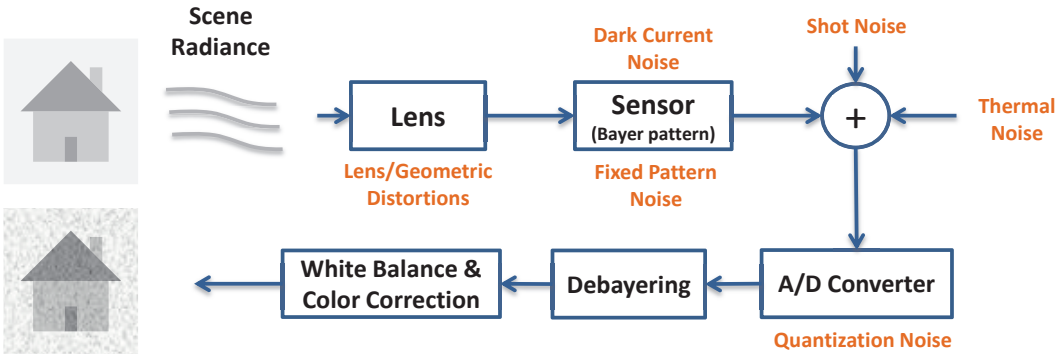


Figure 1.1: Camera imaging pipeline.

sensor. Typical sensors use a color filter array to produce a *Bayer* pattern image of green, red and blue colors. The analog Bayer image is converted to digital pixel values, which are then fed to the debayering interpolator to create three separate red, green and blue images. White balancing, gamma correction and color correction are other adjustments applied on the image before producing the final output.

There are two main distortion sources in the transformation of photons to digital pixels: blur and noise. In general, there are various types of blur; including: motion blur, lens blur and post processing blur. Motion blur is due to the movement between camera and object. It is common when using commercial cameras that are hard to keep steady during the exposure time. Lens blur is usually caused by shallow depth of field and incorrect lens setting. The post processing blur is a natural side effect of enhancement operators such as denoising, super-resolution, etc.

Noise has different sources related and independent from the camera pipeline. As shown in Fig. 1.1, characteristics of the camera sensor may cause the off-set dark current noise which is caused by thermal energy of the sensor elements in the absence of light. There is also fixed pattern noise that is due to the uneven voltage values of the

sensor units under the same light. The thermal noise produced by electronic devices of the pipeline is another source of perturbation that is negligible in short exposure time scenarios. The analog-to-digital converter is also considered to generate uniform quantization noise, especially when the image bit-rate is small.

Unlike the above mentioned noise sources, shot noise is independent of the camera pipeline. Shot noise is associated with the particle nature of the light photons. The number of photons hitting the camera sensor is signal dependent and is described by the Poisson distribution [12]. However, for well-exposed images with enough photons accumulated in each sensor unit, the probability distribution function of the noise is modeled by a white Gaussian noise [13] (zero-mean independent and identically distributed Gaussian). On the other hand, in low light imaging scenarios where the Poisson noise is signal dependent, the Anscombe transform [14] could be used to stabilize the noise variance and approximate it with a Gaussian distributed signal.

Beside denoising and deblurring, other post-processing operations, such as local tone-mapping and high dynamic range imaging are among high interest filtering topics. A scene with a variation in radiance much larger than the range which can be recorded by a conventional camera or displayed on a monitor may result in low quality photos. The problem of strong contrast reduction from the scene radiance to the displayable range while preserving the image details is addressed by tone-mapping.

Due to these inevitable distortions, image filtering is a very basic image processing and computer vision problem. This dissertation provides a comprehensive analysis of the data-dependent filters with a focus on image denoising, and editing. Furthermore, applications of the filter for the purpose of local tone-mapping, contrast enhancement and sharpening are introduced and discussed.

1.2 Contributions

The contributions and implications of this dissertation are organized and discussed in the following chapter summaries. Each chapter makes its own significant argument and builds toward an integrative approach that expands the field of image enhancement.

- **Chapter. 2 Spatially Adaptive Iterative Filtering**

  We propose a spatially-adapted iterative filtering (SAIF) strategy capable of controlling the denoising strength *locally* for any given spatial domain method. The proposed method iteratively filters local image patches, and the iteration method and iteration number are automatically optimized with respect to local MSE, which is estimated from the given image.

- **Chapter. 3 Global Filter**

  Although SAIF does not set any theoretical limitation over the local window size, computational burden of building a matrix filter for a window as large as the whole image is prohibitively high. The Nyström method [15] gives a practical solution when working with huge affinity (similarity) matrices by operating on only a small portion of the complete matrix to produce a low-rank approximation.

- **Chapter. 4 Global Image Denoising**

  We introduce an innovative global image denoising (GLIDE) filter, which takes into account all informative parts of an image. Distinctly, with this global filter in hand, the concept of patch-based processing is no longer restrictive, and we are able to show that the existing patch-based filters are improved upon.

- **Chapter. 5 Asymptotic Analysis of the Global Filter**

We take the analysis of the GLIDE filter a step deeper to show that the performance of the global denoiser always improves as a function of image size, regardless of image content. Moreover, we prove that the improvement rate is a function of the sparsity of the image in a naturally constructed basis adapted to the content of the image.

- **Chapter. 6 Global Image Editing**

  We propose ways to extend our global filtering method to other image processing tasks such as detail manipulation and edit propagation. Having eigenfunctions of the global filters, smoothing or sharpening operators are easily implemented by means of mapping the corresponding eigenvalues. The global nature of the eigenvectors let us propagate these edits throughout the image.

The data-dependent filters and existing image denoising methods are reviewed in the remainder of this chapter.

1.3 Existing Image Denoising Methods

There have been numerous denoising algorithms, and in general they can be divided into two main categories: transform domain methods, and spatial domain methods.

Transform domain methods are developed under the assumption that the clean image can be well represented as a combination of few transform basis vectors, so the signal-to-noise-ratio (SNR) can be estimated and used to appropriately shrink the corresponding transform coefficients. Specifically, if a basis element is detected as belonging to the true signal, its coefficient should be mostly preserved. On the other hand, if

an element is detected as a noise component, its coefficient should be shrunk more, or removed. By doing this, noise can be effectively suppressed while most structures and finer details of the latent image are preserved.

Different algorithms in this category vary in either the transform selection or the shrinkage strategy. Fixed transforms (e.g. wavelet, DCT) are often employed as in [5, 16], and are easy to calculate. However, they may not be effective in representing natural image content with sparse coefficient distributions, and that would inevitably increase the requirement on the shrinkage performance. Non-fixed transforms are also applied. For example, Muresan [17] and Zhang [4] use principle component analysis (PCA). Compared with fixed transformations, PCA is more adaptive to local image content. However, such decompositions can be quite sensitive to noise. K-SVD [18] and K-LLD [19] use over-complete dictionaries generated from training, which is more robust to noise but computationally expensive. The shrinkage strategy is another important factor that needs to be fully considered. Though there are many competing strategies, it has been shown that the Wiener criterion, which determines the shrinking strength according to (estimated) SNR in each basis element, is the best strategy that gets close to the optimal performance with respect to mean-squared-error (MSE) [20]. In fact, in practice it has achieved state-of-the-art denoising performance with even simple fixed transforms (such as DCT in BM3D) [5].

Spatial domain methods concentrate on a different noise suppression approach, which estimates each pixel value as a weighted average of other pixels, where higher weights are assigned to more "similar" pixels [1–3, 21, 22]. Pixel similarities can be calculated in various ways. For the bilateral filter, similarity is determined by both geometric and photometric distances between pixels [1]. Takeda et al. proposed a locally

adaptive regression kernel (LARK) denoising method, robustly measuring the pixel similarity based on geodesic distance [3]. Another successful method called non-local means (NLM) extends the bilateral filter by replacing point-wise photometric distance with patch distances, which is more robust to noise [2].[1]

1.4 Non-parametric Restoration Framework

In this section we provide a brief introduction to the non-parametric image restoration framework. In particular, we study the problem of denoising and present some of the spatial domain methods that have been quite successful in the recent past. A spatial domain denoising process has a transform domain filtering interpretation, where the orthogonal basis elements and the shrinkage coefficients are respectively the eigenvectors and eigenvalues of a symmetric, positive definite (data-dependent) filter matrix. For filters such as NLM and LARK the eigenvectors corresponding to the dominant eigenvalues could well represent latent image contents.

Let us consider the measurement model for the denoising problem:

$$y_i = z_i + e_i, \qquad \text{for } i = 1, ..., n, \tag{1.1}$$

where $z_i = z(\mathbf{x}_i)$ is the underlying image at position $\mathbf{x}_i = [x_{i,1}, x_{i,2}]^T$, y_i is the noisy pixel value, and e_i denotes zero-mean white noise[2] with variance σ^2. The problem of denoising is to recover the set of underlying samples $\mathbf{z} = [z_1, ..., z_n]^T$. The complete

[1] Spatial domain methods include any method that is based upon the computation of a kernel that is applied locally to the pixel data directly. It is possible to approximately implement many regularization based methods in this framework, but we do not believe there is a one to one correspondence between kernel spatial domain methods and regularization based Bayesian methods [23].

[2] We make no other distributional assumptions on the noise.

measurement model for the denoising problem in vector notation is:

$$\mathbf{y} = \mathbf{z} + \mathbf{e}. \tag{1.2}$$

As explained in [3, 23] most spatial domain filters can be represented through the following non-parametric restoration framework:

$$\widehat{z}_i = \arg\min_{z_i} \sum_{j=1}^{n} [z_i - y_j]^2 K(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j), \tag{1.3}$$

where $\widehat{z}_i$ denotes the estimated pixel at position x_i, and the weight (or kernel) function $K(\cdot)$ measures the similarity between the samples y_i and y_j at positions $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively.

1.4.1 Kernel Functions and Weighted Averaging

Perhaps the most well-known kernel function is the Bilateral (BL) filter [1], which smooths images by means of a nonlinear combination of nearby image values. The method combines pixel values based on both their geometric closeness and their photometric similarity. This kernel can be expressed in a separable fashion as follows:

$$K_{ij} = \exp\left\{ \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2} + \frac{-(y_i - y_j)^2}{h_y^2} \right\}, \tag{1.4}$$

in which h_x and h_y are smoothing (control) parameters.

The NLM [2] is another very popular data-dependent filter which closely resembles the bilateral filter except that the photometric similarity is captured in a patch-wise manner:

$$K_{ij} = \exp\left\{ \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2} \right\}, \tag{1.5}$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are patches centered at y_i and y_j, respectively. In theory (though not in actual practice,) the NLM kernel has just the patch-wise photometric distance ($h_x \to \infty$).

More recently, the LARK (also called *Steering Kernel* in some publications) [3] was introduced which exploits the geodesic distance based on estimated gradients:

$$K_{ij} = \exp\{-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{C}_{ij} (\mathbf{x}_i - \mathbf{x}_j)\}, \tag{1.6}$$

in which $\mathbf{C}_{ij}$ is a local covariance matrix of the pixel gradients computed from the given data [3]. The gradient is computed from the noisy measurements y_j in a patch around $\mathbf{x}_i$. Robustness to noise and perturbations of the data is an important advantage of LARK.

In general, all of these restoration algorithms are based on the same framework (1.3) in which some data-adaptive kernels are assigned to each pixel contributing to the filtering. Minimizing equation (1.3) gives a normalized weighted averaging process:

$$\widehat{z}_i = \mathbf{w}_i^T \mathbf{y}, \tag{1.7}$$

where the weight vector $\mathbf{w}_i$ is

$$\mathbf{w}_i = \frac{1}{\sum_{j=1}^{n} K_{ij}} [K_{i1}, K_{i2}, \ldots, K_{in}]^T. \tag{1.8}$$

By stacking the weight vectors together, the filtering process for all the sample pixels can be represented simultaneously through a matrix-vector multiplication form

$$\hat{\mathbf{z}} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{bmatrix} \mathbf{y} = \mathbf{W}\mathbf{y}, \tag{1.9}$$

where $\widehat{\mathbf{z}}$ and $\mathbf{W}$ denote the estimated signal and the filter matrix, respectively. While the filter $\mathbf{W}(\mathbf{y})$ is data-dependent, in Chapter 2 we show that our employed filter $\mathbf{W}(\widetilde{\mathbf{z}})$, which is computed based on the *pre-filtered* image $\widetilde{\mathbf{z}}$, can be closely treated as a filter that is not *stochastically* dependent on the input data.

1.4.2 Filter Matrix

$\mathbf{W}$ is a positive row-stochastic matrix (every row sums up to one). This matrix is not generally symmetric, though it has real, positive eigenvalues [23]. The Perron-Frobenius theory describes the spectral characteristics of this matrix [24], [25]. In particular, the eigenvalues of $\mathbf{W}$ satisfy $0 \leq \lambda_i \leq 1$; the largest one is uniquely equal to one ($\lambda_1 = 1$) while the corresponding eigenvector is $\mathbf{v}_1 = \frac{1}{\sqrt{n}}[1, 1, ..., 1]^T$. The last property implies that a flat image stays unchanged after filtering by $\mathbf{W}$.

Although $\mathbf{W}$ is not a symmetric matrix in general, it can be closely approximated with a symmetric positive definite matrix[3] [26]. The symmetrized $\mathbf{W}$ must also stay row-stochastic, which means we get a symmetric positive definite matrix which is doubly (i.e., row- *and* column-) stochastic. The symmetric $\mathbf{W}$ enables us to compute its eigen-decomposition as follows:

$$\mathbf{W} = \mathbf{V}\mathbf{S}\mathbf{V}^T, \tag{1.10}$$

where $\mathbf{S} = \text{diag}[\lambda_1, ..., \lambda_n]$ contains the eigenvalues in decreasing order $0 \leq \lambda_n \leq ... < \lambda_1 = 1$, and $\mathbf{V}$ is an orthonormal matrix $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_n]$ containing the respective eigenvectors of $\mathbf{W}$ in its columns. Since $\mathbf{V}$ is orthogonal, its columns specify a set of basis functions. So the filtering process can be explained as:

$$\widehat{\mathbf{z}} = \mathbf{W}\mathbf{y} = \mathbf{V}\mathbf{S}\mathbf{V}^T\mathbf{y} \tag{1.11}$$

where the input data $\mathbf{y}$ is first transformed into the domain spanned by the eigenvectors of $\mathbf{W}$; then, each coefficient is scaled by the factor λ_i; and finally an inverse transform is applied, yielding the output. From the above analysis we see that the filtering strength

[3]Indeed, it can be shown that $\frac{1}{n}\|\mathbf{W} - \mathbf{W}_{sym}\|_F = O(n^{-\frac{1}{2}})$. That is, the RMS error gets smaller with increasing dimension.

for each basis of a given filter is thus controlled by $\{\lambda_i\}$. Later we will show that mapping λ_i with specific polynomial functions leads to shrinking or boosting the respective signal components and therefore, enabling the filter with various editing applications.

Summary – The importance of image filtering in digital photography was discussed in this chapter. A review of the main sources of image distortions was explained, followed by discussion of the basics of our proposed framework which is founded on the eigen-decomposition of image affinities. In the following chapters we will exploit this paradigm to develop practical filtering applications.

Chapter 2

Spatially Adaptive Iterative Filtering

Abstract – Spatial domain image filters (e.g. bilateral filter, NLM, LARK) have achieved great success in denoising. However, their overall performance has not generally surpassed the leading transform domain based filters (such as BM3D). One important reason is that spatial domain filters lack an efficient way to adaptively fine tune their denoising strength; something that is relatively easy to do in transform domain method with shrinkage operators. In the pixel domain, the smoothing strength is usually controlled *globally* by, for example, tuning a regularization parameter. In this chapter, we propose SAIF[1] (Spatially Adaptive Iterative Filtering), a new strategy to control the denoising strength *locally* for any spatial domain method. This approach is capable of filtering local image content *iteratively* using the given base filter, while the type of iteration and the iteration number are automatically optimized with respect to estimated risk (i.e. mean-squared error). In exploiting the estimated local SNR, we also present a new risk estimator which is different than the often-employed SURE method and exceeds its performance in many cases. Experiments illustrate that our strategy

[1]SAIF is the middle eastern/arabic name for *sword*. This acronym somehow seems appropriate for what the algorithm does by precisely tuning the value of the iteration number.

can significantly relax the base algorithm's sensitivity to its tuning (smoothing) parameters, and effectively boost the performance of several existing denoising filters to generate state-of-the-art results under both simulated and practical conditions.

2.1 Introduction

In practice, determining the denoising strength for spatial domain methods is a general difficulty. For example, these methods always contain some tuning (smoothing) parameters that may strongly affect the denoising performance. A larger smoothing parameter would suppress more noise and meanwhile erase more useful image information, ending up with an over-smoothed (biased) output. Less smoothing would preserve high-frequency signal but also do little denoising (estimation variance). An interesting alternative to tedious parameter tuning is iterative filtering. With this approach, which we promote here, even with a filter estimated from a badly misplaced smoothing parameter, we can still get a well estimated output by applying the same denoising filter several times. But it would seem that the iteration number then becomes another tuning parameter that needs to be carefully treated. Some approaches were developed to handle such parameters. Ramani's Monte-Carlo SURE is capable of optimizing any denoising algorithm with respect to MSE [27], but it requires Gaussian assumption on noise. In [28] we developed a no-reference image content measure named Metric Q to select optimal parameters. However, both Monte-Carlo SURE and Metric Q can only adjust the filtering degree *globally*. Much more efficient estimates could be obtained by smartly changing the denoising strength *locally* as we propose in this chapter. More specifically, we present an approach capable of *automatically* adjusting the denoising strength of spatial domain methods according to local SNR. A second contribution is a

novel method for estimating the local SNR.

As discussed previously, a spatial domain denoising process can always be approximated as a transform domain filter, where the orthogonal basis elements are the eigenvectors of a symmetric and positive definite matrix determined by the filter; and the shrinkage coefficients are the corresponding eigenvalues ranging in $[0, 1]$. For filters such as NLM and LARK the eigenvectors corresponding to the dominant eigenvalues could well represent latent image contents. Based on this idea, we propose a spatially adapted iterative filtering (SAIF) strategy capable of controlling the denoising strength *locally* for any given spatial domain method [29]. The proposed method iteratively filters local image patches, and the iteration method and iteration number are automatically optimized with respect to local MSE, which is estimated from the given image. To estimate the MSE for each patch, we propose a new method called *plug-in* risk estimator. This method is biased and works based on a "pilot" estimate of the latent image. For the sake of comparison, we also derive the often used Stein's unbiased risk estimator (SURE) [30] for the data dependent filtering scheme. While [31] also uses SURE to optimize the NLM kernel parameters, we illustrate that (1) the plug-in estimator can be superior for the same task, and (2) the adaptation approach can be extended to be spatially varying.

2.2 Shrinkage Strategies

Optimal shrinkage strategies based on various spatial domain filters have been discussed in [23], where statistical analysis shows that the optimal filter with respect to MSE is the local Wiener filter with $\lambda_i = \frac{1}{1+\mathrm{snr}_i^{-1}}$, where snr_i denotes signal-to-noise ratio of the i-th channel. However, the local Wiener filter requires exact knowledge of

the local signal-to-noise (SNR) in each basis channel, which is not directly accessible in practice. In denoising schemes such as [5] and [4] the Wiener shrinkage criterion works based on a pilot estimate of the latent image. Still, the Wiener filter's performance strictly relies on accuracy of this estimate. Iterative filtering can be a reliable alternative for reducing sensitivity of the basis shrinkage to the estimated local SNR. Then, the iteration number will be the only parameter to be locally optimized.

To approach the locally optimal filter performance in a stable way, we propose the use of two iterative local operators; namely *diffusion* and *boosting*. In [32] we have shown that performance of any type of kernel could be enhanced by iterative diffusion which gradually removes the noise in each iteration. Yet, diffusion filtering also takes away latent details from the underlying signal. On the other hand, iterative boosting is a mechanism to preserve these lost details of the signal. By using the two iterative filtering methods, we can avoid either over-smoothing or under-smoothing due to incorrect parameter settings. In other words, these two methods provide a way to start with any filter, and properly control the values of shrinkage factors $\{\lambda_i\}$ to achieve a good and stable approximation of the Wiener filter. In the following we discuss the two approaches, separately.

2.2.1 Diffusion Iteration

The idea of diffusion in image filtering was originally motivated by the physical principles of heat propagation and described using a partial differential equation. In our context, we consider the discrete version of it, which is conveniently represented by repeated applications of the same filter as described in [23]:

$$\widehat{\mathbf{z}}_k = \mathbf{W}\widehat{\mathbf{z}}_{k-1} = \mathbf{W}^k\mathbf{y}, \tag{2.1}$$

Each application of $\mathbf{W}$ can be interpreted as one step of anisotropic diffusion with the filter $\mathbf{W}$. Choosing a small iteration number k preserves the underlying structure, but also does little denoising. Conversely, a large k tends to over-smooth and remove noise and high frequency details at the same time. Minimization of MSE (or more accurately an estimate of it) determines when is the best time to stop filtering, which will help avoid under- or over- smoothing.

As long as $\mathbf{W}$ is symmetric, the filter in the iterative model (2.1) can be decomposed as:

$$\mathbf{W}^k = \mathbf{V}\mathbf{S}^k\mathbf{V}^T, \tag{2.2}$$

in which $\mathbf{S}^k = \text{diag}[\lambda_1^k, ..., \lambda_n^k]$. It is worth noting that despite the common interpretation of k as a discrete step, the spectral decomposition of $\mathbf{W}^k$ makes it possible to replace k with any positive real number.

The latent image $\mathbf{z}$ can be written in the column space of $\mathbf{V}$ as $\mathbf{b} = \mathbf{V}^T\mathbf{z}$, where $\mathbf{b} = [b_1, b_2, ..., b_n]^T$, and $\{b_i^2\}$ denote the projected signal energy over all the eigenvectors. As shown in [23] the iterative estimator $\widehat{\mathbf{z}}_k = \mathbf{W}^k\mathbf{y}$ has the following squared bias:

$$\|\text{bias}_k\|^2 = \|(\mathbf{I} - \mathbf{W}^k)\mathbf{z}\|^2 = \sum_{i=1}^{n}(1 - \lambda_i^k)^2 b_i^2, \tag{2.3}$$

Correspondingly, the estimator's variance is:

$$\text{var}(\widehat{\mathbf{z}}_k) = \text{tr}(\text{cov}(\widehat{\mathbf{z}}_k)) = \sigma^2 \sum_{i=1}^{n} \lambda_i^{2k}, \tag{2.4}$$

Overall, the MSE is given by

$$\text{MSE}_k = \|\text{bias}_k\|^2 + \text{var}(\widehat{\mathbf{z}}_k) = \sum_{i=1}^{n}(1 - \lambda_i^k)^2 b_i^2 + \sigma^2\lambda_i^{2k}. \tag{2.5}$$

As the iteration number k grows, the bias term increases, but the variance decays to the constant value of σ^2. Of course, this expression for the MSE is not practically useful

yet, since the coefficients $\{b_i^2\}$ are not known. Later we describe a way to estimate the MSE in practice. But first, let us introduce the second iterative mechanism which we will employ. Boosting is discussed in the following and as we will see, its behavior is quite different from the diffusion filtering.

2.2.2 Twicing Iteration

Although the classic diffusion filtering has been used widely, this method often fails in denoising image regions with low SNR. This is due to the fact that each diffusion iteration is essentially one step of low-pass filtering. In other words, diffusion always removes some components of the noise and signal, concurrently. This shortcoming is tackled effectively by means of boosting which *recycles* the removed components of signal from the residuals, in each iteration. Defining the residuals as the difference between the estimated signal and the noisy signal: $\mathbf{r}_k = \mathbf{y} - \widehat{\mathbf{z}}_{k-1}$, the iterated estimate can be expressed as

$$\widehat{\mathbf{z}}_k = \widehat{\mathbf{z}}_{k-1} + \mathbf{W}\mathbf{r}_k = \sum_{j=0}^{k} \mathbf{W}(\mathbf{I}-\mathbf{W})^j \mathbf{y} = \left(\mathbf{I} - (\mathbf{I}-\mathbf{W})^{k+1}\right)\mathbf{y} \tag{2.6}$$

where $\widehat{\mathbf{z}}_0 = \mathbf{W}\mathbf{y}$. As can be seen, as k increases, the estimate returns to the noisy signal $\mathbf{y}$. In other words, the boosting filter has fundamentally different behavior than the diffusion iteration where the estimated signal gets closer to a constant after each iteration. The squared magnitude of the bias after k iterations is

$$\|\text{bias}_k\|^2 = \|(\mathbf{I}-\mathbf{W})^{k+1}\mathbf{z}\|^2 = \sum_{i=1}^{n} (1-\lambda_i)^{2k+2} b_i^2, \tag{2.7}$$

And the variance of the estimator also is

$$\text{var}(\widehat{\mathbf{z}}_k) = \text{tr}(\text{cov}(\widehat{\mathbf{z}}_k)) = \sigma^2 \sum_{i=1}^{n} \left(1-(1-\lambda_i)^{k+1}\right)^2. \tag{2.8}$$

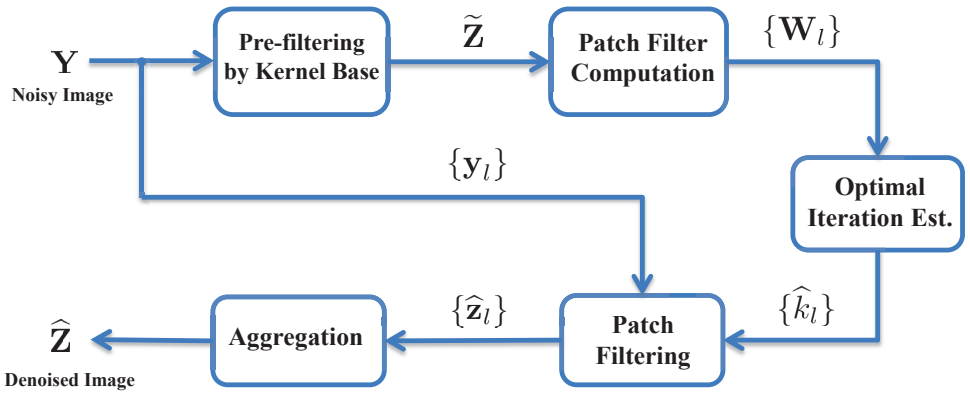


Figure 2.1: Diagram of SAIF method.

Then the overall MSE is

$$\mathrm{MSE}_k = \|\mathrm{bias}_k\|^2 + \mathrm{var}(\widehat{\mathbf{z}}_k) = \sum_{i=1}^{n} (1-\lambda_i)^{2k+2} b_i^2 + \sigma^2 \left(1-(1-\lambda_i)^{k+1}\right)^2. \qquad (2.9)$$

As k grows, the bias term decreases and the variance increases. Contrasting the behavior of the diffusion iteration, we observe that *when diffusion fails to improve the filtering performance, it can be replaced by boosting.* This is the fundamental observation that motivates our approach. More specifically, the contribution of this work is that we simultaneously and automatically optimize the type and number of iterations locally to boost the performance of a given base filtering procedure.

2.3 Practical MSE Estimation

Based on the analysis from Section 2.2 we propose an image denoising strategy which, given any filter using the framework (1.3), can boost its performance by utilizing its spatially adapted transform and by employing an optimized iteration method.

This iterative filtering is implemented patch-wise, so that it is capable of automatically adjusting the local smoothing strength according to local SNR. Fig. 2.1 depicts a block diagram of the proposed approach. Starting from the noisy image $\mathbf{Y}$ and splitting it into N overlapping patches $\{\mathbf{y}_l\}_{l=1}^{N}$, we aim to denoise each noisy patch $\mathbf{y}_l$, separately. To calculate the local filter $\mathbf{W}_l$, we use an estimated image $\widetilde{\mathbf{Z}}$ which is filtered by the standard kernel baseline. Next, MSEs for the two iteration approaches (diffusion and boosting) are estimated for each patch and by comparing their values, the optimal iteration method and consequently the iteration number $\widehat{k}_l$ is selected, generating the filtered patch $\widehat{\mathbf{z}}_l$. Since these filtered patches are overlapped, an aggregation method is finally carried out to compute the denoised image $\widehat{\mathbf{Z}}$. The key steps of this approach are the optimal iteration estimation and the aggregation, which will be described in the rest of this section.

Given a patch $\mathbf{y}$ and its filter matrix $\mathbf{W}$, the task of this step is to select the best iteration method (either diffusion or boosting) and its iteration number that minimizes the MSE. More explicitly, the optimal stopping time $\widehat{k}$ for each iteration method can be expressed as:

$$\widehat{k} = \arg\min_{k} \ \mathrm{MSE}_k \tag{2.10}$$

One way to compute an unbiased estimate of MSE is the often-used SURE [30]. An alternative we propose here is the plug-in risk estimator, which is biased and works based on an estimate of the local SNR. First, note that in practice, eigenvalues and eigenvectors of the filter are always estimated from a *pre-filtered* patch $\widetilde{\mathbf{z}}$, obtained using the base filter with some arbitrary parameter settings. More explicitly we have:

$$\mathbf{W}(\widetilde{\mathbf{z}}) = \mathbf{V}\mathbf{S}\mathbf{V}^T \tag{2.11}$$

It is worth repeating that despite the earlier interpretation of k as a discrete step, the

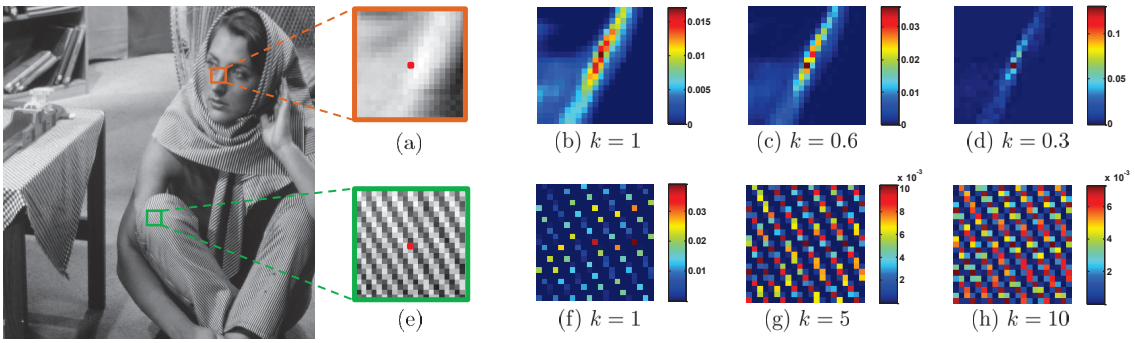


Figure 2.2: Filters based on the NLM kernel with different iteration number k. (a) Smooth patch and the j-th pixel, (b) j-th row of the patch filter $\mathbf{W}$, (c) and (d) j-th row of the iterated patch filter $\mathbf{W}^k$ for different iteration numbers, (e) texture patch and the j-th pixel, (f) j-th row of the patch filter $\mathbf{W}$, (g) and (h) j-th row of the iterated patch filter $\mathbf{W}^k$ for different iteration numbers.

spectral decomposition of $\mathbf{W}^k$ makes it clear that in practice, k can be any positive real number. To be more specific, $\mathbf{W}^k = \mathbf{V}\mathbf{S}^k\mathbf{V}^T$, with $\mathbf{S}^k = \text{diag}[\lambda_1^k, ..., \lambda_n^k]$ where k is any non-negative real number. In actual implementation, the filter can be applied with modified eigenvalues for any $k > 0$. This may seem like a minor point, but in practice can significantly improve the performance as compared to when k is restricted to only positive integers. In effect, a real-valued k automatically and smoothly adjusts the local bandwidth of the filter. Fig. 2.2 illustrates the iterated filters for two different patches. As can be seen, while decreasing the iteration number k can be interpreted as smaller tuning parameter h_y for NLM kernel, larger k is equivalent to a wider kernel.

Next, we discuss the two risk estimators and show that the plug-in can exploit the estimated local SNR to have better performance as compared to the SURE estimator.

2.3.1 Plug-in Estimator

The plug-in estimator is described in Algorithm 1. In this method, risk estimators for diffusion and boosting are computed based on the pre-filtered patch $\widetilde{\mathbf{z}}$, computed using the base filter with arbitrary parameters. More explicitly, the signal coefficients can be estimated as:

$$\widetilde{\mathbf{b}} = \mathbf{V}^T \widetilde{\mathbf{z}} \tag{2.12}$$

This estimate's contribution can be interpreted as equipping the risk estimator with some prior knowledge of the local SNR of the image. The estimated signal coefficients allow us to use (2.5) and (2.9) to estimate MSE_k in each patch:

Algorithm 1: Plug-in Risk Estimator

Input: Noisy Patch: $\mathbf{y}$, Pre-filtered Patch: $\widetilde{\mathbf{z}}$, Patch Filter: $\mathbf{W}$

Output: Denoised Patch: $\widehat{\mathbf{z}}$

1- Eigen-decomposition of the filter $\mathbf{W}(\widetilde{\mathbf{z}}) = \mathbf{V}\mathbf{S}\mathbf{V}^T$

2- $\widetilde{\mathbf{b}} = \mathbf{V}^T\widetilde{\mathbf{z}} \Leftarrow$ Compute the signal coefficients

3- Plug-in$_k^{df}$, Plug-in$_k^{bs} \Leftarrow$ Compute the estimated risks

4- **if** $\min\{\text{Plug-in}_k^{df}\} < \min\{\text{Plug-in}_k^{bs}\}$

$\widehat{k} = \underset{k}{\text{argmin}}\ \text{Plug-in}_k^{df} \Leftarrow$ Diffusion optimal iteration number

$\widehat{\mathbf{z}} = \mathbf{V}\mathbf{S}^{\widehat{k}}\mathbf{V}^T\mathbf{y} \Leftarrow$ Diffusion patch denoising

5- **else**

$\widehat{k} = \underset{k}{\text{argmin}}\ \text{Plug-in}_k^{bs} \Leftarrow$ Boosting optimal iteration number

$\widehat{\mathbf{z}} = \mathbf{V}\left(\mathbf{I} - (\mathbf{I} - \mathbf{S})^{\widehat{k}+1}\right)\mathbf{V}^T\mathbf{y} \Leftarrow$ Boosting patch denoising

end

$$\text{Diffusion Plug-in Risk Estimator :} \quad \text{Plug-in}_k^{df} = \sum_{i=1}^{n}(1-\lambda_i^k)^2\widetilde{b}_i^2 + \sigma^2\lambda_i^{2k} \tag{2.13}$$

$$\text{Boosting Plug-in Risk Estimator :} \quad \text{Plug-in}_k^{bs} = \sum_{i=1}^{n}(1-\lambda_i)^{2k+2}\widetilde{b}_i^2 + \sigma^2\left(1-(1-\lambda_i)^{k+1}\right)^2 \tag{2.14}$$

In each patch, minimum values of Plug-in_k^{df} and Plug-in_k^{bs} as a function of k are computed and compared, and the iteration type with the least risk is chosen. It is worth mentioning that since the optimal iteration number $\widehat{k}$ can be any real positive value, in the implementation of the diffusion filter, $\mathbf{W}^{\widehat{k}}$ is replaced by $\mathbf{V}\mathbf{S}^{\widehat{k}}\mathbf{V}^T\mathbf{y}$ in which $\mathbf{S}^{\widehat{k}} = \text{diag}[\lambda_1^{\widehat{k}}, ..., \lambda_n^{\widehat{k}}]$. This has been similarly shown for the boosting filter in Algorithm 1. Next, for the sake of comparison, the SURE estimator is discussed.

2.3.2 SURE

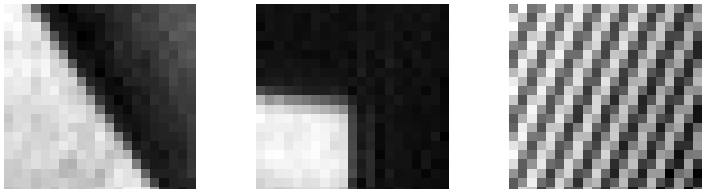
Denoting $F(\mathbf{y})$ as an estimate of the latent signal $\mathbf{z}$, the SURE estimator for MSE is defined as:

$$\text{SURE}(\mathbf{y}) = \|\mathbf{y} - F(\mathbf{y})\|^2 + 2\sigma^2\text{div}(F(\mathbf{y})) - n\sigma^2, \tag{2.15}$$

where $\text{div}(F(\mathbf{y})) \equiv \sum_i \frac{\partial F_i(\mathbf{y})}{\partial y_i}$. Under the additive Gaussian noise assumption, this random variable is an unbiased estimate of the MSE. In our context, $F(\mathbf{y})$ is replaced by $\mathbf{W}^k\mathbf{y}$ which can be approximately treated as a linear filtering framework (see Appendix 5.A). With this linear approximation we have: $\text{div}(F(\mathbf{y})) \approx \text{tr}(\mathbf{W}^k)$, and then the SURE estimator for the diffusion process can be expressed as:

$$\text{SURE}_k^{df} = \|(\mathbf{I} - \mathbf{W}^k)\mathbf{y}\|^2 + 2\sigma^2\text{tr}(\mathbf{W}^k) - n\sigma^2 \tag{2.16}$$

Considering the eigen-decomposition of the filter, replacing $\mathbf{y}$ with $\mathbf{V}\check{\mathbf{b}}$ (where $\check{\mathbf{b}}$ is the energy distribution of the *noisy* signal over the eigenvectors; $\check{\mathbf{b}} = \mathbf{V}^T\mathbf{y}$) after some



(a) (b) (c)

Figure 2.3: Clean patches from *Barbara*: (a) Edge, (b) Corner, (c) Texture.

simplifications, we have

$$\begin{aligned} \mathrm{SURE}_k^{df} &= (\mathbf{V}\check{\mathbf{b}})^T\mathbf{V}(\mathbf{I}-\mathbf{S}^k)^2\mathbf{V}^T(\mathbf{V}\check{\mathbf{b}}) + 2\sigma^2\mathrm{tr}(\mathbf{W}^k) - n\sigma^2 \\ &= \sum_{i=1}^{n}(1-\lambda_i^k)^2\check{b}_i^2 + 2\sigma^2\lambda_i^k - \sigma^2 \end{aligned} \tag{2.17}$$

It is easy to show that the expected value of SURE_k^{df} will replace $\check{b}_i^2$ with $b_i^2+\sigma^2$, which after simplification indeed yields (2.5). Replacing $F(\mathbf{y})$ with $\widehat{\mathbf{z}}_k = \left(\mathbf{I}-(\mathbf{I}-\mathbf{W})^{k+1}\right)\mathbf{y}$ for boosting filtering, we can get the corresponding SURE as:

$$\mathrm{SURE}_k^{bs} = \|(\mathbf{I}-\mathbf{W})^{k+1}\mathbf{y}\|^2 + 2\sigma^2\mathrm{tr}(\mathbf{I}-(\mathbf{I}-\mathbf{W})^{k+1}) - n\sigma^2 \tag{2.18}$$

Doing the same simplifications as before and replacing $\mathbf{y}$ with $\mathbf{V}\check{\mathbf{b}}$, we get

$$\begin{aligned} \mathrm{SURE}_k^{bs} &= (\mathbf{V}\check{\mathbf{b}})^T\mathbf{V}(\mathbf{I}-\mathbf{S})^{2k+2}\mathbf{V}^T(\mathbf{V}\check{\mathbf{b}}) + 2\sigma^2\mathrm{tr}(\mathbf{I}-(\mathbf{I}-\mathbf{W})^{k+1}) - n\sigma^2 \\ &= \sum_{i=1}^{n}(1-\lambda_i)^{2k+2}\check{b}_i^2 + 2\sigma^2(1-(1-\lambda_i)^{2k+2}) - \sigma^2 \end{aligned} \tag{2.19}$$

Again, we can show that the expected value of SURE_k^{bs} yields (2.9). This SURE estimator also has a similar algorithmic description to the plug-in represented in Algorithm 1.

2.4 Selection of the Best Iteration Method and Iteration Number

While SURE estimator is widely used for the task of risk estimation, our introduced plug-in estimator is superior in many cases. Figs. 2.4, 2.5 and 2.6 illustrate MSE denoising curves of the three patches shown in Fig. 2.3, perturbed[2] by AWGN with $\sigma = 25$. Three denoiser kernels (Bilateral [1], NLM [2] and LARK [3]) are used in our experiments, true MSE values of diffusion and twicing methods and corresponding estimated risks are compared for each patch. As can be seen, the plug-in outperforms the SURE estimator in most instances, certainly insofar as the shape of the curves are concerned. [3]

While this experiment anecdotally shows that the plug-in is superior to the SURE estimator, a theoretical analysis is much more convincing. We detailed this analysis in Appendix 6.A, and summarize it below. Our study shows that two important factors affect performance of the plug-in estimator; namely the pre-filter, and the baseline kernel type. Intuitively, the major advantage of the plug-in is use of the local SNR, which is estimated by the pre-filter. However, the SURE estimator is only a function of the kernel type (eigenvalues of the kernel $\{\lambda_i\}$).

The accuracy of these two risk estimators is analyzed in Appendix 6.A. With Gaussian assumption for the noise in the pre-filtered patch $\widetilde{\mathbf{z}} = \mathbf{z} + \boldsymbol{\eta}$ where $\boldsymbol{\eta} = \mathbf{N}(\mathbf{0}, \nu^2\mathbf{I})$, and defining $\beta = \frac{\nu^2}{\sigma^2}$ as relative variance of the noise in the pre-filtered

[2] Averaging over 50 noise realizations in a Monte-Carlo simulation.

[3] We also compared the SURE estimator with the Monte-Carlo SURE in [27], but this did not yield better results. As Ramani et al discuss [27], their Monte-Carlo SURE has a sufficiently low variance estimate when $F(\mathbf{y})$ mostly performs "local" operations (or equivalently, $\mathbf{W}(\mathbf{y})$ is quite sparse in $F(\mathbf{y}) = \mathbf{W}(\mathbf{y})\mathbf{y}$). Yet, when $\mathbf{y}$ is an image patch, $F(\mathbf{y})$ is not a local operator (or equivalently, $\mathbf{W}(\mathbf{y})$ is not "nearly" diagonal). As a result, variance of the Monte-Carlo SURE estimator will be large for the purpose of patch-based risk estimation.

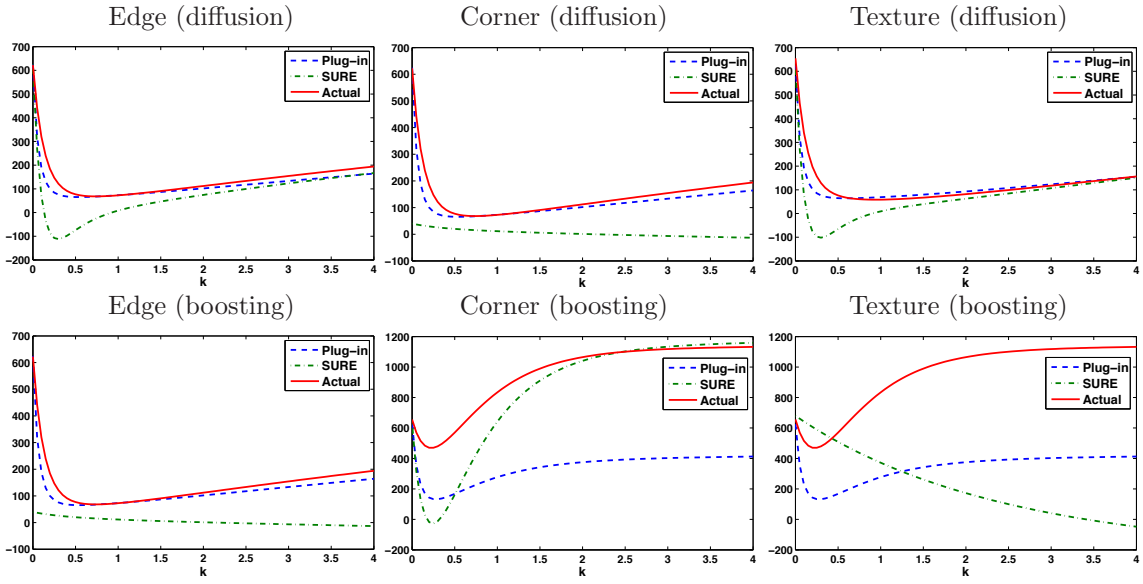


Figure 2.4: MSE of the three patches using **Bilateral kernel** [1] with diffusion/boosting iterations and plug-in/SURE estimators.

and noisy signal, we prove that the plug-in estimator outperforms SURE when in each channel i:

$$\mathrm{snr}_i \geq \frac{\beta^2(n+2)-2}{4(1-\beta)}, \tag{2.20}$$

where $\mathrm{snr}_i = \frac{b_i^2}{\sigma^2}$ is the signal-to-noise ratio of channel i. For the plug-in estimator to be superior to SURE, regardless of snr_i, the relative variance must be $\beta \leq \sqrt{\frac{2}{n+2}}$. This implies that for a typical patch size (say $n = 11 \times 11$) and a moderately effective pre-filter ($\beta \leq 0.13$), the plug-in estimator is consistently better than SURE.

Another key factor in the comparison of the two estimators is their relative sensitivity to the type of kernel. In Appendix 6.B we study the effect of kernel type, specifically the filter eigenvalues. This analysis shows that when $\lambda_i \to 0$, as long as (2.20) holds, the plug-in is less sensitive than SURE. NLM and Bilateral are examples of these "aggressive" kernels with many eigenvalues close to 0. On the other hand, our

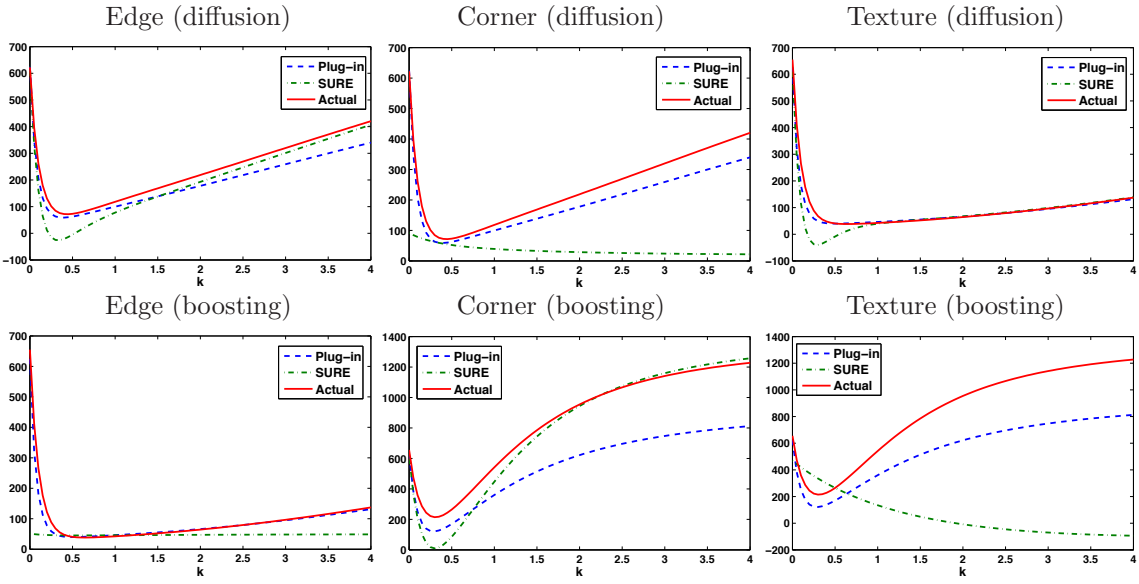


Figure 2.5: MSE of the three patches using **NLM kernel** [2] with diffusion/boosting iterations and plug-in/SURE estimators.

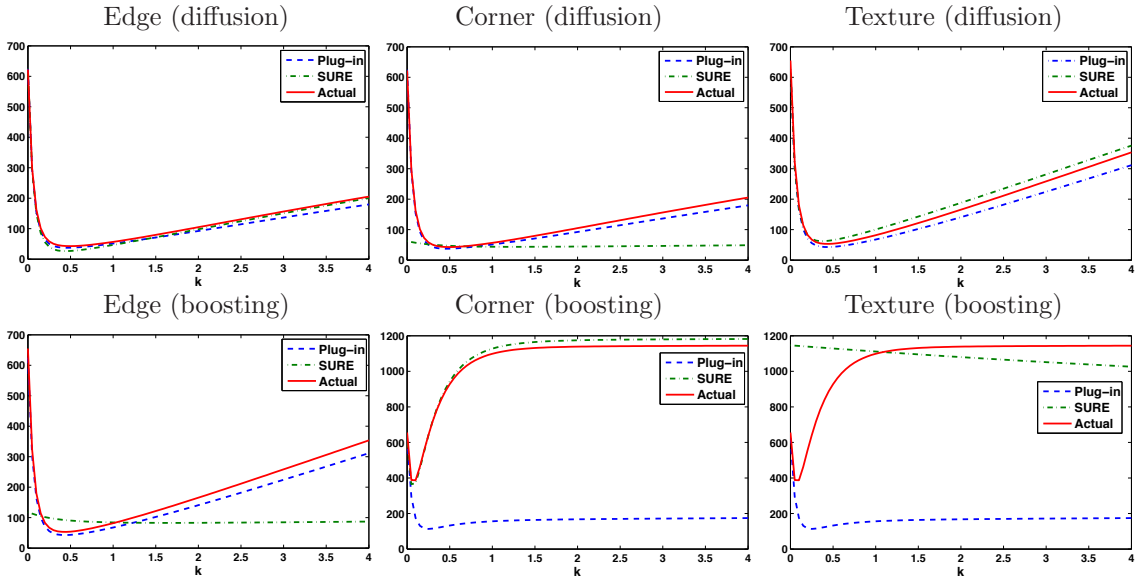


Figure 2.6: MSE of the three patches using **LARK kernel** [3] with diffusion/boosting iterations and plug-in/SURE estimators.

sensitivity analysis also implies that for filters with typically larger eigenvalues $\{\lambda_i\}$, the plug-in estimator is more sensitive.

These results motivate replacing SURE with the plug-in risk estimator for many estimation instances. In our experiments (Section 2.6) on the whole image we compare performance of the two estimators for each base kernel and confirm our claims. But before moving on to the experiments, we explain our aggregation strategy in the next section.

2.5 Aggregating Overlapping Patches

So far, we have found the best estimate for the iteration number in each patch. Our optimized per-patch filtering can be expressed as choosing between one of these two iterations:

$$\text{Diffusion :} \qquad \widehat{\mathbf{z}}_j = \mathbf{W}^{\widehat{k}_j}\mathbf{y}_j, \tag{2.21}$$

$$\text{Boosting :} \qquad \widehat{\mathbf{z}}_j = \left(\mathbf{I} - (\mathbf{I}-\mathbf{W})^{\widehat{k}_j+1}\right)\mathbf{y}_j, \tag{2.22}$$

in which $\mathbf{y}_j$ and $\widehat{\mathbf{z}}_j$ are the *j-th* noisy and denoised patch, respectively, and $\widehat{k}_j$ denotes the estimated ideal stopping time for this patch. To simplify the notation, let us denote the two filters in $\mathbf{W}^{\widehat{k}_j}$ and $\left(\mathbf{I} - (\mathbf{I}-\mathbf{W})^{\widehat{k}_j+1}\right)$ as $\widehat{\mathbf{W}}_j$. In other words, each patch is estimated as $\widehat{\mathbf{z}}_j = \widehat{\mathbf{W}}_j\mathbf{y}_j$, where $\widehat{\mathbf{W}}_j$ can be computed from either diffusion or boosting process. As a result of the overlapped patches, multiple estimates are obtained for each pixel. We need to aggregate all of these estimates to compute the final estimate for each pixel. What we have is a vector of estimates of the pixel z_l which need to be aggregated to form the final denoised pixel $\widehat{z}_l$. Fig. 2.7 illustrates an example of three overlapping patches and the computed estimates in each of them. The weighted

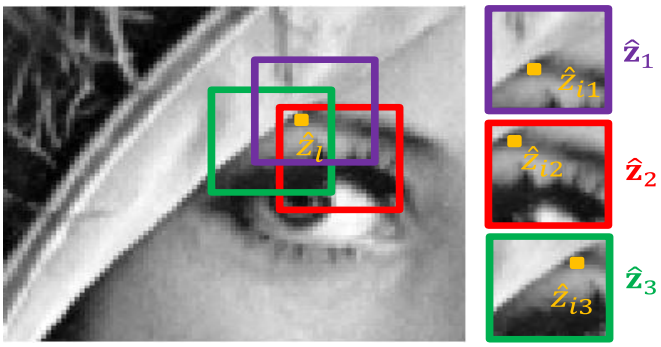


Figure 2.7: Overlapping patches give multiple estimates for each pixel. Example of three overlapping patches $\widehat{\mathbf{z}}_1$, $\widehat{\mathbf{z}}_2$ and $\widehat{\mathbf{z}}_3$ give three estimates $\widehat{z}_{i1}$, $\widehat{z}_{i2}$ and $\widehat{z}_{i3}$ for computing the final denoised pixel $\widehat{z}_i$.

averaging can improve the aggregation especially when the weights are estimated based on the risk associated with each estimate. Considering that the plug-in and SURE assign, respectively, biased and unbiased risk estimates to each pixel, we discuss two aggregation strategies which best fit each estimator. In our framework, a variance-based aggregation is employed for SURE and an exponentially weighted averaging is used for the plug-in estimator.

2.5.1 Variance-basd Aggregation

A possible weighted average is the LMMSE (linear minimum mean-squared-error) scheme that takes into account the relative confidence in each estimate as measured by the inverse of the estimator error variance. More explicitly, the error covariance of our proposed estimator is approximated as:

$$\mathbf{C}_e = \mathrm{cov}(\widehat{\mathbf{z}}_j - \mathbf{z}_j) = \mathrm{cov}(\widehat{\mathbf{W}}_j \mathbf{e}_j) = \sigma^2 \widehat{\mathbf{W}}_j^2 \tag{2.23}$$

We denote $\widehat{z}_{ij}$ as the denoised estimate for the i-th pixel in the j-th patch $\mathbf{z}_j$. Then, the variance of the error associated with the i-th pixel estimate in the j-th patch, v_{ij} ,

is given by the i-th diagonal element of $\mathbf{C}_e$. Inverse of the estimator error variances v_{ij}, are the weights we use for the aggregation:

$$\widehat{z}_l = \sum_{j=1}^{M} \frac{\frac{\widehat{z}_{ij}}{v_{ij}}}{\sum_{j=1}^{M} \frac{1}{v_{ij}}}, \tag{2.24}$$

where M is the number of computed estimates for the l-th pixel. This approach is adequate for the case when the estimated risk is unbiased (as in SURE). When using the plug-in estimator of risk, we must take bias into account with an exponential aggregator, described next.

2.5.2 MSE-basd Aggregation

A way to take the bias into account is to consider the overall MSE rather than the variance. This has been studied in [33] and [34] where the exponentially weighted aggregation is introduced. The estimated risk associated with the *i-th* estimate in the *j-th* patch, r_{ij}, is computed by the plug-in estimator in (2.14) and (26). Thus, the l-th pixel has the following estimate:

$$\widehat{z}_l = \sum_{j=1}^{M} \frac{\widehat{z}_{ij} \exp(-r_{ij})}{\sum_{j=1}^{M} \exp(-r_{ij})}, \tag{2.25}$$

where the confidence coefficients $\{\exp(-r_{ij})\}$, are the weights we use for the aggregation.

2.6 Results and Comparisons

In this section we evaluate performance of the proposed method for denoising various images. Since our method is motivated to improve performance of any kernel-based denoising, we first compare our results with ones from the three standard kernels:

Table 2.1: PSNR values for the application of **Bilateral kernel** [1] with fixed parameters for each noise realization (1st column); SAIF with SURE estimator (2nd column), and SAIF with the plug-in risk estimator (3rd column)

σ	**Peppers** (512 × 512)			**Lena** (512 × 512)			**Cameraman** (256 × 256)			**Man** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	37.12	37.19	**37.24**	37.31	37.48	**37.63**	37.30	37.35	**37.63**	36.01	36.33	**36.36**
15	31.64	31.18	**32.57**	31.34	31.10	**32.48**	30.18	30.35	**30.87**	29.61	29.82	**30.29**
25	28.69	27.87	**30.44**	28.48	27.78	**30.29**	27.03	26.66	**28.14**	26.97	26.81	**27.94**
σ	**Boat** (512 × 512)			**Stream** (512 × 512)			**Parrot** (256 × 256)			**Mandrill** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	35.95	**36.24**	36.22	34.83	35.05	**35.06**	36.83	36.93	**37.1**	34.48	34.50	**34.57**
15	29.88	29.81	**30.60**	27.79	28.16	**28.21**	29.85	29.64	**30.52**	26.99	27.34	**27.33**
25	27.16	26.62	**28.31**	25.32	25.26	**25.83**	26.86	26.30	**27.84**	24.20	24.30	**25.57**

Table 2.2: PSNR values for the application of **NLM kernel** [2] with fixed parameters for each noise realization (1st column); SAIF with SURE estimator (2nd column), and SAIF with the plug-in risk estimator (3rd column)

σ	**Peppers** (512 × 512)			**Lena** (512 × 512)			**Cameraman** (256 × 256)			**Man** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	37.34	**37.62**	37.48	38.02	38.42	**38.45**	37.75	37.98	**38.06**	36.73	37.01	**37.07**
15	31.94	33.14	**33.34**	31.93	33.12	**33.39**	30.86	31.62	**31.73**	29.96	30.66	**30.82**
25	29.10	30.96	**31.29**	28.91	30.55	**30.94**	27.84	28.94	**28.98**	27.05	28.02	**28.19**
σ	**Boat** (512 × 512)			**Stream** (512 × 512)			**Parrot** (256 × 256)			**Mandrill** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	36.62	**36.90**	36.89	35.36	35.42	**35.55**	37.18	37.58	**37.66**	34.92	34.95	**35.05**
15	30.38	30.83	**31.17**	28.06	28.53	**28.58**	30.51	31.24	**31.32**	27.64	27.96	**28.03**
25	27.43	28.10	**28.58**	25.24	25.83	**25.88**	27.71	28.67	**28.87**	24.55	24.96	**25.13**

LARK, NLM and Bilateral. We also test stability of SAIF when an arbitrary tuning parameter is used. In all cases the proposed estimators show a promising improvement over the standard kernels. We will show that the resulting SAIF-ly improved filters are comparable, in terms of MSE (PSNR) and SSIM [10], to state-of-the-art denoising methods, and in many cases visually superior.

In our first simulations the patch size is set as 11×11 and in a Monte-Carlo simulation, 10 independent noise realizations were used. We varied k from 0 to 6 with

Peppers Lena Cameraman Man Boat Mandrill

Figure 2.8: Some benchmark images used to evaluate performance of denoising methods.

Table 2.3: PSNR values for the application of the **LARK kernel** [3] with fixed parameters for each noise realization (1st column); SAIF with SURE estimator (2nd column), and SAIF with the plug-in risk estimator (3rd column)

σ	**Peppers** (512 × 512)			**Lena** (512 × 512)			**Cameraman** (256 × 256)			**Man** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	36.18	**37.20**	36.89	36.49	**38.39**	37.75	36.31	**37.92**	37.37	35.66	**37.16**	36.78
15	32.08	**33.33**	33.13	32.41	**33.75**	33.52	30.34	**31.41**	30.98	30.60	**31.17**	30.95
25	30.04	**31.38**	31.32	30.12	**31.40**	31.34	27.95	**28.69**	28.19	27.95	**28.54**	28.28
σ	**Boat** (512 × 512)			**Stream** (512 × 512)			**Parrot** (256 × 256)			**Mandrill** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
5	35.72	**36.57**	36.34	34.90	**35.66**	35.51	36.02	**37.73**	37.22	34.69	**35.16**	35.10
15	31.03	**31.63**	31.54	28.29	**28.76**	28.20	30.35	**31.23**	30.92	27.20	**28.05**	27.38
25	28.43	**29.12**	28.98	25.69	**26.18**	25.59	27.58	**28.62**	28.26	24.01	**25.12**	24.02

0.05 as the step size. As an initial guess for the smoothing parameters in the kernels, we set $h_x = 2\sqrt{2}$ and $h_y = 20\sqrt{2}\sigma$ in the Bilateral kernel, $h_y = 0.43\sigma$ in the NLM filter and the smoothing parameter in LARK [3] is fixed as 0.25σ.

Fig. 2.8 shows some benchmark images we used. Tables 2.1, 2.2 and 2.3 show PSNR results of the standard kernel (fixed parameters in Bilateral, NLM, or LARK), SURE and the plug-in estimators. It can be seen that for Bilateral and Non-local means kernels, the plug-in estimator shows consistent improvement over both the standard estimate using the kernel, and the optimally iterated kernel from SURE. For the LARK kernel, the SURE method outperforms the plug-in estimator. Apparently, this performance difference occurs most noticeably for highly textured images such as Mandrill.

In Fig. 2.9 the two types of iterations, diffusion and boosting, are compared for the three kernels. The iteration map identifies the type and number of applied iterations on the patches of the image. The map colorbar represents positive and negative values for diffusion and boosting iterations, respectively. As can be seen, while diffusion recovers most of the flat and smooth patches, boosting takes care of the texture and more complex ones. It is worth noting that applying overlapped patches also has the advantage of computing multiple estimates for each pixel from the *both* iteration

types. In other words, in the aggregation process, some pixels may be from diffusion in one patch, whereas others may be from boosting in another overlapping patch. This is especially useful for pixels at the border of smooth and texture regions.

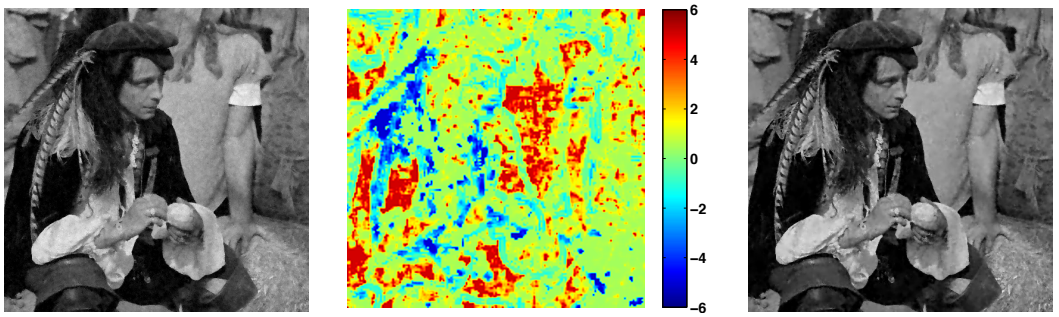
The effect of the parameter tuning is studied in Table 2.4 . In this set of simulations, NLM kernel weights with different control parameter, h_y, were computed for each image and then fed to the plug-in estimator. As can be seen, across a large range of h_y, performance of the proposed estimators is quite stable and shows consistent improvement over the baseline kernel. From these results, we can see that it is possible to improve the performance of the standard kernel with an arbitrary starting parameter by employing the proper number and type of iteration with the proposed MSE estimator.

Performances of the proposed SAIF algorithm and other methods are quantified across different noise levels in Table 2.5, which shows that the proposed method is quantitatively quite comparable to LPG-PCA [4] and BM3D [5].

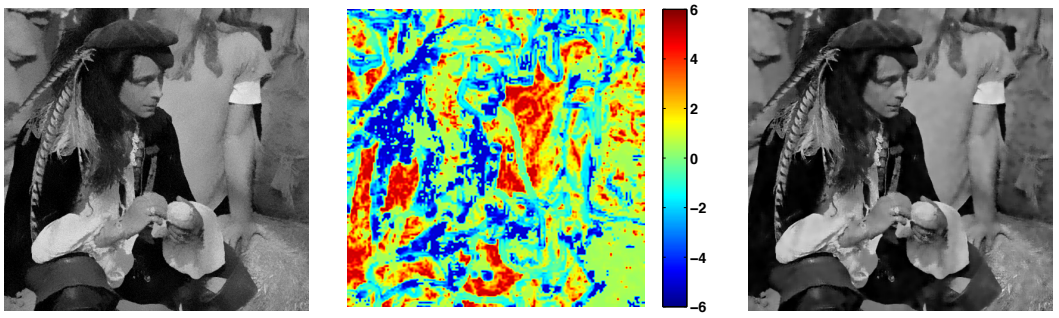
Fig. 2.10 demonstrates the denoising results of Parrot image obtained by different methods compared to the proposed method using the plug-in estimator. In addition to about 1 dB improvement over the baseline NLM in terms of PSNR, visual quality of the proposed method also is comparable and even superior to the state of the art denoising. In this case SAIF appears to recover some texture regions which were over-smoothed by BM3D and LPG-PCA. Fig. 2.11 shows performance of the SURE estimator for the Stream image. As can be seen, result by SAIF is visually close to BM3D and LPG-PCA. We note that our Matlab code and additional results are available at the project website[4].

In terms of computational complexity, denoising a 256×256 grayscale image

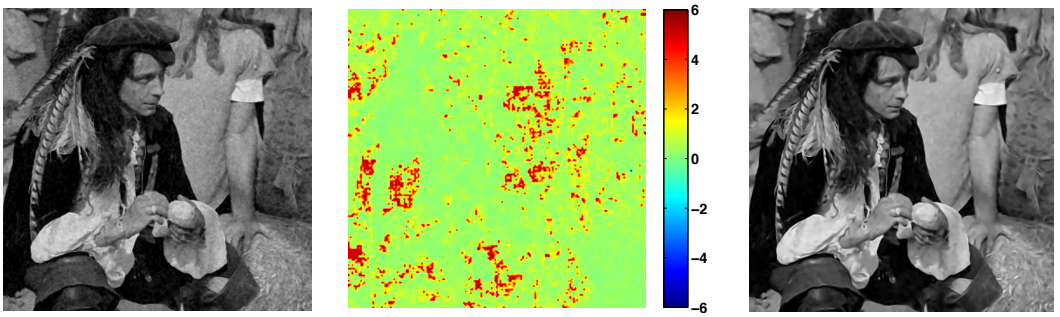
[4] http://www.soe.ucsc.edu/~htalebi/SAIF.php.



(a) Bilateral, PSNR=26.97 dB (b) Bilateral iteration map (c) SAIF (Bilateral), PSNR=27.92 dB



(d) NLM, PSNR=27.05 dB (e) NLM iteration map (f) SAIF (NLM), PSNR=28.19 dB



(g) LARK, PSNR=27.95 dB (h) LARK iteration map (i) SAIF (LARK), PSNR=28.54 dB

Figure 2.9: Denoising example for the plug-in and SURE estimators with different kernels. AWGN with $\sigma = 25$ is added to Man image: (a) Bilateral kernel, (b) Iteration map for the Bilateral kernel (the colorbar depicts positive iteration numbers for diffusion and negative ones for boosting), (c) The plug-in estimator for Bilateral kernel, (d) NLM kernel, (e) Iteration map for the NLM kernel, (f) The plug-in estimator for NLM kernel, (g) LARK kernel, (h) Iteration map for the NLM kernel, (i) The SURE estimator for LARK kernel.

Table 2.4: Performance of the plug-in estimator for the **NLM kernel** with different smoothing parameters under WGN corruption with $\sigma = 15$.

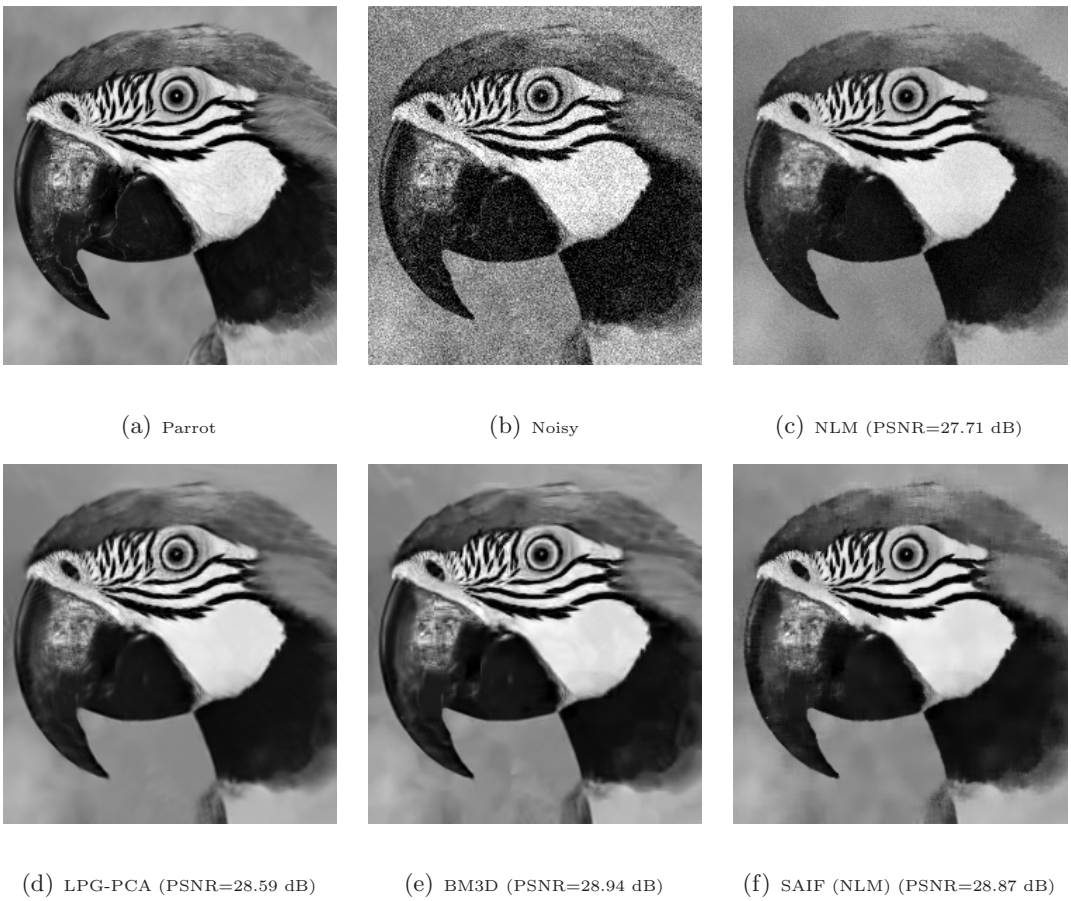
h_y	**Peppers** (512 × 512)			**Lena** (512 × 512)			**Cameraman** (256 × 256)			**Man** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
3	29.56	31.84	33.16	29.64	31.99	33.34	28.88	30.71	31.52	28.44	30.29	30.85
6	31.89	32.77	33.30	31.73	32.80	33.40	30.53	31.42	31.63	29.31	30.49	30.79
9	30.63	32.93	33.22	30.20	32.84	33.21	28.57	31.45	31.52	27.38	30.30	30.57
h_y	**Boat** (512 × 512)			**Stream** (512 × 512)			**Parrot** (256 × 256)			**Mandrill** (512 × 512)		
	Standard	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**	**Standard**	**SURE**	**Plug-in**
3	28.80	30.60	31.31	27.12	28.41	28.61	28.49	30.37	31.09	26.66	27.69	27.90
6	29.80	30.84	31.20	27.23	28.46	28.58	30.25	31.03	31.32	26.72	27.95	28.04
9	27.87	30.68	30.97	25.15	28.25	28.42	28.72	31.06	31.26	24.35	27.79	27.93

Table 2.5: Denoising performance of some popular methods (LPG-PCA [4], BM3D [5]) under WGN corruption, compared to SAIF for the LARK [3] and NLM [2] kernels. Results noted are average PSNR (top) and SSIM [10] (bottom) over 10 independent noise realizations for each σ.

σ	**Man** (512 × 512)				**Parrot** (256 × 256)			
	LPG-PCA	**BM3D**	**SAIF** (LARK)	**SAIF** (NLM)	**LPG-PCA**	**BM3D**	**SAIF** (LARK)	**SAIF** (NLM)
5	37.13	**37.33**	37.16	37.07	37.76	**37.92**	37.73	37.66
	0.951	**0.956**	0.953	0.952	0.966	**0.967**	**0.967**	0.964
15	30.84	**31.24**	31.17	30.82	31.14	**31.43**	31.23	31.32
	0.850	**0.863**	0.860	0.847	0.892	**0.896**	0.890	0.890
25	28.25	**28.81**	28.54	28.19	28.59	**28.94**	28.62	28.87
	0.772	**0.794**	0.782	0.759	0.843	**0.850**	0.839	0.847
σ	**Stream** (512 × 512)				**Mandrill** (512 × 512)			
	LPG-PCA	**BM3D**	**SAIF** (LARK)	**SAIF** (NLM)	**LPG-PCA**	**BM3D**	**SAIF** (LARK)	**SAIF** (NLM)
5	35.62	**35.75**	35.66	35.55	**35.31**	35.25	35.16	35.05
	0.963	**0.965**	**0.965**	0.963	0.957	0.958	**0.959**	0.956
15	28.53	28.74	**28.76**	28.58	28.09	**28.17**	28.05	28.03
	0.835	0.846	**0.849**	0.834	0.832	0.843	**0.844**	0.837
25	25.86	**26.21**	26.18	25.88	25.16	**25.45**	25.12	25.13
	0.720	0.739	**0.743**	0.710	0.726	**0.746**	0.737	0.724

with an unoptimized implementation of our method in Matlab take, on average, about 180 seconds. For such images, LPG-PCA (implemented by its authors for Matlab) take, on average, 280 seconds. BM3D, with its optimized implementation (implemented by the authors mostly in C and compiled as Mex for Matlab), takes significantly less time (about 1 second on average) for these images. However, our method is sped up significantly by reducing the amount of overlap between patches. For example, when estimating every fifth patch, our method requires only 120 seconds on average (including pre-filtering) with a minor drop in performance (about 0.1 dB).

Summary – We have presented a framework for improved denoising by data-

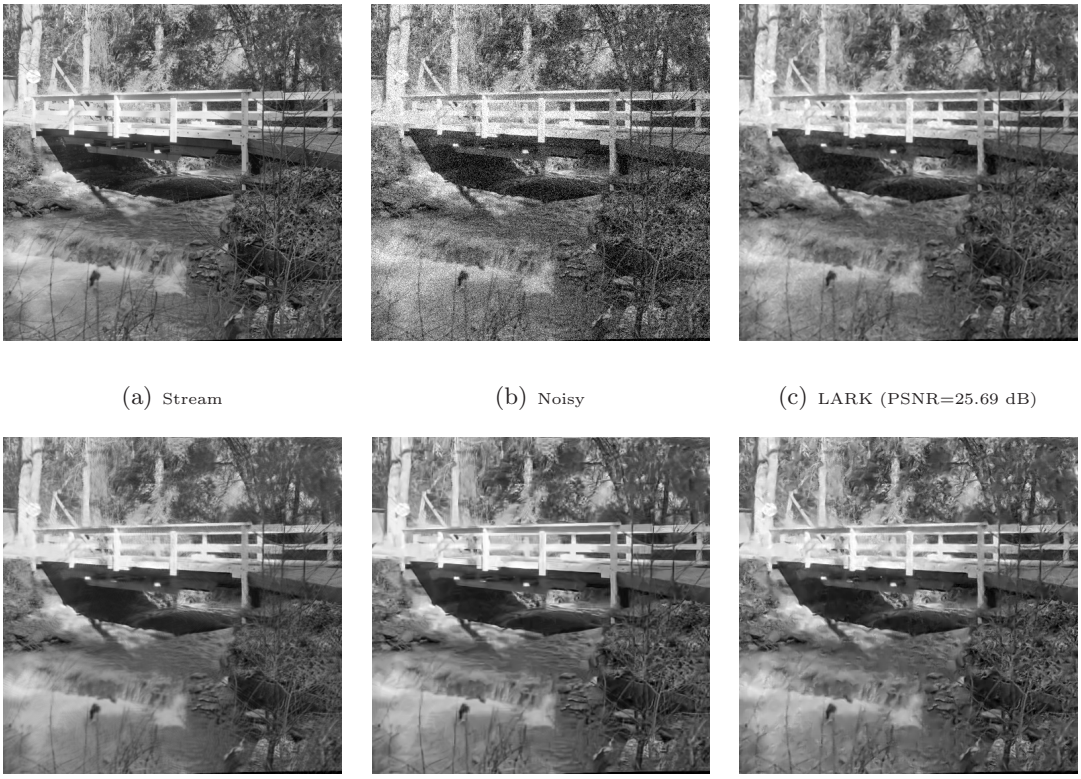


(a) Parrot (b) Noisy (c) NLM (PSNR=27.71 dB)

(d) LPG-PCA (PSNR=28.59 dB) (e) BM3D (PSNR=28.94 dB) (f) SAIF (NLM) (PSNR=28.87 dB)

Figure 2.10: Comparison of denoising performance on noisy parrot image corrupted by AWGN of $\sigma = 25$. (a) Original image, (b) Noisy input, (c) NLM [2], (d) LPG-PCA [4], (e) BM3D [5], (f) Proposed SAIF (NLM).

dependent kernels. Given any spatial domain filter, we can boost its performance to near state-of-the-art by employing optimized iteration methods. This iterative filtering is implemented patch-wise. Armed with diffusion and boosting as two *complementary* iteration techniques, each patch is filtered by the optimum local filter. More specifically, by exploiting the best iteration number and method which minimizes MSE in each patch, SAIF is capable of automatically adjusting the local smoothing strength according to local SNR. The experimental results demonstrate that the proposed approach improves



(a) Stream (b) Noisy (c) LARK (PSNR=25.69 dB)

(d) LPG-PCA (PSNR=25.86 dB) (e) BM3D (PSNR=26.21 dB) (f) SAIF (LARK) (PSNR=26.18 dB)

Figure 2.11: Comparison of denoising performance on noisy stream image corrupted by AWGN of $\sigma = 25$. (a) Original image, (b) Noisy input, (c) LARK [3], (d) LPG-PCA [4], (e) BM3D [5], (f) Proposed SAIF (LARK).

the performance of kernel based filtering methods in terms of both PSNR (MSE) and subjective visual quality. Using the estimated local SNR as empirical prior knowledge of the latent signal, we proposed the plug-in estimator which can outperform SURE estimator in many cases.

2.A Approximation of the Data-dependent Filter

By applying an effective pre-filtering, the filter weight matrix $\mathbf{W}$ is largely dependent on the *latent* image rather than the noisy input image [26]. To be more specific, consider the practical implementation of the filter as follows:

$$\widehat{\mathbf{z}} = \mathbf{W}(\widetilde{\mathbf{z}})\mathbf{y} \tag{2.26}$$

Denote the estimate of the j-th pixel as:

$$\widehat{z}_j = \mathbf{w}_j^T(\widetilde{\mathbf{z}})\mathbf{y} \tag{2.27}$$

where $\mathbf{w}_j^T$ denotes the j-th row the filter $\mathbf{W}(\widetilde{\mathbf{z}})$. Assuming that the pre-filtered image is now only corrupted by a small additive noise: $\widetilde{\mathbf{z}} = \mathbf{z} + \boldsymbol{\eta}$, we can make the following first order Taylor approximation:

$$\widehat{z}_j = \mathbf{w}_j^T(\widetilde{\mathbf{z}})\mathbf{y} \approx \mathbf{w}_j^T(\mathbf{z})\mathbf{y} + \boldsymbol{\eta}^T\mathbf{G}_j\mathbf{y} \tag{2.28}$$

where the $n \times n$ diagonal matrix $\mathbf{G}_j$ contains the vector $\nabla\mathbf{w}_j$ along its diagonal entries, in which $\nabla\mathbf{w}_j(\mathbf{z})$ denotes the gradient of the vector $\mathbf{w}_j$ with respect to its argument. The first term in the above $(\mathbf{w}_j^T(\mathbf{z})\mathbf{y})$ is the *oracle* filter. The second term is the error between the oracle and the practical filter:

$$\triangle_j = \mathbf{w}_j^T(\widetilde{\mathbf{z}})\mathbf{y} - \mathbf{w}_j^T(\mathbf{z})\mathbf{y} \approx \boldsymbol{\eta}^T\mathbf{G}_j\mathbf{y} \tag{2.29}$$

As can be seen, this error is dependent on the quality of the pre-filter (how small is $\boldsymbol{\eta}$) and also smoothness of the kernel ($\nabla\mathbf{w}_j$). We note that this gradient refers to the shape of the baseline kernel, and the way it depends on its argument (Gaussian being typical) and not on the actual underlying image. While a good choice of pre-filter can sufficiently suppress the noise, tuning the smoothness parameter of the baseline kernel

guarantees the gradient of the filter to be small. As a result, we can be assured that our approximation is reliable for the performance analysis in this chapter.

2.B Mean-Squared Error of The Plug-in and SURE Estimators

Here we derive an expression for expected error of each risk estimator and then use this to compare the plug-in and SURE estimators. We start from the expression for diffusion plug-in risk estimator:

$$\text{Plug-in}_k^{df} = \sum_{i=1}^{n}(1-\lambda_i^k)^2\widetilde{b}_i^2 + \sigma^2\lambda_i^{2k} \tag{2.30}$$

We assume that $\widetilde{b}_i = b_i + \eta_i$, where η_i is the projected noise of the pre-filtered image on the eigenvectors of the filter, where η_i are AWGN with mean zero and covariance $\nu^2\mathbf{I}$. The expected value of eq. 2.30 can be expressed as:

$$\text{E}[\text{Plug-in}_k^{df}] = \sum_{i=1}^{n}(1-\lambda_i^k)^2(b_i^2+\nu^2) + \sigma^2\lambda_i^{2k} \tag{2.31}$$

Consequently the bias of the plug-in estimator can be written as:

$$\text{bias}(\text{Plug-in}_k^{df}) = \nu^2\sum_{i=1}^{n}(1-\lambda_i^k)^2 \tag{2.32}$$

The variance term also is:

$$\text{var}(\text{Plug-in}_k^{df}) = 2\nu^2\sum_{i=1}^{n}(1-\lambda_i^k)^4(\nu^2+2b_i^2) \tag{2.33}$$

Then the MSE of the Plug-in_k^{df} estimator is as follows:

$$\text{MSE}_{\text{Plug-in}_k^{df}} = \nu^4(\sum_{i=1}^{n}(1-\lambda_i^k)^2)^2 + 2\nu^2\sum_{i=1}^{n}(1-\lambda_i^k)^4(\nu^2+2b_i^2) \tag{2.34}$$

Unsurprisingly, as ν decreases, MSE of the estimator tends to zero. We can also derive MSE of the SURE estimator in (2.17) by replacing $\check{b}_i$ with $b_i + e_i$ where $\mathbf{e} = \mathbf{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Then the variance term and the MSE are:

$$\mathrm{MSE}_{\mathrm{SURE}_k^{df}} = \mathrm{var}(\mathrm{SURE}_k^{df}) = 2\sigma^2 \sum_{i=1}^{n} (1-\lambda_i^k)^4 (\sigma^2 + 2b_i^2) \tag{2.35}$$

Comparison of (2.34) and (2.35) can determine the better risk estimator. To accomplish this comparison, we aim to define an upper bound for the error incurred by the plug-in estimator risk. Applying the Cauchy-Schwartz inequality to the squared bias term in (2.34):

$$\nu^4 (\sum_{i=1}^{n} (1-\lambda_i^k)^2)^2 \leq n\nu^4 \sum_{i=1}^{n} (1-\lambda_i^k)^4 \tag{2.36}$$

Then we have:

$$\mathrm{MSE}_{\mathrm{Plug\text{-}in}_k^{df}} \leq \sum_{i=1}^{n} (1-\lambda_i^k)^4 \left((n+2)\nu^4 + 4\nu^2 b_i^2 \right) \tag{2.37}$$

Defining $\beta = \frac{\nu^2}{\sigma^2}$ as relative variance of the noise in the pre-filtered and noisy signal, comparison of (2.35) and (2.37) shows that the plug-in estimator outperforms SURE when in each channel i:

$$\mathrm{snr}_i \geq \frac{\beta^2(n+2) - 2}{4(1-\beta)}, \tag{2.38}$$

where $\mathrm{snr}_i = \frac{b_i^2}{\sigma^2}$ is the signal-to-noise ratio of channel i. Similarly for the boosting iteration the MSE of the plug-in estimator is:

$$\mathrm{MSE}_{\mathrm{Plug\text{-}in}_k^{bs}} = \nu^4 (\sum_{i=1}^{n} (1-\lambda_i)^{2k+2})^2 + 2\nu^2 \sum_{i=1}^{n} (1-\lambda_i)^{4k+4} (\nu^2 + 2b_i^2) \tag{2.39}$$

and also for the boosting SURE estimator in (2.9) we have:

$$\mathrm{MSE}_{\mathrm{SURE}_k^{bs}} = \mathrm{var}(\mathrm{SURE}_k^{bs}) = 2\sigma^2 \sum_{i=1}^{n} (1-\lambda_i)^{4k+4} (\sigma^2 + 2b_i^2) \tag{2.40}$$

Doing the same analysis as we did for the diffusion iteration, the given constraint in (2.38) is obtained again for the boosting iteration.

2.C Sensitivity of The Plug-in and SURE Estimators

What we study here is the sensitivity of each estimator to the baseline kernel type. Assuming the MSE expressions as functions of the filter eigenvalues $\{\lambda_i\}$, their derivatives can explain the sensitivity of the risk to the filter (kernel) type. Defining $\varepsilon_i = (1-\lambda_i^k)^2$, we have

$$\frac{\partial \text{MSE}_{\text{Plug-in}_k^{df}}}{\partial \varepsilon_j} = 2\nu^4 \sum_{i=1}^{n} \varepsilon_i + (4\nu^4 + 8\nu^2 b_j^2)\varepsilon_j \tag{2.41}$$

and also for the SURE risk estimator given by (2.35) the derivative is

$$\frac{\partial \text{MSE}_{\text{SURE}_k^{df}}}{\partial \varepsilon_j} = (4\sigma^4 + 8\sigma^2 b_j^2)\varepsilon_j \tag{2.42}$$

As $\varepsilon_j \to 1$, we can see that the plug-in sensitivity is bounded as:

$$\lim_{\varepsilon_j \to 1} \frac{\partial \text{MSE}_{\text{Plug-in}_k^{df}}}{\partial \varepsilon_j} \leq 2(n+2)\nu^4 + 8\nu^2 b_j^2 \tag{2.43}$$

and also for the SURE estimator we have:

$$\lim_{\varepsilon_j \to 1} \frac{\partial \text{MSE}_{\text{SURE}_k^{df}}}{\partial \varepsilon_j} = (4\sigma^4 + 8\sigma^2 b_j^2) \tag{2.44}$$

Comparison of (2.43) and (2.44) shows that when $\lambda_i \to 0$, as long as (2.38) holds, the plug-in estimator is less sensitive than SURE.

On the other hand as $\varepsilon_j \to 0$ the SURE estimator's derivative tends to 0 and yet the sensitivity of the plug-in method remains dependent on the other eigenvalues of the filter:

$$\lim_{\varepsilon_j \to 0} \frac{\partial \text{MSE}_{\text{Plug-in}_k^{df}}}{\partial \varepsilon_j} = 2\nu^4 \sum_{i=1}^{n} \varepsilon_i \tag{2.45}$$

This, in particular shows that the SURE estimator is more reliable when the base kernel has eigenvalues $\{\lambda_i\}$ closer to one. The LARK filter [3] is an example of this type of

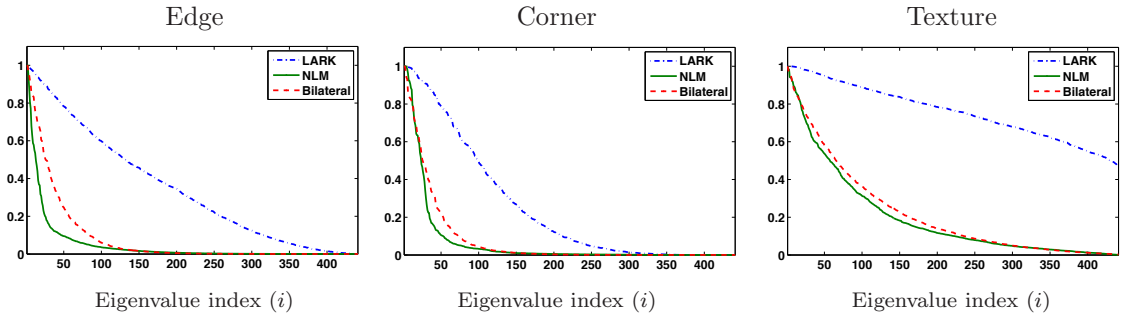


Figure 2.12: Spectrum of filters computed from the patches in Fig. 2.3. Among the three kernels, LARK eigenvalues are larger than NLM and Bilateral.

kernels with less aggressive spectrum than NLM [2] and Bilateral [1]. Fig. 2.12 compares spectrum of the three kernels for the selected patches in Fig. 2.3. As can be seen, the LARK spectrum has eigenvalues that are larger than the ones from NLM and Bilateral kernels for all the tested patches. Overall, we can conclude that the plug-in estimator better fits aggressive kernel bases like NLM and Bilateral.

Sensitivity analysis and results of the boosting iteration are similar to the presented diffusion process. The only difference is that the derivation variable ε_i should be defined as $(1-\lambda_i)^{2k+2}$.

Chapter 3

Global Filter

Abstract – This chapter introduces an innovative global filter, which takes into account all informative parts of an image. The complexity of computing the global filter is effectively reduced by approximating the leading eigen functions of the low-rank affinity matrix. This approximation is efficiently implemented using the Nyström extension that works based on sampling image pixels.

3.1 Introduction

Patch-based filtering is founded on the assumption that the latent image has a locally sparse representation in some transform domain. Wavelet and DCT in [5], principal component analysis (PCA) in [4], and over-complete dictionaries in [18] are the frequently used transforms. The filtering process is defined as applying a shrinkage function to the transform coefficients and recovering the estimated patches by inverse transform. However, performance of these patch-based methods is strictly dependent on how well the similar patches are matched [20]. Specifically, for images that are well

represented by locally sparse transform (i.e. images with locally repetitive structure such as House in Fig. 3.3), the shrinkage operator keeps most of the basis elements belonging to the latent signal and effectively removes the noise components. Yet, when the similar patches are not easily representable in a sparse way (i.e. images with locally non-repetitive, or semistochastic structures such as Mandrill in Fig. 3.3), the signal components and the noise elements can be mistakenly shrunk together. Consequently, performance of the patch-based filtering will be affected by the lack of locally (in the nearest neighbor sense) similar patches [35].

As shown in Chapter 1, a spatial domain denoising process has a transform domain filtering interpretation, where the orthogonal basis elements and the shrinkage coefficients are respectively the eigenvectors and eigenvalues of a symmetric, positive definite (data-dependent) filter matrix. For filters such as NLM and LARK the eigenvectors corresponding to the dominant eigenvalues could well represent latent image contents. Based on this idea, the SAIF filter capable of controlling the denoising strength locally for any given spatial domain method was discussed in Chapter 2. SAIF iteratively filters local image patches, and the iteration method and iteration number are automatically optimized with respect to locally estimated MSE. Although this algorithm does not set any theoretical limitation over this local window size, computational burden of building a matrix filter for a window as large as the whole image is prohibitively high.

As shown by Williams and Seeger [36], the Nyström method [15] gives a practical solution when working with huge affinity (similarity) matrices by operating on only a small portion of the complete matrix to produce a low-rank approximation. The Nyström method was initially introduced as a technique for finding numerical solutions to eigen-decomposition problems in [15] and [37]. The Nyström extension has been

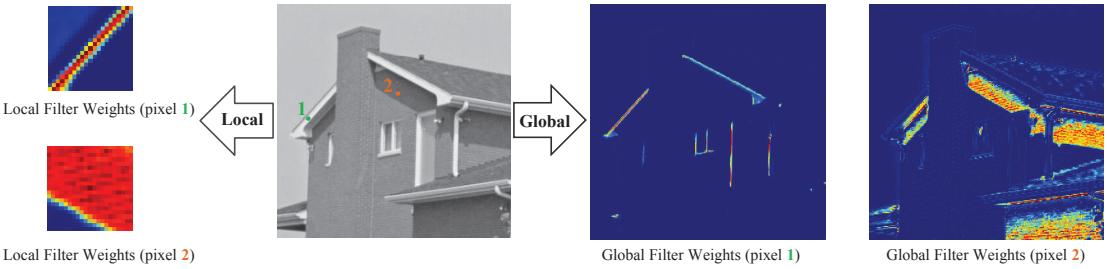


Figure 3.1: Comparison of the local and global filter weights for the NLM kernel [2]. The filter weights are computed for the two labeled pixels.

useful for different applications such as manifold learning [38], image segmentation [39], and image editing [40]. Fortunately, in our global filtering framework, the filter matrix can be closely approximated with a low-rank matrix using the Nyström method [41].

3.1.1 Local vs. Global

Fig. 3.1 illustrates local and global filter weights computed for two example pixels. As can be seen, similarity weights are not limited to the local windows and may be found across the image. Until now, our assumption was that we can compute the filter $\mathbf{W}$ for the whole image. For an image containing n pixels, this filter matrix is of size $n \times n$, which obviously demands a high computational and storage cost. In general, the computational complexity of computing and storing the filter $\mathbf{W}$ for an image of size n is $O(n^2)$. The proposed low-rank approximation of the filter matrix is explained next.

3.2 Filter Approximation

Since we only need the eigen-decomposition of this matrix, we can approximate the first p eigenvectors and eigenvalues of it without direct computation of all elements of $\mathbf{W}$. Although this idea has not been studied for the purpose of filtering before, [39] used it in the context of spectral grouping, where at first the matrix $\mathbf{K}$ is approximated by means of the Nyström method [15], and then a symmetric normalized version of this matrix is used for a data clustering scheme. Our objective is different because in the end we need to approximate the filtering matrix $\mathbf{W}$, hence we first review what is done in [39] and then adapt it to the approximation we need to effect here.

Fig. 3.2 shows our filter approximation pipeline. First, the Nyström approach for approximating the similarity (affinity) matrix $\mathbf{K}$ is used and then, the Sinkhorn method (sec. 3.2.2) is applied to estimate the eigen-decomposition of the symmetric, doubly-stochastic filter $\mathbf{W}$. Since the approximated eigenvectors are not exactly orthogonal, finally an orthogonalization procedure is employed to obtain an orthonormal approximation for eigen-decomposition of $\mathbf{W}$. These steps are shown in Algorithm 2 and we will discuss them in more details below.

3.2.1 Nyström Approximation

This method is a numerical approximation for estimating the eigenvectors of the symmetric kernel matrix $\mathbf{K}$:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Pi}\mathbf{\Phi}^T \tag{3.1}$$

where $\mathbf{\Phi} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_n]$ represents the orthonormal eigenvectors and $\mathbf{\Pi} = [\pi_1, \pi_2, ..., \pi_n]$ contains the eigenvalues of $\mathbf{K}$. Nyström [15] suggests that instead of computing all the entries of $\mathbf{K}$, we can sample our data points and estimate the leading eigenvectors of the

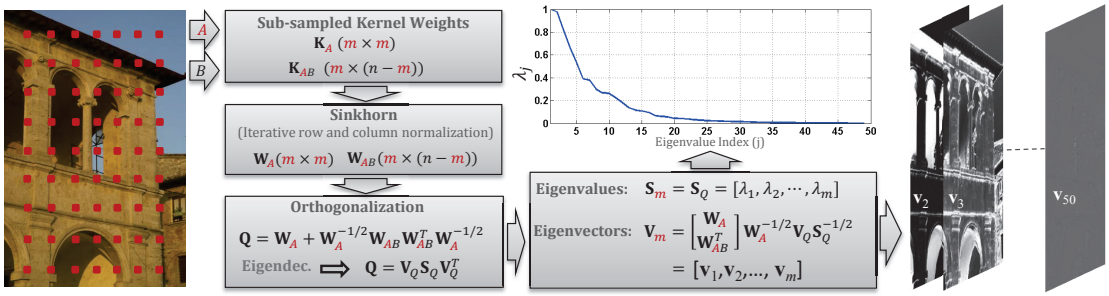


Figure 3.2: Filter approximation using Nyström extension. Set A represents m samples from input image and set B contains the rest of pixels $(n-m)$. Matrix $\mathbf{K}_A$ represents the kernel weights of the sample set A and $\mathbf{K}_{AB}$ shows the kernel weights between set A and set B. Sinkhorn algorithm approximates the filter sub-matrices $\mathbf{W}_A$ and $\mathbf{W}_{AB}$ through an iterative normalization procedure. These sub-matrices can be used to approximate m leading orthonormal eigenvectors and eigenvalues of the filter matrix. In this example m is set as 50.

matrix $\mathbf{K}$ and, as a result, an approximation $\widetilde{\mathbf{K}}$ can then be built from those estimated eigenvectors.

Having m pixels in a sampled subimage $\mathbf{A}$, we can compute the $m \times m$ kernel matrix $\mathbf{K}_A$ which represents the similarity weights of pixels in $\mathbf{A}$. We also define the subimage $\mathbf{B}$ containing the remaining $(n-m)$ pixels, followed by the $m \times (n-m)$ matrix $\mathbf{K}_{AB}$, which contains the kernel weights between pixels in $\mathbf{A}$ and $\mathbf{B}$. The similarity matrix $\mathbf{K}$ in block form is therefore:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_A & \mathbf{K}_{AB} \\ \mathbf{K}_{AB}^T & \mathbf{K}_B \end{bmatrix} \tag{3.2}$$

where $\mathbf{K}_B$ denotes the $(n-m) \times (n-m)$ similarity weights between pixels in the subimage[1] $\mathbf{B}$. As can be seen, (3.2) can be thought of as a permutation of the old $\mathbf{K}$.

[1] When $n \gg m$, $\mathbf{K}_B$ can be huge.

Algorithm 2: Spectral Approximation of the Filter $\mathbf{W}$

Input: Sub-blocks of the similarity matrix $\mathbf{K}$: $\{\mathbf{K}_A, \mathbf{K}_{AB}\}$, let

$(m, m) = \text{size}(\mathbf{K}_A)$ and $(m, n-m) = \text{size}(\mathbf{K}_{AB})$

Output: m leading orthogonal eigenvectors and eigenvalues of the

approximated filter $\mathbf{W}_m$: $\{\mathbf{V}_m, \mathbf{S}_m\}$

Nyström Approximation:

1- $\mathbf{K}_A = \mathbf{\Phi}_A \mathbf{\Pi}_A \mathbf{\Phi}_A^T$; $\Leftarrow$ Eigen-decomposition of the sub-block $\mathbf{K}_A$

2- $\widetilde{\mathbf{\Phi}} = \begin{bmatrix} \mathbf{\Phi}_A \\ \mathbf{K}_{AB}^T \mathbf{\Phi}_A \mathbf{\Pi}_A^{-1} \end{bmatrix}$; $\Leftarrow$ Approximate the m leading eigenvectors of $\mathbf{K}$

Sinkhorn:

3- $\mathbf{r} = \text{ones}(n, 1)$; $\boldsymbol{\pi}_A = \text{diag}(\mathbf{\Pi}_A)$;

4- for $i = 1 : iter$

$\mathbf{c} = 1./\left(\widetilde{\mathbf{\Phi}}(\boldsymbol{\pi}_A.(\widetilde{\mathbf{\Phi}}^T \mathbf{r}))\right)$; $\Leftarrow$ Column normalization weights

$\mathbf{r} = 1./\left(\widetilde{\mathbf{\Phi}}(\boldsymbol{\pi}_A.(\widetilde{\mathbf{\Phi}}^T \mathbf{c}))\right)$; $\Leftarrow$ Row normalization weights

end

5- for $i = 1 : m$

$\mathbf{W}_{A,AB} = \mathbf{r}(i)(\boldsymbol{\pi}_A^T.\widetilde{\mathbf{\Phi}}(i, :))(\text{repmat}(\mathbf{c}, [1, m]).\widetilde{\mathbf{\Phi}})^T$;

end

6- $\mathbf{W}_A = \mathbf{W}_{A,AB}(:, 1 : m)$; $\Leftarrow$ Sub-block of the symmetrized $\mathbf{W}_{sym}$

7- $\mathbf{W}_{AB} = \mathbf{W}_{A,AB}(:, m+1 : n)$; $\Leftarrow$ Sub-block of the symmetrized $\mathbf{W}_{sym}$

Orthogonalization:

8- $\mathbf{W}_A^{1/2} = \text{sqrtm}(\mathbf{W}_A)$;

9- $\mathbf{Q} = \mathbf{W}_A + \mathbf{W}_A^{-1/2} \mathbf{W}_{AB} \mathbf{W}_{AB}^T \mathbf{W}_A^{-1/2}$;

10- $\mathbf{Q} = \mathbf{V}_Q \mathbf{S}_Q \mathbf{V}_Q^T$; $\Leftarrow$ Eigen-decomposition of the symmetric matrix $\mathbf{Q}$

11- $\mathbf{V}_m = \begin{bmatrix} \mathbf{W}_A \\ \mathbf{W}_{AB}^T \end{bmatrix} \mathbf{W}_A^{-1/2} \mathbf{V}_Q \mathbf{S}_Q^{-1/2}$; $\Leftarrow$ Approximated orthogonal

eigenvectors

12- $\mathbf{S}_m = \mathbf{S}_Q$; $\Leftarrow$ Approximated eigenvalues

Nyström suggests the following approximation for the first m eigenvectors of $\mathbf{K}$:

$$\widetilde{\mathbf{\Phi}} = \begin{bmatrix} \mathbf{\Phi}_A \\ \mathbf{K}_{AB}^T \mathbf{\Phi}_A \mathbf{\Pi}_A^{-1} \end{bmatrix} \tag{3.3}$$

where $\mathbf{K}_A = \mathbf{\Phi}_A \mathbf{\Pi}_A \mathbf{\Phi}_A^T$. Intuitively, we can say that the first m entries of $\widetilde{\mathbf{\Phi}}$ are computed exactly, and the $(n-m)$ remaining ones are approximated by a weighted projection of $\mathbf{K}_{AB}$ over the eigenvectors of $\mathbf{K}_A$. Then the approximated similarity matrix will be:

$$\begin{aligned} \widetilde{\mathbf{K}} &= \widetilde{\mathbf{\Phi}} \mathbf{\Pi}_A \widetilde{\mathbf{\Phi}}^T \\ &= \begin{bmatrix} \mathbf{\Phi}_A \\ \mathbf{K}_{AB}^T \mathbf{\Phi}_A \mathbf{\Pi}_A^{-1} \end{bmatrix} \mathbf{\Pi}_A \begin{bmatrix} \mathbf{\Phi}_A^T & \mathbf{\Pi}_A^{-1} \mathbf{\Phi}_A^T \mathbf{K}_{AB} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_A & \mathbf{K}_{AB} \\ \mathbf{K}_{AB}^T & \mathbf{K}_{AB}^T \mathbf{K}_A^{-1} \mathbf{K}_{AB} \end{bmatrix} \end{aligned} \tag{3.4}$$

Comparing (3.2) and (3.4) it can be seen that the huge matrix $\mathbf{K}_B$ is approximated by $\mathbf{K}_{AB}^T \mathbf{K}_A^{-1} \mathbf{K}_{AB}$.

A key aspect of the Nyström approximation is the sampling procedure in which the columns (or rows) of the original $\mathbf{K}$ are selected. The Nyström method was first introduced by a uniform distribution sampling over data [36]. Efficiency of the uniform sampling has been explored in many practical applications [38, 39]. More recently, theoretical aspects of nonuniform sampling techniques on real-world data sets have been studied [42, 43]. In general, these nonuniform sampling procedures are biased toward selection of the most informative points of the data-set. However, due to the imposed complexity of the nonuniform distribution updating procedure, practical application of these adaptive methods is limited.

In the current framework, our data are images which contain a high degree of spatial correlation between pixels. This leads us to use *spatially uniform* sampling instead of the random sampling procedure (comparisons given in Sec. 3.2.4). Spatially uniform sampling is a simple but effective approach in which samples are always equally spaced.

To study the performance of the Nyström approximation, we evaluate the relative accuracy defined in [43]:

$$\textit{Relative Accuracy} = \frac{\|\mathbf{K} - \mathbf{K}_{(r)}\|_F}{\|\mathbf{K} - \widetilde{\mathbf{K}}_{(r)}\|_F} \times 100$$

where $\mathbf{K}$ and $\mathbf{K}_{(r)}$ are the actual kernel and its exact rank-r approximation. The approximated kernel $\widetilde{\mathbf{K}}_{(r)}$ is reconstructed by using r leading eigenvectors from the Nyström method. The relative accuracy is lower bounded by zero and will ideally approach 100%.

The relative accuracy of approximating the globalized NLM kernel [2] as a function of the sampling rate is shown for some benchmark images in Fig. 3.3. We fixed $r = 50$ to capture about 90% of the spectral energy of the global kernel for each image. The samples are uniformly selected over the image lattice, and the relative accuracy is averaged for 20 sampling realizations. It can be seen that while higher sampling percentage leads to smaller error in the approximated kernel matrix, a saturation point is reached beyond 20% sampling density. Furthermore, for a fixed sampling rate the error depends on the contents of the underlying image. Surprisingly, textured images with high frequency components such as Mandrill produce less error compared to smooth images like House. This observation is consistent with results of [44] where it is shown that the error of the Nyström approximation is proportional to coherence of the kernel eigenvectors.

One could assume that at this point we can easily compute our approximated $\mathbf{W}$ and we are done! But as discussed earlier, statistical analysis of this filter needs access to its eigen-decomposition. Constructing a huge $\mathbf{W}$ matrix and then computing its eigenvectors is too expensive. Instead, in the following we explore an efficient way to find the eigenvectors of $\mathbf{W}$ directly.

3.2.2 Sinkhorn

The filter $\mathbf{W}$ is the row-normalized kernel matrix $\mathbf{K}$:

$$\mathbf{W} = \mathbf{D}^{-1}\mathbf{K} \tag{3.5}$$

where $\mathbf{D} = \text{diag}[\sum_{j=1}^{n} K_{1j}, \sum_{j=1}^{n} K_{2j}, .., \sum_{j=1}^{n} K_{nj}]$. We approximate the matrix $\mathbf{W}$ with a doubly-stochastic (symmetric) positive definite matrix, using Sinkhorn's algorithm [26]. Based on this method, given a positive valued matrix $\mathbf{K}$, there exist diagonal matrix $\mathbf{R} = \text{diag}(\mathbf{r})$ such that $\mathbf{W}_{sym} = \mathbf{RKR}$.

Since we have estimated the leading eigenvectors of $\mathbf{K}$, there is no need to compute $\mathbf{RKC}$. Instead, as can be seen in Algorithm 2, $\mathbf{W}_{sym}$ is approximated by its two sub-blocks $\mathbf{W}_A$ and $\mathbf{W}_{AB}$ where:

$$\mathbf{W}_{sym} = \begin{bmatrix} \mathbf{W}_A & \mathbf{W}_{AB} \\ \mathbf{W}_{AB}^T & \mathbf{W}_B \end{bmatrix} \tag{3.6}$$

Again, the Nyström method could give the approximated eigenvectors, but the only minor problem is that these eigenvectors are not quite orthogonal. In the following we discuss an approximation of the orthogonal eigenvectors.

[1] In the iterative row and column normalization process of Algorithm 2, construction of the matrix $\widetilde{\mathbf{K}}$ is avoided by using element-wise multiplication.

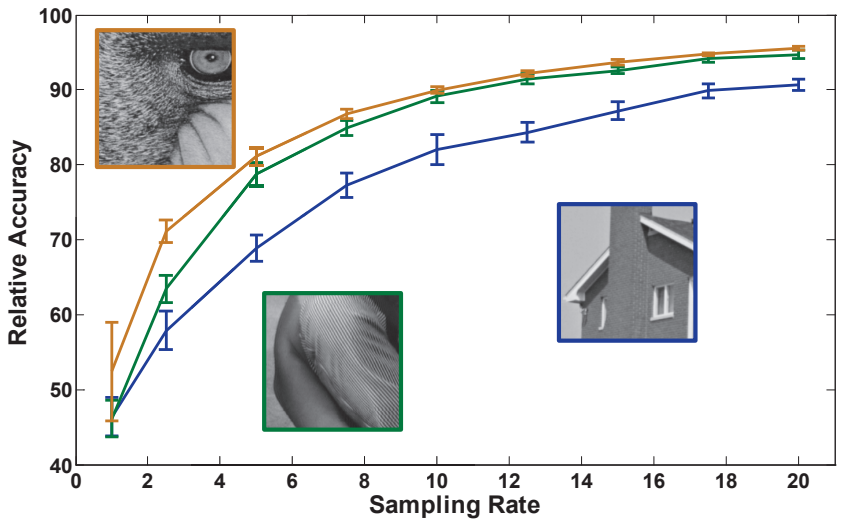


Figure 3.3: Accuracy of the kernel approximation for different sampling rates (sampling rate percentage is defined as $\frac{m}{n} \times 100\%$ where m denotes the number of samples and n represents number of pixels in the image). For the ease of computation of the exact filter, 150×150 subimages of Mandrill, Barbara and House are selected.

3.2.3 Orthogonalization

With the two sub-blocks $\mathbf{W}_A$ and $\mathbf{W}_{AB}$ in hand, here we derive an expression for approximating the orthogonalized eigenvectors $\mathbf{V}_m$. As discussed in [39], for any positive definite matrix, the orthogonalized approximated eigenvectors can be solved in one step. Let $\mathbf{W}_A^{1/2}$ denote the symmetric positive definite square root of $\mathbf{W}_A$. We define $\mathbf{Q} = \mathbf{W}_A + \mathbf{W}_A^{-1/2}\mathbf{W}_{AB}\mathbf{W}_{AB}^T\mathbf{W}_A^{-1/2}$ and we also consider the eigen-decomposition of this symmetric matrix as $\mathbf{Q} = \mathbf{V}_Q\mathbf{S}_Q\mathbf{V}_Q^T$. Then, it can been shown that the approximated symmetric $\mathbf{W}_m$ is diagonalized by $\mathbf{S}_m = \mathbf{S}_Q$ and $\mathbf{V}_m$ where:

$$\mathbf{V}_m = \begin{bmatrix} \mathbf{W}_A \\ \mathbf{W}_{AB}^T \end{bmatrix} \mathbf{W}_A^{-1/2}\mathbf{V}_Q\mathbf{S}_Q^{-1/2} \tag{3.7}$$

Then the approximated filter can be expressed as:

$$\mathbf{W}_m = \mathbf{V}_m\mathbf{S}_m\mathbf{V}_m^T \tag{3.8}$$

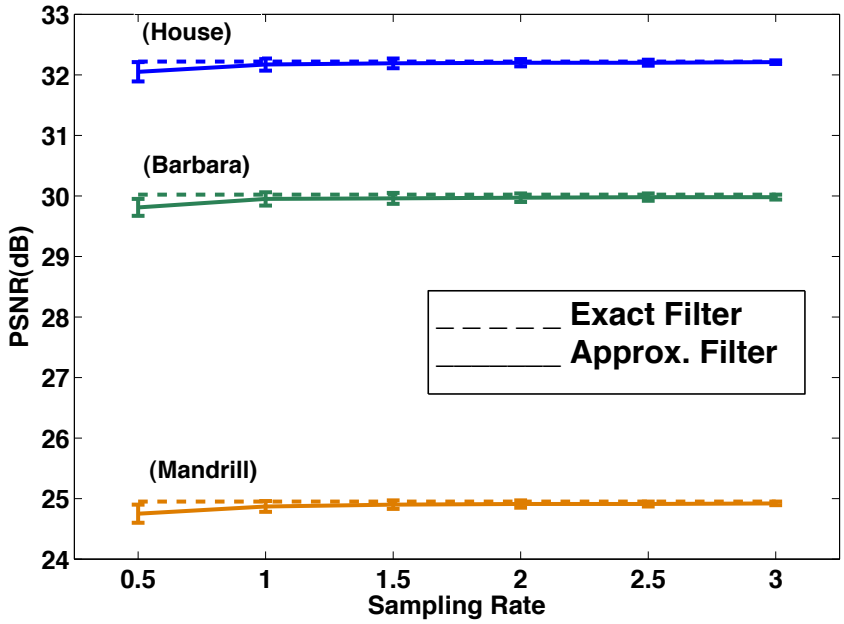


Figure 3.4: Comparison of the denoising performance (AWG with $\sigma = 20$) of the exact and approximated filter for the subimages in Fig. 3.3.

Proof of this approximation is given in Appendix 5.A.

The described three-step procedure provides us with an approximation of the leading eigenvectors and eigenvalues of the filter. Denoising performance of the approximated filter is compared to the exact filter in Fig. 3.4. These results suggest that the proposed approximation with a very small sampling rate can nearly match the performance of the exact filter.

As discussed previously, the spatially uniform sampling seems to fit our algorithm the best. To further understand the difference between uniform distribution and spatially uniform sampling, an experiment is carried out next.

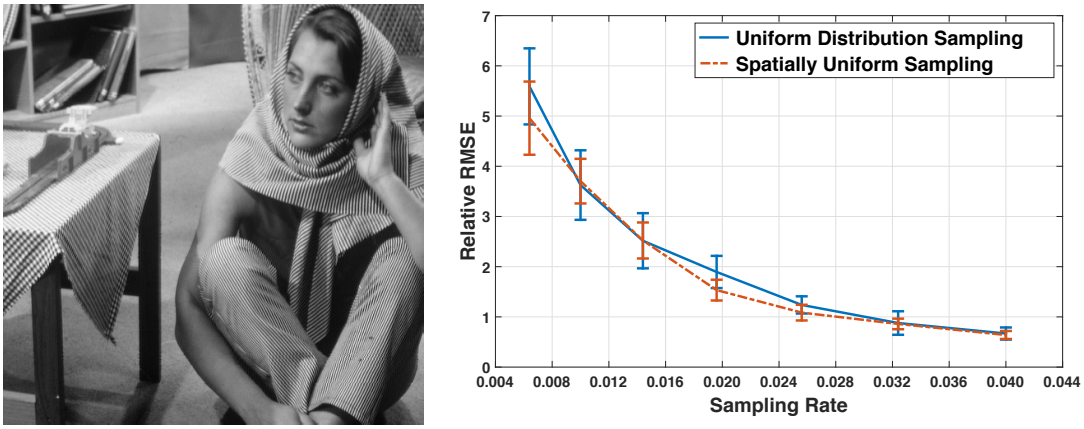


Figure 3.5: Left: Input image, Right: Approximation error of global filter computed for the input image as relative RMSE $= \frac{\|\mathbf{y}-\mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}\|}{\|\mathbf{y}\|} \times 100\%$ for spatially uniform, and uniform distribution sampling. For each sampling rate ($\frac{m}{n} \times 100\%$), associated error of 10 different realizations of the sampling methods are averaged. The error bars correspond to the standard deviation of the relative RMSE.

3.2.4 Spatially Uniform vs. Random Sampling

The approximated filter spans the input image onto its eigenvectors. This assumption means that the underlying image $\mathbf{y}$ should be "well represented" (or spanned) by these eigenvector bases. For a better validation of our sampling scheme, Fig. 3.5 shows relative root-mean-square-error (relative RMSE $= \frac{\|\mathbf{y}-\mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}\|}{\|\mathbf{y}\|} \times 100\%$) for the input test image. The relative RMSE values are computed for the image in range [0,255] and various values of m. Unsurprisingly, the approximation error of both sampling schemes (spatially uniform and uniform distribution) shrink as the number of retained eigenvectors grows. As can be seen, error associated with the spatial sampling, at worst, equals error from the random sampling. Another advantage of the spatially uniform sampling is the smaller variance of error, which leads to a more stable filter approximation.

Summary – Our proposed filter approximation procedure was explained in this chapter. Our contribution to the existing line of research is to lower the complexity of computing the global filter from quadratic to linear. This approximation relies on the low-rank property of the global matrix affinities, which is due to the self-similarities in natural images. The next chapters will discuss applications of the proposed filtering framework.

3.A Eigenvector orthonormalization

Having the two sub-blocks $\mathbf{W}_A$ and $\mathbf{W}_{AB}$ of the filter $\mathbf{W}$ and defining $\mathbf{Q} = \mathbf{W}_A + \mathbf{W}_A^{-1/2}\mathbf{W}_{AB}\mathbf{W}_{AB}^T\mathbf{W}_A^{-1/2}$ with the eigen-decomposition $\mathbf{Q} = \mathbf{V}_Q\mathbf{S}_Q\mathbf{V}_Q^T$, we aim to show that the orthonormal eigenvector bases for the estimated filter $\mathbf{W}_m$ are:

$$\mathbf{V}_m = \begin{bmatrix} \mathbf{W}_A \\ \mathbf{W}_{AB}^T \end{bmatrix} \mathbf{W}_A^{-1/2}\mathbf{V}_Q\mathbf{S}_Q^{-1/2} \tag{3.9}$$

We first need to check $\mathbf{W}_m = \mathbf{V}_m\mathbf{S}_Q\mathbf{V}_m^T$:

$$\begin{aligned} \mathbf{W}_m &= \{\begin{bmatrix} \mathbf{W}_A \\ \mathbf{W}_{AB}^T \end{bmatrix} \mathbf{W}_A^{-1/2}\mathbf{V}_Q\mathbf{S}_Q^{-1/2}\}\mathbf{S}_Q\{\mathbf{S}_Q^{-1/2}\mathbf{V}_Q^T\mathbf{W}_A^{-1/2} \begin{bmatrix} \mathbf{W}_A & \mathbf{W}_{AB} \end{bmatrix}\} \\ &= \mathbf{V}_m\mathbf{S}_Q\mathbf{V}_m^T \end{aligned} \tag{3.10}$$

In addition, we check the orthogonality of $\mathbf{V}_m$ as follows:

$$\begin{aligned} \mathbf{V}_m^T\mathbf{V}_m &= \mathbf{S}_Q^{-1/2}\mathbf{V}_Q^T\mathbf{W}_A^{-1/2} \begin{bmatrix} \mathbf{W}_A & \mathbf{W}_{AB} \end{bmatrix} \begin{bmatrix} \mathbf{W}_A \\ \mathbf{W}_{AB}^T \end{bmatrix} \mathbf{W}_A^{-1/2}\mathbf{V}_Q\mathbf{S}_Q^{-1/2} \\ &= \mathbf{S}_Q^{-1/2}\mathbf{V}_Q^T\mathbf{Q}\mathbf{V}_Q\mathbf{S}_Q^{-1/2} \\ &= \mathbf{I} \end{aligned} \tag{3.11}$$

As a result, the approximated eigen-decomposition is orthogonal.

Chapter 4

Global Image Denoising

Abstract – Most existing state-of-the-art image denoising algorithms are based on exploiting similarity between a relatively modest number of *patches*. These patch-based methods are strictly dependent on patch matching, and their performance is hamstrung by the ability to reliably find sufficiently similar patches. As the number of patches grows, a point of diminishing returns is reached where the performance improvement due to more patches is offset by the lower likelihood of finding sufficiently close matches. The net effect is that while patch-based methods such as BM3D are excellent overall, they are ultimately limited in how well they can do on (larger) images with increasing complexity. In this work, we address these shortcomings by developing a paradigm for truly global filtering where each pixel is estimated from *all* pixels in the image. Our objective in this chapter is to give a statistical analysis of our proposed global filter, based on a spectral decomposition of its corresponding operator. This framework relies on the filter approximation discussed in Chapter 3, where the spectral (principal) components of the global affinities are approximated using the Nyström extension. Experiments illustrate that our strategy can effectively *globalize* any existing

denoising filters to estimate each pixel using all pixels in the image, hence improving upon the best patch-based methods.

4.1 Global Denoising Scheme

The block diagram of the proposed global image denoising (GLIDE) framework is illustrated in Fig. 4.1. As can be seen, after applying a pre-filter on the noisy image, a small fraction of the pixels are sampled to be fed to the Nyström method. Then, the global filter is approximated through its eigenvalues and eigenvectors. The final estimate of the image is constructed by means of shrinkage of the filter eigenvalues.

Denoting n as the total number of the pixels in the image, our one-shot, global, full-space filter for the whole image can be expressed as:

$$\widehat{\mathbf{z}} = \mathbf{W}\mathbf{y} = \mathbf{V}\mathbf{S}\mathbf{V}^T\mathbf{y}, \tag{4.1}$$

in which the eigenvectors $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_n]$ specify a complete orthonormal basis for $\mathbb{R}^n$ and $\mathbf{S} = \text{diag}[\lambda_1, ..., \lambda_n]$ contains the eigenvalues indexed in decreasing order $0 \leq \lambda_n \leq ... < \lambda_1 = 1$. This implies that the input image $\mathbf{y}$ is first projected onto the full-space eigenvectors of $\mathbf{W}$, then each mode of the projected signal is shrunk by its corresponding eigenvalue, and finally after mapping back to the signal domain, the recovered signal $\widehat{\mathbf{z}}$ is produced.

Not surprisingly, the computational burden of constructing and decomposing such a large matrix as $\mathbf{W}$ is prohibitively high. However, the Nyström approximation, combined with our statistical analysis allows an efficient solution [41]. The low-rank approximation procedure described in Chapter 3 shows that:

$$\mathbf{W} \approx \mathbf{W}_m = \mathbf{V}_m\mathbf{S}_m\mathbf{V}_m^T, \tag{4.2}$$

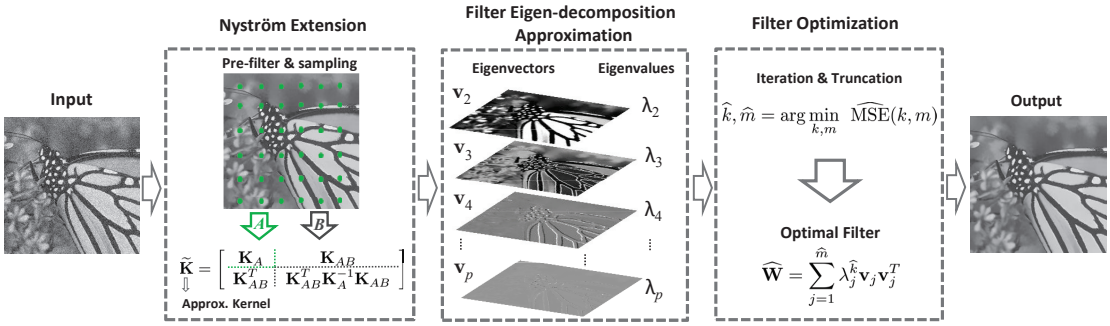


Figure 4.1: GLIDE's pipeline. From left to right, for a noisy image we first apply a pre-filter to reduce the noise level. Then using a spatially uniform sampling, the global kernel is approximated by employing the Nyström extension (A and B represent the samples and the rest of the pixels in the image, respectively). As discussed in Chapter. 3, using the obtained kernel, the leading eigenvalues and eigenvectors of the filter are approximated (The eigenvector $\mathbf{v}_1$ is not shown because it is constant). Finally, the optimal filter is constructed by shrinking (iteration and truncation) the eigenvalues. The filter optimization step is detailed in Section 4.2.

where $\mathbf{V}_m = [\mathbf{v}_1, ..., \mathbf{v}_m]$ denote an orthonormal basis for $\mathbb{R}^m$ and $\mathbf{S}_m = \text{diag}[\lambda_1, ..., \lambda_m]$ in which $m << n$. The behavior of the full-space and approximated filter in terms of MSE is analyzed next.

4.2 Statistical Analysis of the Global Filter

Let's assume that all of the eigenmodes of the filter $\mathbf{W}$ are used without any change and the filter $\mathbf{W}$ is stochastically independent from the input image $\mathbf{y}$. Then, starting from the MSE of each pixel, in Chapter 2 we showed that the overall ideal MSE for the whole image is:

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(1-\lambda_j)^2 b_j^2 + \sigma^2\lambda_j^2 \tag{4.3}$$

where in the above, $\|\text{bias}(\widehat{\mathbf{z}})\|^2 = \frac{1}{n}\sum_{j=1}^{n}(1-\lambda_j)^2 b_j^2$ and $\text{var}(\widehat{\mathbf{z}}) = \frac{\sigma^2}{n}\sum_{j=1}^{n}\lambda_j^2$ and $\mathbf{b} = \mathbf{V}^T\mathbf{z} = [b_1, ..., b_n]^T$ contains the projected signal in all modes. Apparently, MSE

is a function of the latent signal, noise, filter eigenvalues and eigenvectors. The filter eigenvalues are the shrinkage factors which directly tune the filtering performance.

As discussed in [23], minimization of the MSE as a function of the filter eigenvalues leads to the Wiener filter:

$$\lambda_j^* = \frac{1}{1 + \text{snr}_j^{-1}} \tag{4.4}$$

where $\text{snr}_j = \frac{b_j^2}{\sigma^2}$. This optimum shrinkage requires exact knowledge of the signal-to-noise ratio in each channel.

Estimation accuracy can sometimes be improved by shrinking or setting some coefficients to zero. By doing so we may sacrifice some bias to reduce the variance of the estimated values, and hence may improve the overall estimation accuracy.

4.2.1 Truncated Filter

The filtering framework in (4.1) can be performed for the leading (say $m < n$) eigenvalues of the filter $\mathbf{W}$. As we show in Appendix 5.A, such a filter has the following MSE:

$$\text{MSE}(m) = \frac{1}{n}\sum_{i=1}^{n} z_i^2 + \frac{1}{n}\sum_{j=1}^{m}\left((\lambda_j^2 - 2\lambda_j)b_j^2 + \sigma^2\lambda_j^2\right) \tag{4.5}$$

where in the above, $\|\text{bias}(\widehat{\mathbf{z}})\|^2 = \sum_{i=1}^{n} z_i^2 + \sum_{j=1}^{m}(\lambda_j^2 - 2\lambda_j)b_j^2$ and $\text{var}(\widehat{\mathbf{z}}) = \sigma^2\sum_{j=1}^{m}\lambda_j^2$. For the sake of comparison, we can assume that all the signal modes are available and then $\sum_{i=1}^{n} z_i^2$ can be replaced with $\sum_{i=j}^{n} b_j^2$. After some simplifications we can rewrite our MSE expression as:

$$\text{MSE}(m) = \underbrace{\frac{1}{n}\sum_{j=1}^{n}(1-\lambda_j)^2 b_j^2 + \sigma^2\lambda_j^2}_{\text{MSE}(n)} + \underbrace{\frac{1}{n}\sum_{j=m+1}^{n}(2\lambda_j - \lambda_j^2)b_j^2 - \sigma^2\lambda_j^2}_{\Delta\text{MSE}} \tag{4.6}$$

As can be seen, MSE(m) of the truncated filter differs from (4.3) by the amount given in the second term of (4.6); i.e. $\Delta\text{MSE} = \frac{1}{n}\sum_{j=m+1}^{n}(2\lambda_j - \lambda_j^2)b_j^2 - \sigma^2\lambda_j^2$. This difference is also composed of bias and variance parts as $\Delta\text{MSE} = \Delta\|\text{bias}\|^2 + \Delta\text{var}$ where

$$\Delta\|\text{bias}\|^2 = \frac{1}{n}\sum_{j=m+1}^{n}(2\lambda_j - \lambda_j^2)b_j^2 \tag{4.7}$$

$$\Delta\text{var} = -\frac{1}{n}\sum_{j=m+1}^{n}\sigma^2\lambda_j^2 \tag{4.8}$$

This shows, consistent with intuition, that the truncation lowers the variance and increases the bias. Given this analysis, we can determine when truncation improves the MSE. That is, when $\Delta\text{MSE} < 0$. A simple sufficient condition is that for all j,

$$\text{snr}_j < \frac{\lambda_j}{2 - \lambda_j} \tag{4.9}$$

Intuitively we can conclude that all the channels in the range of $m + 1 \leq j \leq n$ with sufficiently small signal-to-noise ratio should be set to zero. This inequality can also be expressed as:

$$\lambda_j > \frac{2}{1 + \text{snr}_j^{-1}} \tag{4.10}$$

Comparing this inequality with the Wiener shrinkage criterion in (4.4), it can be seen that $\lambda_j > 2\lambda_j^*$. That is, the condition implied by (4.10) is a stronger form of shrinkage than what the Wiener condition would dictate.

4.2.2 Iterative Filter

Although the estimated MSE can be reduced by truncating some of the eigenmodes, hard thresholding prevents the accuracy of the estimation to be close to optimal. To ameliorate this shortcoming, iteration can gradually tune the (truncated) filter to softly vary its filtering strength. As such, the iteration and truncation numbers are the

only parameters to be globally optimized. Our iterative diffusion model [23] is:

$$\widehat{\mathbf{z}} = \widehat{\mathbf{W}}\mathbf{y} = \mathbf{V}_m \mathbf{S}_m^k \mathbf{V}_m^T \mathbf{y}, \tag{4.11}$$

where $\mathbf{V}_m = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]$, $\mathbf{S}_m^k = \text{diag}[\lambda_1^k, \lambda_2^k, ..., \lambda_m^k]$ and k denotes the iteration number.[1] Fig. 4.2 illustrates corresponding filter weights of the marked pixel in the House image. As can be seen, different iteration and truncation numbers can effectively vary the behavior of the filter to find similar pixels all over the image.

With this model, we can rewrite (4.5) for the truncated iterative filter:

$$\text{MSE}(k, m) = \frac{1}{n}\sum_{i=1}^{n} z_i^2 + \frac{1}{n}\sum_{j=1}^{m}\left((\lambda_j^{2k} - 2\lambda_j^k)b_j^2 + \sigma^2\lambda_j^{2k}\right) \tag{4.12}$$

Overall, our minimization problem will be extended to estimating the shrinkage ($\widehat{k}$) and truncation ($\widehat{m}$) factors from an estimate of MSE:

$$\widehat{k}, \widehat{m} = \arg\min_{k,m} \ \widehat{\text{MSE}}(k, m) \tag{4.13}$$

where $\widehat{\text{MSE}}(k, m)$ denotes an estimate of $\text{MSE}(k, m)$. The shrinkage and truncation parameters are simultaneously optimized such that $(\widehat{k}, \widehat{m})$ is the global minimum of $\widehat{\text{MSE}}(k, m)$. Minimization of $\widehat{\text{MSE}}(k, m)$ determines the best parameters to help avoid under- or over-smoothing.

Although the diffusion iteration is chosen in our framework, our analysis makes it possible to use other iterations too. In general, any iterative approach can be defined as substituting the eigenvalues λ_j with a shrinkage function $f_k(\lambda_j)$ where in the case of diffusion $f_k(\lambda_j) = \lambda_j^k$. Another alternative can be the *boosting* iteration which is a complementary mechanism to recycle lost details of the filtered signal (See Chapter. 2 for more details). In this case, the eigenvalues will be modified as $f_k(\lambda_j) = 1 - (1 - \lambda_j)^{k+1}$.

[1] It is noteworthy that the spectral decomposition of $\mathbf{W}^k$ makes it possible to replace k with any real number. It is important to note that because of this, k can really be thought of as a shrinkage

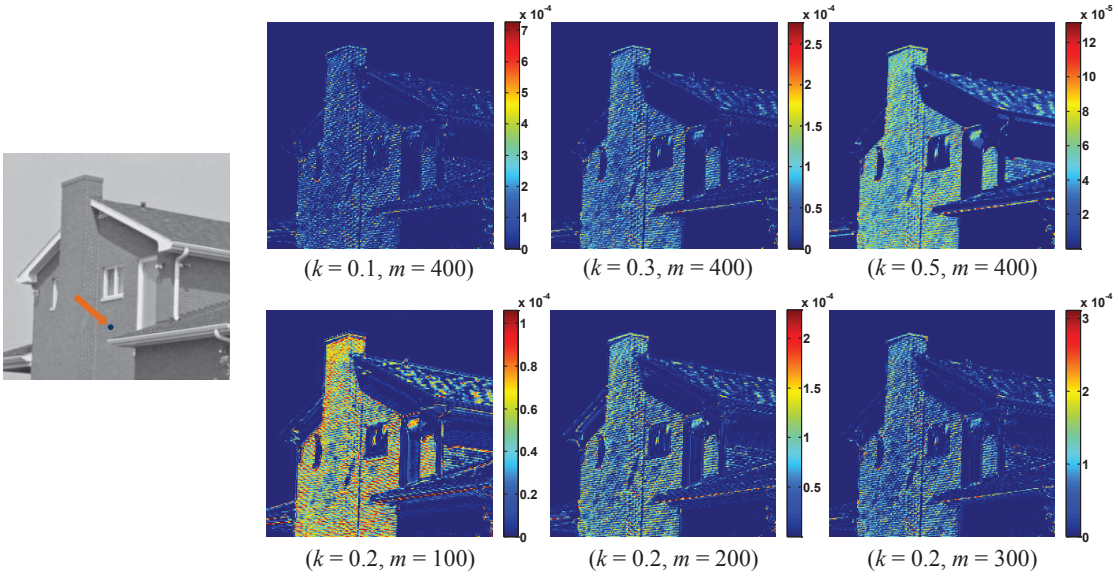


Figure 4.2: Filter weights with different shrinkage (k) and truncation (m) parameters are computed for the labeled pixel in the House image.

4.2.3 Practical Filtering

In the estimation of the MSE in (4.12), we assumed that the filter $\mathbf{W}$ is stochastically independent from the input image $\mathbf{y}$. It has been shown that in the case of a smooth filter (kernel with small gradient), a pre-filter can effectively decouple $\mathbf{W}$ from $\mathbf{y}$ [26]. The smoothness of the filter is approximately true when the filter is computed for locally homogenous patches. By approximating the local signal-to-noise ratio, this type of MSE estimator has been shown to work quite well in Chapter 2. However, in the case of the global filter, pixels with different local structures are connected to each other; which means drastic changes in the filter values. In other words, $\mathbf{W}$ and $\mathbf{y}$ are not stochastically decoupled. This nonlinearity specifically affects the estimated variance in

parameter that controls the rate of decay of the modified eigenvalues λ_j^k. Going forward, we will use the terms "shrinkage factor" and "iteration number" interchangeably.

the ideal MSE presented in (4.12). Inspired by the SURE estimator [30], we can show that the estimated variance can be modified as:

$$\text{var}(\widehat{\mathbf{z}}) \approx \frac{\sigma^2}{n}\sum_{j=1}^{m}\lambda_j^{2k} + \frac{2\sigma^2}{n}\left(\text{div}(\widehat{\mathbf{z}}(\mathbf{y})) - \sum_{j=1}^{m}\lambda_j^{k}\right) \tag{4.14}$$

where $\text{div}(\widehat{\mathbf{z}}(\mathbf{y})) \equiv \sum_i \frac{\partial \widehat{z}_i(\mathbf{y})}{\partial y_i}$. In the case of a strictly linear filter, $\text{div}(\widehat{\mathbf{z}}(\mathbf{y})) = \sum_{j=1}^{m}\lambda_j^k$, which leads to the ideal variance in (4.12). Intuitively, the second term in (4.14) takes care of the variance due to the nonlinearity.

It is quite straightforward to show that the bias term in (4.12) can be better estimated as:

$$\|\text{bias}(\widehat{\mathbf{z}})\|^2 \approx \frac{1}{n}\sum_{i=1}^{n} y_i^2 - \sigma^2 + \frac{1}{n}\sum_{j=1}^{m}(\lambda_j^{2k} - 2\lambda_j^k)(\check{b}_j^2 - \sigma^2) \tag{4.15}$$

where $\check{\mathbf{b}} = \mathbf{V}^T\mathbf{y}$. The expected value of this estimator is exactly the ideal squared bias in (4.12). Overall, the estimated MSE of the general nonlinear filter has the following form:

$$\widehat{\text{MSE}}(k,m) = \text{SURE}(k,m) = \frac{1}{n}\sum_{i=1}^{n} y_i^2 - \sigma^2 + \frac{1}{n}\sum_{j=1}^{m}(\lambda_j^{2k} - 2\lambda_j^k)\check{b}_j^2 + 2\sigma^2\text{div}(\widehat{\mathbf{z}}(\mathbf{y})) \tag{4.16}$$

which is the SURE estimator [30]. For large data sets (such as our global framework), closeness of the SURE risk estimator to the actual MSE is assured by the law of large numbers.

Approximation of $\text{div}(\widehat{\mathbf{z}}(\mathbf{y}))$ has been studied in [45] and [27]. Ramani's Monte-Carlo algorithm [27] uses a first-order difference approximation to obtain an estimate of the divergence term. Based on this method, the divergence term can be computed from $\text{div}(\widehat{\mathbf{z}}(\mathbf{y})) = \frac{1}{\epsilon}\mathbf{a}^T(\widehat{\mathbf{z}}(\mathbf{y}) - \widehat{\mathbf{z}}(\mathbf{y}'))$ where $\mathbf{y}' = \mathbf{y} + \epsilon\mathbf{a}$ in which $\mathbf{a}$ is a zero-mean i.i.d random vector of unit variance and in practice ϵ gets small positive values (in theory $\epsilon \longrightarrow 0$).

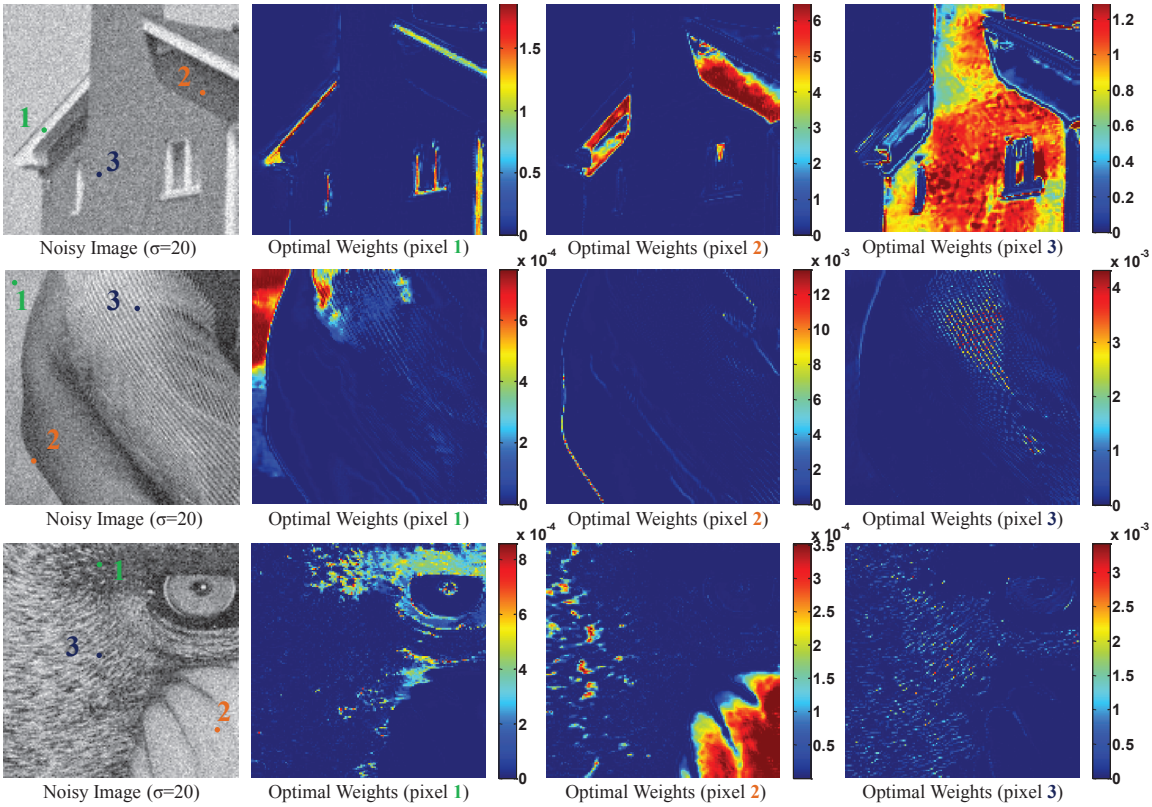


Figure 4.3: Optimal filter weights for the labeled pixels in the images. The optimal iteration and truncation numbers for each image are estimated as, House: $\widehat{k} = 0.16$ and $\widehat{m} = 40$, Barbara: $\widehat{k} = 0.14$ and $\widehat{m} = 65$, Mandrill: $\widehat{k} = 0.33$, $\widehat{m} = 165$.

Performance of the proposed estimator in (4.16) is evaluated in Figs. 4.3 and 4.4. As can be seen in Fig. 4.3, the optimized filter weights for the labeled pixels are computed based on the estimated iteration and truncation parameters. The actual and estimated MSE plots are shown in Fig. 4.4 where, as can be seen, the estimated shrinkage parameters are very close to their actual values.

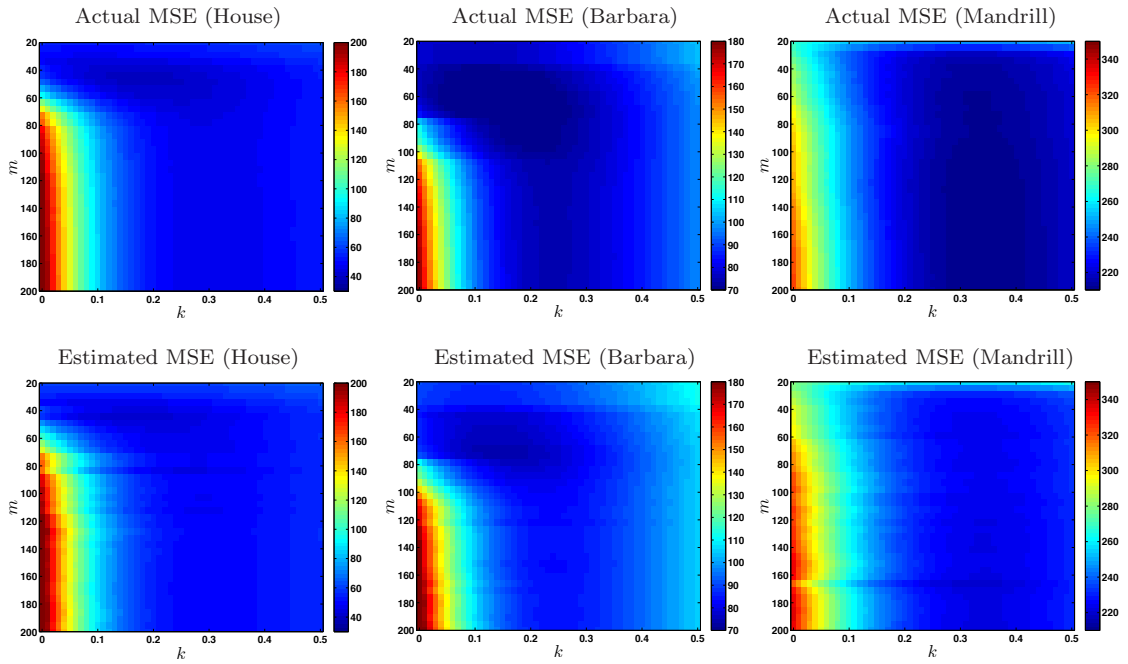


Figure 4.4: Corresponding MSE of the images in Fig. 4.3. The ideal and estimated iteration and truncation numbers are respectively: House (ideal: 0.19, 45, estimated: 0.16, 40), Barbara (ideal: 0.14, 65, estimated: 0.14, 65), Mandrill (ideal: 0.34, 160, estimated: 0.33, 165).

4.3 Results and Comparisons

In this section, performance of our algorithm is compared to state-of-the-art denoising methods for some benchmark images. We selected NLM [2] as our baseline kernel; however, any other non-local kernel could also be used. The filter approximation described in Chapter 3 is employed in our experiments. Pixel samples for the Nyström extension are uniformly selected and the sampling rate is set as 1% and is kept fixed throughout the experiments.

Performance of the proposed filter is quantified across different noise levels in Table 4.1. For each noise level, we report the Monte-Carlo average performance for each algorithm over 5 different noise realizations. We highlight (in bold) both the best

Table 4.1: PSNR values of NLM [2] (1st column), and the proposed method (2nd column). Results noted are average PSNR (top) and SSIM [10] (bottom) over 5 independent noise realizations for each σ.

σ	Monarch		House		Cameraman		Aerial		Mandrill		Stream		Avgerage Improvement
	NLM	G-NLM	NLM	G-NLM	NLM	G-NLM	NLM	G-NLM	NLM	G-NLM	NLM	G-NLM	
10	33.43	**33.50**	35.25	**35.46**	33.43	**33.62**	30.54	**30.78**	30.01	**30.30**	30.54	**30.79**	0.22
	0.940	**0.945**	0.893	**0.901**	0.914	**0.921**	**0.930**	**0.932**	0.891	**0.903**	0.893	**0.903**	0.009
20	29.26	**29.67**	32.37	**32.63**	29.41	**29.49**	26.23	**26.59**	25.69	**25.96**	26.36	**26.67**	0.29
	0.894	**0.903**	0.847	**0.857**	0.834	**0.852**	0.829	**0.834**	0.769	**0.781**	0.736	**0.748**	0.014
30	27.16	**27.65**	30.18	**30.57**	27.54	**27.78**	24.36	**24.65**	23.85	**24.42**	24.69	**25.19**	0.39
	0.841	**0.863**	0.796	**0.827**	0.786	**0.817**	0.755	**0.770**	0.652	**0.684**	0.646	**0.671**	0.031
40	25.53	**26.32**	28.32	**29.01**	25.98	**26.47**	23.01	**23.34**	22.69	**22.36**	23.61	**24.29**	0.62
	0.778	**0.833**	0.721	**0.799**	0.715	**0.787**	0.691	**0.713**	0.578	**0.628**	0.579	**0.630**	0.055
50	24.44	**25.24**	26.94	**27.73**	24.90	**25.28**	21.95	**22.39**	21.80	**22.64**	22.75	**23.56**	0.71
	0.727	**0.802**	0.670	**0.773**	0.657	**0.735**	0.629	**0.658**	0.507	**0.584**	0.518	**0.592**	0.074

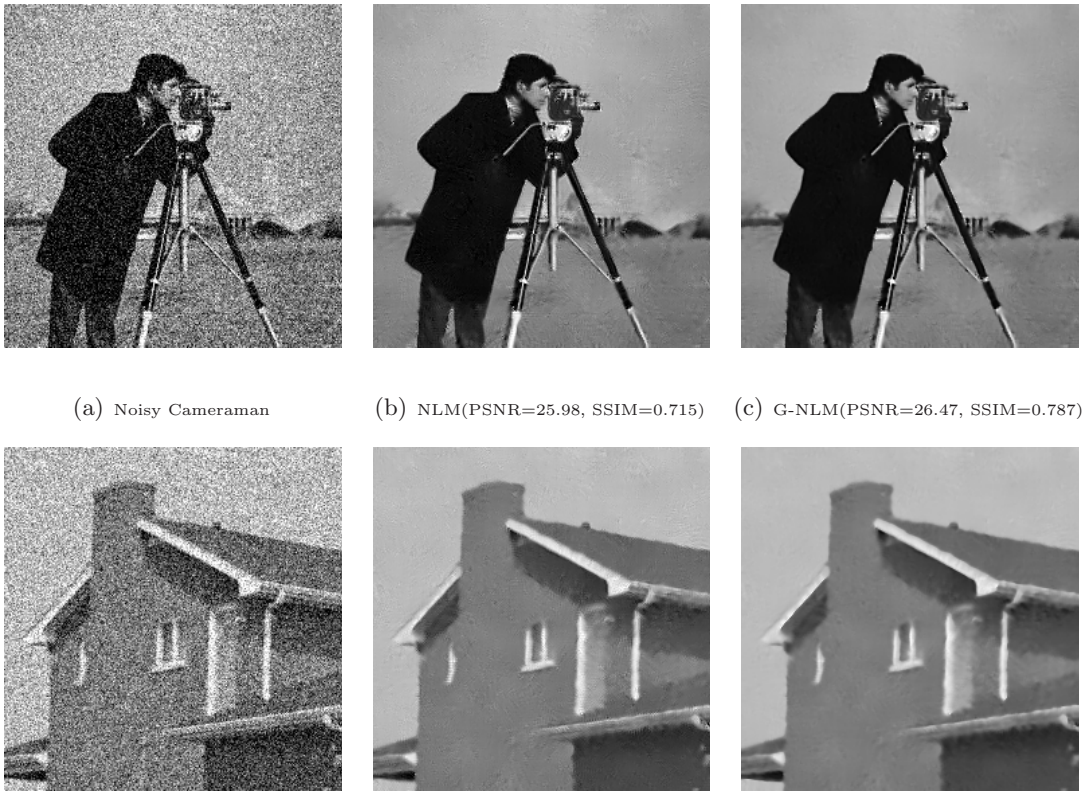
results, and also results that are statistically within the margin of standard error from the best results (0.05 dB in PSNR and 0.005 SSIM). In this set of experiments, the pre-filtered images are obtained from NLM. As can be seen, our method consistently improves upon NLM in terms of PSNR and SSIM index [10], especially for high noise levels where local similar pixels are more difficult to find.

Fig. 4.5 demonstrates the denoising results obtained by NLM compared to the proposed method. In addition to the PSNR improvement, visual quality of the proposed method also is superior to the NLM filter. As it can be seen, both edges and smooth features of the image are preserved better than the other methods.

In the next set of experiments shown in Table 4.2, the pre-filtered images are obtained from BM3D [5]. Our method can improve upon BM3D especially at high noise levels and for images with semi-stochastic textures which contain relatively few similar patches.

Denoising results for the Mandrill and Monarch images for BM3D and the globalized BM3D are compared in Fig. 4.6. As can be seen, the proposed method can bootstrap the performance of BM3D.

Since in practice the distribution of the noise is not additive white Gaussian,



(a) Noisy Cameraman (b) NLM(PSNR=25.98, SSIM=0.715) (c) G-NLM(PSNR=26.47, SSIM=0.787)

(d) Noisy House (e) NLM(PSNR=28.32, SSIM = 0.721) (f) G-NLM(PSNR=29.01, SSIM = 0.799)

Figure 4.5: Comparison of denoising performance on noisy images corrupted by AWGN of $\sigma = 40$. (a),(d) Noisy input, (b),(e) NLM [2], (c),(f) G-NLM.

we also tested our algorithm for real noise in color images[2]. In this set of experiments the best results are optimized using the no-reference quality metric in [28]. Fig. 4.7 shows performance of the proposed method for improving the NLM filter. We also compare our results to the commercial Neat Image™denoising software in Fig. 4.8. As can be seen, our result is competitive to the commercial state-of-the art denoising. We note that our Matlab code and additional results are available at the project website[3].

Running time for denoising a 256×256 grayscale image with an unoptimized

[2]The color denoising is applied in YUV space, where the weights are computed from the Y channel, and applied to the U and V channels.

[3]http://www.soe.ucsc.edu/~htalebi/GLIDE.php.

Table 4.2: PSNR values of BM3D [5] (1st column), and the proposed method (2nd column). Results noted are average PSNR (top) and SSIM [10] (bottom) over 5 independent noise realizations for each σ.

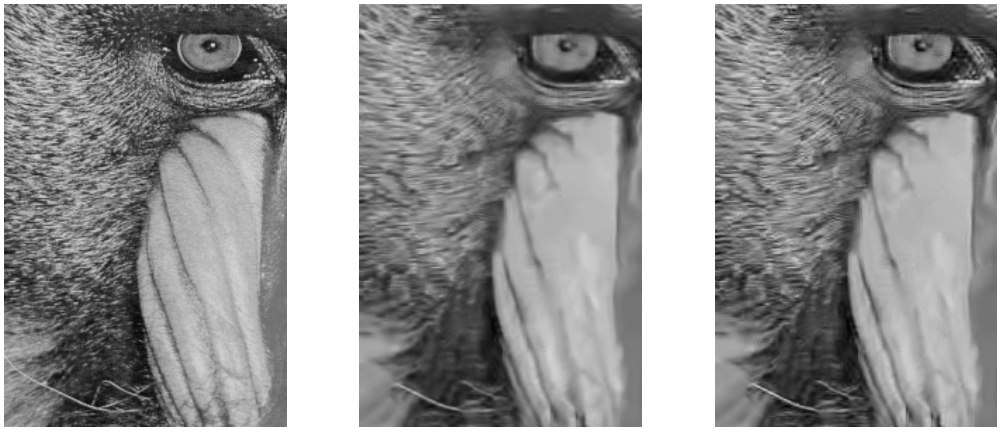
σ	Monarch		House		Cameraman		Aerial		Mandrill		Stream		Average
	BM3D	G-BM3D	BM3D	G-BM3D	BM3D	G-BM3D	BM3D	G-BM3D	BM3D	G-BM3D	BM3D	G-BM3D	Improvement
10	**34.12**	**34.08**	**36.67**	**36.70**	**34.05**	**34.06**	**31.09**	**31.07**	30.57	**30.65**	**31.14**	**31.17**	0.02
	0.956	**0.956**	**0.920**	**0.921**	**0.930**	**0.932**	**0.938**	**0.939**	0.896	**0.902**	**0.906**	**0.909**	0.002
20	30.42	**30.56**	**33.78**	**33.81**	**30.41**	**30.45**	27.22	**27.32**	26.58	**26.71**	27.25	**27.33**	0.08
	0.920	**0.923**	0.871	**0.872**	0.874	**0.878**	0.864	**0.869**	0.790	**0.802**	0.790	**0.798**	0.006
30	28.42	**28.52**	**32.04**	**32.09**	**28.58**	**28.61**	25.24	**25.31**	24.53	**24.70**	25.46	**25.59**	0.09
	0.885	**0.889**	0.846	**0.849**	0.835	**0.840**	0.798	**0.806**	0.697	**0.728**	0.700	**0.719**	0.012
40	26.66	**26.85**	30.56	**30.61**	27.09	**27.20**	23.77	**23.90**	23.07	**23.25**	24.32	**24.51**	0.15
	0.846	**0.851**	0.822	**0.828**	0.804	**0.819**	0.737	**0.740**	0.614	**0.652**	0.630	**0.654**	0.015
50	25.72	**25.98**	29.64	**29.73**	26.05	**26.28**	22.94	**23.14**	22.32	**22.57**	23.57	**23.79**	0.21
	0.820	**0.827**	0.809	**0.813**	0.779	**0.801**	0.690	**0.705**	0.545	**0.587**	0.575	**0.596**	0.019

implementation of our method is about 160 seconds on a 2.8 GHz Intel Core i7 processor. However, parallelizing can significantly speed up our method. For instance, running time of the parallelized version of our code executed with 4 separate cores takes about 50 seconds.

4.4 Oracle Results and Existing Room for Improvement

For better understanding of the global filter, we studied the oracle performance of the proposed method and compared this to the oracle performance [4] of other (mainly patch-based) methods in Table 4.3. As can be seen, the oracle GLIDE outperforms other oracle methods by a significant margin. While this margin is only a bound on how much improvement we can expect in practice, it does convey an interesting and tantalizing message. Namely (at least asymptotically) patch-based methods are inherently limited in performance [20] in a way that global filtering is not. More specifically, the oracle PSNR values for the global filter point to essentially perfect reconstruction of the noise-free image, which is apparently impossible to achieve for oracle versions of algorithms

[4]The oracle NLM has all the kernel weights computed from the clean image and in the case of the oracle BM3D [5] which has been shown to be near optimal [46] (among patch-based methods), the pre-filtered image is replaced by the clean image (which means that the Wiener shrinkage and the patch grouping are implemented perfectly). Similarly, the oracle GLIDE has all the global weights computed from the clean image.



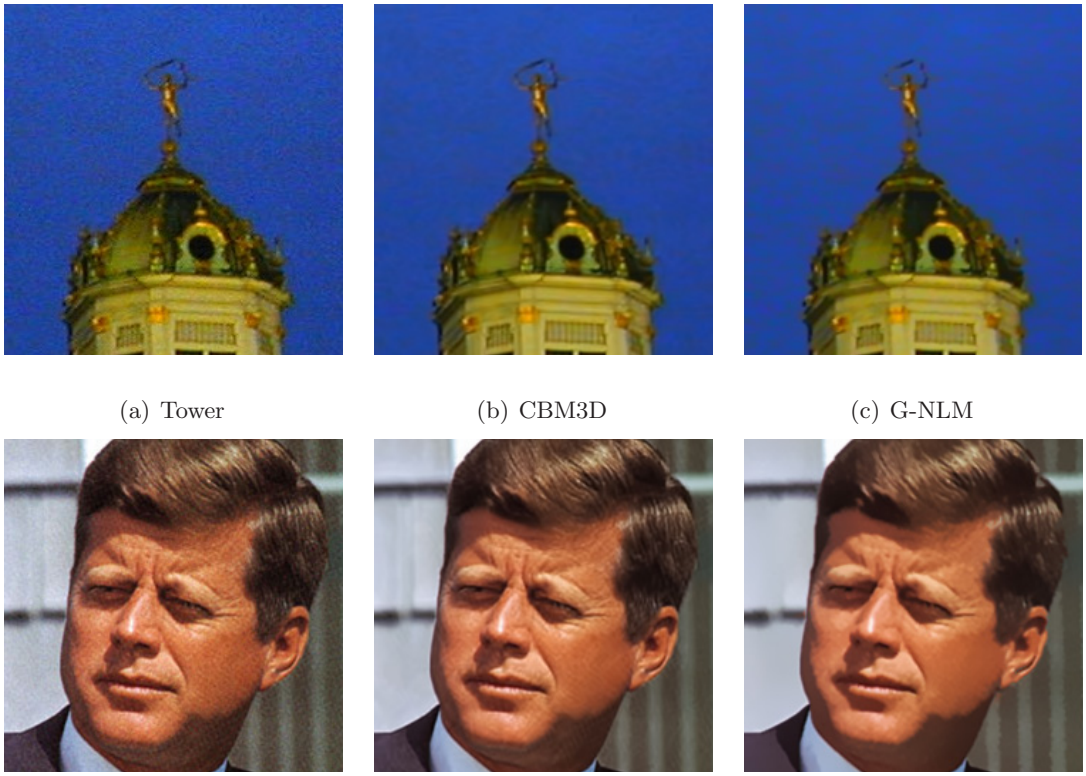
(a) Zoomed Mandrill (b) BM3D (c) G-BM3D

(d) Zoomed Monarch (e) BM3D (f) G-BM3D

Figure 4.6: Comparison of denoising performance on noisy images corrupted by AWGN of $\sigma = 50$. (a),(d) Original image, (b) BM3D [5] (PSNR=22.32, SSIM=0.545), (c) G-BM3D (PSNR=22.57, SSIM=0.587), (e) BM3D [5] (PSNR=25.72, SSIM=0.820), (f) G-BM3D (PSNR=25.98, SSIM=0.827).

like such as BM3D, even if all the filter parameters are known exactly.

The next chapter discusses oracle performance of the global filter in more depth. Using the estimated MSE, an upper bound on the performance of the global filter is derived in Chapter 5. More specifically, regardless of the image content, the MSE bound has a decay rate of $O(\frac{1}{n})$ where n denotes the number of pixels.

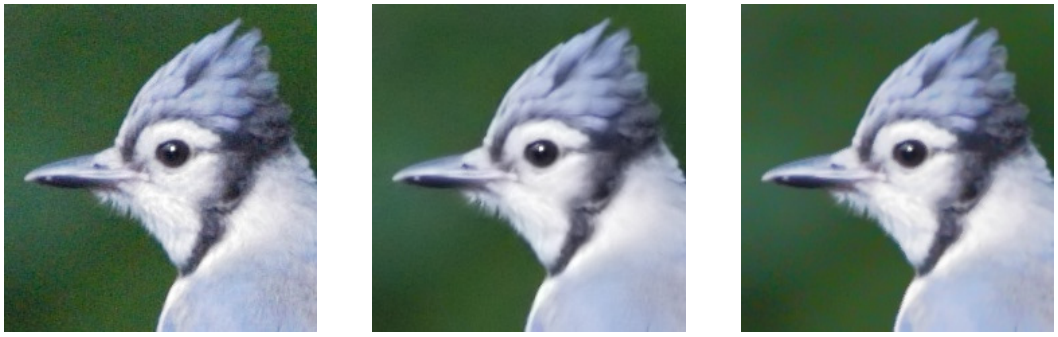


(a) Tower (b) CBM3D (c) G-NLM

(d) JFK (e) CBM3D (f) G-NLM

Figure 4.7: Comparison of denoising performance on the real noise. (a) and (d) Noisy image, (b) and (e) CBM3D [5], (c) and (f) G-NLM.

Summary – This work is, to our knowledge, the very first truly global denoising algorithm to be proposed. The global approach goes beyond the dominant paradigm of non-local patch-based processing, which we have shown here to be inherently limited. The specific contribution we have made is to develop a practical algorithm to compute a global filter which in effect uses all the pixels in the input image to denoise every single pixel. By exploiting the Nyström extension, we have made the global approach computationally tractable. Since the global filter uses all the pixels of the image, exact computation of the filter weights has a complexity $O(n^2)$, whereas the proposed sample



(a) Bird (Noisy) (b) Neat Image™ (c) G-NLM

Figure 4.8: Comparison of denoising performance on the real noise. (a) Noisy image, (b) Neat Image™, (c) G-NLM. (Neat Image™denoising software is available at http://www.neatimage.com.)

Table 4.3: PSNR values of **oracle** NLM [2] (1st column), **oracle** BM3D [5] (2nd column), and the **oracle** GLIDE (3rd column). Results noted are average PSNR over 5 independent noise realizations for each σ.

σ	Monarch			House			Cameraman			Aerial		
	NLM	**BM3D**	**GLIDE**	**NLM**	**BM3D**	**GLIDE**	**NLM**	**BM3D**	**GLIDE**	**NLM**	**BM3D**	**GLIDE**
10	35.01	36.55	**43.16**	36.52	39.29	**51.24**	34.82	36.72	**41.58**	31.95	33.56	**48.75**
20	30.87	33.02	**41.83**	34.05	36.54	**47.18**	31.01	33.15	**40.49**	27.78	29.75	**45.37**
30	28.95	31.08	**40.41**	32.48	34.97	**44.89**	28.91	31.20	**39.28**	26.13	27.74	**42.99**
40	27.62	29.75	**39.27**	31.18	33.82	**43.08**	27.39	29.89	**38.17**	24.91	26.41	**41.21**
50	26.84	28.12	**38.25**	29.96	32.48	**41.60**	26.85	28.56	**37.16**	23.85	25.50	**39.86**

based approximation, the complexity is reduced to linear time $O(mn)$, where m is the number of samples, typically a small fraction of the total number of pixels. At the same time, the experimental results demonstrated that the proposed approach improves over the best existing patch-based methods in terms of both PSNR and subjective visual quality. While this improvement is modest, it is only a starting point, as we have good reason to believe that the improvement in performance brought by the global approach will grow substantially with increasing image size. In Chapter 5, we will present a more detailed analysis of the asymptotic performance of global denoising filters and quantify this gain as a function of image size and the degrees of freedom implied by the the image

content.

4.A MSE analysis of the truncated filter

Each row of the truncated filter can be expressed as:

$$\widetilde{\mathbf{w}}_i^T = \sum_{j=1}^{m} \lambda_j \mathbf{v}_j(i)\mathbf{v}_j^T, \tag{4.17}$$

where $\mathbf{v}_j(i)$ denotes the i-th entry of the j-th eigenvector. Then each estimated pixel $\widehat{z}_i$ has the following form:

$$\widehat{z}_i = \sum_{j=1}^{m} \lambda_j \mathbf{v}_j(i)\mathbf{v}_j^T \mathbf{y}, \tag{4.18}$$

Bias of this estimate can be expressed as[5]:

$$\text{bias}(\widehat{z}_i) = z_i - \text{E}(\widehat{z}_i) = z_i - \sum_{j=1}^{m} \lambda_j \mathbf{v}_j(i) b_j \tag{4.19}$$

where $\mathbf{b} = \mathbf{V}^T\mathbf{z} = [b_1, ..., b_m]^T$ contains the projected signal in the first m modes. The variance term also has the following form:

$$\text{var}(\widehat{z}_i) = \sigma^2(\widetilde{\mathbf{w}}_i^T \widetilde{\mathbf{w}}_i) = \sigma^2(\sum_{j=1}^{m} \lambda_j \mathbf{v}_j(i)\mathbf{v}_j^T)(\sum_{j'=1}^{m} \lambda_{j'} \mathbf{v}_{j'}(i)\mathbf{v}_{j'}) = \sigma^2 \sum_{j=1}^{m} \lambda_j^2 \mathbf{v}_j(i)^2 \tag{4.20}$$

The truncated filter's expected squared error for the i-the pixel is::

$$\begin{aligned} \text{E}^{(m)}[(\widehat{z}_i - z_i)^2] &= \text{bias}(\widehat{z}_i)^2 + \text{var}(\widehat{z}_i) \\ &= (z_i - \sum_{j=1}^{m} \lambda_j \mathbf{v}_j(i) b_j)^2 + \sigma^2 \sum_{j=1}^{m} \lambda_j^2 \mathbf{v}_j(i)^2 \\ &= z_i^2 + \sum_{j=1}^{m} \left(\lambda_j^2(b_j^2 + \sigma^2)\mathbf{v}_j(i)^2 - 2z_i\lambda_j \mathbf{v}_j(i) b_j\right) \end{aligned} \tag{4.21}$$

We can show that the total MSE for the whole image is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \text{E}^{(m)}[(\widehat{z}_i - z_i)^2] = \frac{1}{n}\sum_{i=1}^{n} z_i^2 + \frac{1}{n}\sum_{j=1}^{m} \left((\lambda_j^2 - 2\lambda_j)b_j^2 + \sigma^2\lambda_j^2\right) \tag{4.22}$$

[5]It is worth pointing out that in the truncated space $z_i \neq \sum_{j=1}^{m} \mathbf{v}_j(i)\mathbf{v}_j^T\mathbf{z}$, because $\mathbf{V}\mathbf{V}^T \neq \mathbf{I}$.

where $\|\text{bias}(\widehat{\mathbf{z}})\|^2 = \frac{1}{n}\sum_{i=1}^{n} z_i^2 + \frac{1}{n}\sum_{j=1}^{m}(\lambda_j^2 - 2\lambda_j)b_j^2$ and $\text{var}(\widehat{\mathbf{z}}) = \frac{\sigma^2}{n}\sum_{j=1}^{m}\lambda_j^2$.

Chapter 5

Asymptotic Analysis of the Global Filter

Abstract – This chapter provides an upper bound on the rate of convergence of the mean-squared error for global image denoising, and illustrates that this upper bound decays to zero with increasing image size. Hence, global denoising introduced in Chapter 4 is asymptotically optimal. This property does not hold for patch-based methods such as BM3D, thereby limiting their performance for large images. As observed in practice and shown in this work, this gap in performance is small for moderate size images, but it can grow quickly with image size.

5.1 Introduction

In the previous chapter, we advocated abandoning the explicit use of patches (as done in leading methods such as BM3D) in favor of a global approach where every pixel contributes to the denoising of every other pixel in the image [41]. The similarity of pixels in this approach *can* still be measured using patches, but the application of the filter is truly global. The advantage of this approach is that it is asymptotically optimal

in the sense that its mean-squared-error converges to zero with increasing image size – a property that does not hold for any of the leading patch-based methods [5] – even if the size of the image grows infinitely large, and the range of search for similar patches is allowed to grow as well [46].

It is by now beyond dispute that images (or natural signals generally) contain many redundancies. This notion has been cleverly exploited to design high performance image denoisers with great success. It stands to reason then that a good denoiser should exhibit improved performance as the number of samples (i.e. image size) grows. This concept is not new. In fact, we can go back as far as Shannon who pointed out [47] more than 60 years ago that "*If the source already has a certain redundancy and no attempt is made to eliminate it, a sizable fraction of the letters can be received incorrectly and still [perfectly] reconstructed by the context.*" More recently, this fundamental result was in fact shown for the case of restoring binary images from context in [48,49]. More relevant still, the seminal paper on the Non-Local Means (NLM) method [2] was inspired in part by [48, 49] and itself gave a proof of the asymptotic consistency of the NLM method.

Over time, however, the idea of globally considering the denoising problem was abandoned in favor of more computationally friendly methods that treat patches (or groups of patches) together [5, 19]. Our approach in Chapter 4 relied on a truly global methodology where the effect of every pixel was taken into account to develop a denoiser, and the significant questions of computational complexity were also dealt with by using a subsampling strategy based on the Nyström extension. In the course of that work, we noted that the performance of the global approach consistently improved with image size, but the same was not observed for patch-based methods, hence motivating the work presented in this chapter. Intuitively, in larger images, as the total number

of patches grows, the expected performance improvement due to availability of more overall patches is offset by the lower likelihood of finding closely matching patches (see Fig. 5.1). Increasing the size of the patches reduces the number of available patches, but increases the dimension of the space in which these patches live, hence sometimes providing a helpful effect, but never enough to drive the error to zero asymptotically (see Fig. 5.2). As a result, performance flattens out with increasing image size.

The story is very different (and much more favorable) with global filters. These use *all* the pixels in the input image to denoise every single pixel. Here, we take the analysis a step deeper to prove that the performance of the global approach *always* improves as a function of image size, regardless of image content. Furthermore, we provide a rate for this improvement, and show that this rate is a function of the sparsity of the image in a naturally constructed basis adapted to the content of the image. More specifically, we give an oracle upper bound on the mean-squared-error for estimating each pixel using all the pixels in the image, and show that for typical images, it decays at the rate of *at least* $n^{-\alpha}$ for a $\sqrt{n} \times \sqrt{n}$ image where $\alpha > 0$ depends on the content of the input image [50, 51].

5.2 Computing and Bounding the Oracle Global MSE

The filter $\mathbf{W}$, being data dependent, is of course impacted by the noise in the given image. In practice the filter is never computed directly from the raw, noisy input pixels. Instead, a "pre-filter" is always applied to $\mathbf{y}$ first to reduce the effect of noise, and then the filter weights are computed from this result. When it comes time to the actual filtering, however, this is done using the filter coefficients applied to the original noisy pixels. In the present discussion, since we are interested in the *oracle* performance,

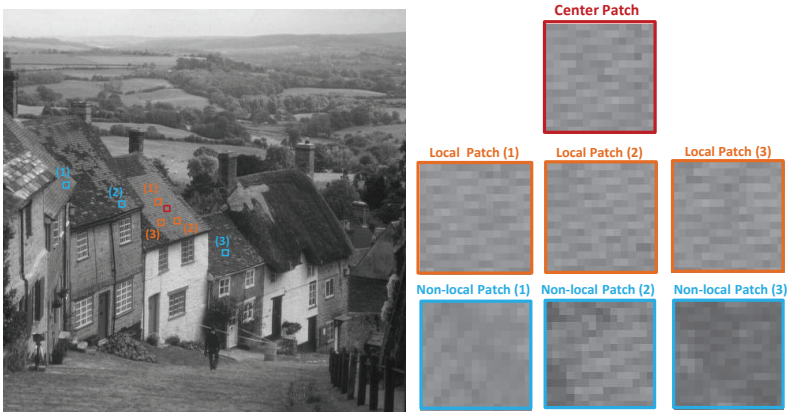


Figure 5.1: Comparison of patch matching for local and non-local patches. Likelihood of finding *closely* similar patches drops as the size of the search window increases.

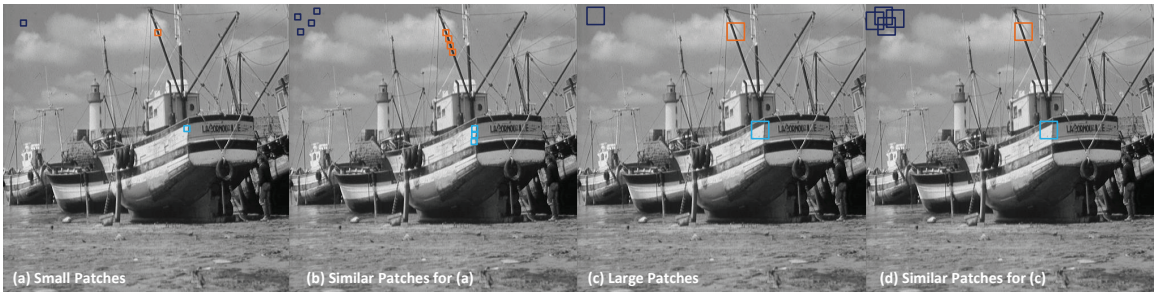


Figure 5.2: Comparison of patch matching for different patch sizes. As the patch size grows, fewer similar patches are available.

we consider the case where the filter is directly computed from the clean latent image $\mathbf{z}$ and is therefore deterministic.

Recall that the filtering framework $\widehat{\mathbf{z}} = \mathbf{W}\mathbf{y} = \mathbf{V}\mathbf{S}\mathbf{V}^T\mathbf{y}$ leads to the estimated closed form MSE as (see Chapter 2):

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(\lambda_j - 1)^2 b_j^2 + \sigma^2\lambda_j^2 \tag{5.1}$$

where $\mathbf{b} = \mathbf{V}^T\mathbf{z} = [b_1, ..., b_n]^T$ contains the coefficients representing the latent image in the global basis, λ_j denotes the filter eigenvalue and the σ^2 is the noise variance.

Minimizing the MSE with respect to the eigenvalues λ_i requires a simple differentiation:

$$\frac{\partial \mathrm{MSE}(\lambda)}{\partial \lambda} = 0 \implies \lambda_j^* = \frac{1}{1 + \mathrm{snr}_j^{-1}}, \tag{5.2}$$

where, somewhat unsurprisingly, the "optimal" eigenvalues $\{\lambda_j^*\}$ are the Wiener coefficients with $\mathrm{snr}_j = \frac{b_j^2}{\sigma^2}$. This shrinkage strategy leads to the minimum value of the MSE[1]:

$$\mathrm{MMSE} = \mathrm{MSE}(\lambda^*) = \frac{\sigma^2}{n} \sum_{j=1}^{n} \lambda_j^* \tag{5.3}$$

Fig. 5.3 depicts Wiener shrinkage factors of some test images shown in Fig. 5.7. Evidently, stationary images with repetitive patterns such as Wall show a faster decay rate of λ_j^*. This indicates that sparse signals (in the basis defined by the filter) are easier to recover from additive white noise. In the case of stationary signals, as the image size grows the decay rate of the Wiener shrinkage factors increases. Assuming that the shrinkage factors decay in some fashion, the minimum MSE given in (5.3) can be bounded (see Section 5.2.1). Furthermore, by clustering pixels into relatively stationary subimages, this condition can be eased and a more generic denoising bound will be obtained (see Section 5.2.3).

5.2.1 Bounding the Oracle MSE of Stationary Images

Expanding the minimum MSE given by (5.3):

$$\mathrm{MMSE} = \frac{\sigma^2}{n} \sum_{j=1}^{n} \lambda_j^* = \frac{1}{n} \sum_{j=1}^{n} \frac{\sigma^2 b_j^2}{\sigma^2 + b_j^2} \tag{5.4}$$

[1]Since the equivalent shrunk filter should be kept doubly-stochastic, λ_1^* should in theory be 1; a constraint which increases the minimum MSE. This MSE increment is $\Delta\mathrm{MSE} = \frac{\sigma^2}{n}(1 - \lambda_1^*) = \frac{\sigma^4}{n(b_1^2 + \sigma^2)}$. Having the first eigenvector $\mathbf{v}_1 = \frac{1}{\sqrt{n}}\mathbf{1}_n$, the squared signal projection coefficient can be expressed as $b_1^2 = \frac{1}{n}(\sum_{i=1}^{n} z_i)^2$. Practically, for a moderate size image in the range of [0,255], $\Delta\mathrm{MSE}$ is very small and can be neglected (or equivalently $\lambda_1^* \approx 1$). Consequently, it is not necessary to impose $\lambda_1^* = 1$ as a constraint in our analysis.

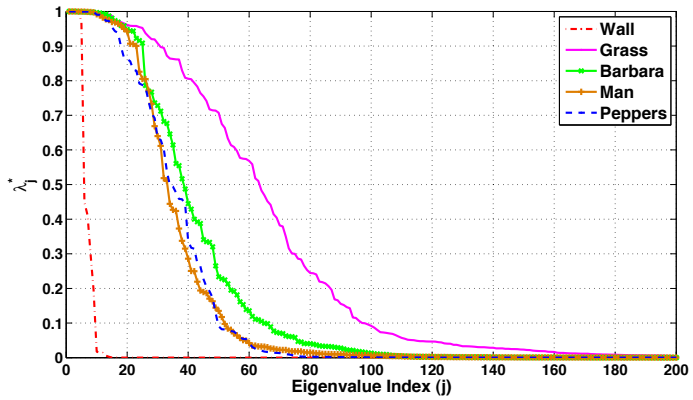


Figure 5.3: Wiener shrinkage eigenvalues (λ_j^*) computed for some test images shown in Fig. 5.7. Images with repetitive patterns such as Wall represent fast decaying Wiener coefficients. On the contrary, the optimal shrinkage factors of non-stationary images (e.g. Grass) drop off in a slow fashion.

The last equality can be expressed as:

$$\text{MMSE} = \frac{1}{2n}\sum_{j=1}^{n}\frac{\sigma|b_j|}{\frac{\sigma^2+b_j^2}{2}}\sigma|b_j| \tag{5.5}$$

Using the Arithmetic-Geometric means inequality [52] we have $\sigma|b_j| \leq \frac{\sigma^2+b_j^2}{2}$, which implies:

$$\text{MMSE} = \frac{1}{2n}\sum_{j=1}^{n}\frac{\sigma|b_j|}{\frac{\sigma^2+b_j^2}{2}}\sigma|b_j| \leq \frac{1}{2n}\sum_{j=1}^{n}\sigma|b_j| \tag{5.6}$$

And this in turn means

$$\text{MMSE} \leq \frac{\sigma}{2n}\|\mathbf{b}\|_1 \tag{5.7}$$

The oracle MSE is evidently bounded by the l_1 norm of the projection coefficients $\mathbf{b}$. This implies that for a given n, the more sparse the signal is in the basis given by the filter kernel, the smaller the MSE error will be. Furthermore, for a signal (image) with finite energy, the 1-norm of $\mathbf{b}$ can not grow faster than n with increasing dimension, so the upper bound must collapse to zero asymptotically. Let's consider the worst case pathology wherein $|b_j| = c$ (a constant), resulting in linear grown of

$\|\mathbf{b}\|_1$ with n. This essentially corresponds to the signal being "white noise" in the basis defined by the kernel. In this worst case scenario, the MMSE is upper bounded by a constant $\frac{c\sigma}{2}$. In general, however, we expect the coefficients to drop off at some rate, say $\alpha > 0$. That is, $|b_j| = \frac{c}{j^\alpha}$, which implies that

$$\frac{\sigma}{2n}\sum_{j=1}^{n}|b_j| = \frac{\sigma}{2n}\sum_{j=1}^{n}\frac{c}{j^\alpha} \tag{5.8}$$

As $n \to \infty$, MMSE will tend to zero for all $\alpha > 0$, so this establishes the most general (stationary) case of MSE convergence. Now let's have a look at the rate of convergence in more detail. For this purpose, it is useful to consider the coefficients b_j as samples of the function $|b(t)| = c/t^\alpha$. That is, define $b_j = b(j)$.

Using the integral test for convergence (Maclaurlin-Cauchy test) [53], we have the following lower and upper bound:

$$\frac{1}{n}\int_1^{n+1}\frac{c}{t^\alpha}\,dt \le \frac{1}{n}\sum_{j=1}^{n}|b_j| \le \frac{1}{n}(c + \int_1^{n}\frac{c}{t^\alpha}\,dt) \tag{5.9}$$

For $0 < \alpha < 1$ we have:

$$c(\frac{(n+1)^{1-\alpha}-1}{(1-\alpha)n}) \le \frac{1}{n}\sum_{j=1}^{n}\frac{c}{j^\alpha} \le c(\frac{n^{1-\alpha}-\alpha}{(1-\alpha)n}) \tag{5.10}$$

which means a convergence rate of $O(n^{-\alpha})$.

On the other hand, for $\alpha = 1$ we have:

$$\frac{c\ln(n+1)}{n} \le \frac{1}{n}\sum_{j=1}^{n}\frac{c}{j} \le \frac{c(1+\ln(n))}{n} \tag{5.11}$$

which indicates a rate of $O(n^{-1}\ln(n))$. Finally, the decay rate is $O(n^{-1})$ for $\alpha > 1$ since the summation in (5.8) converges to a finite constant. In summary, as long as the coefficients decay at all, *at whatever rate*, the minimum MSE is guaranteed to approach zero. Of course in the case of stationary images this decay rate is guaranteed to be

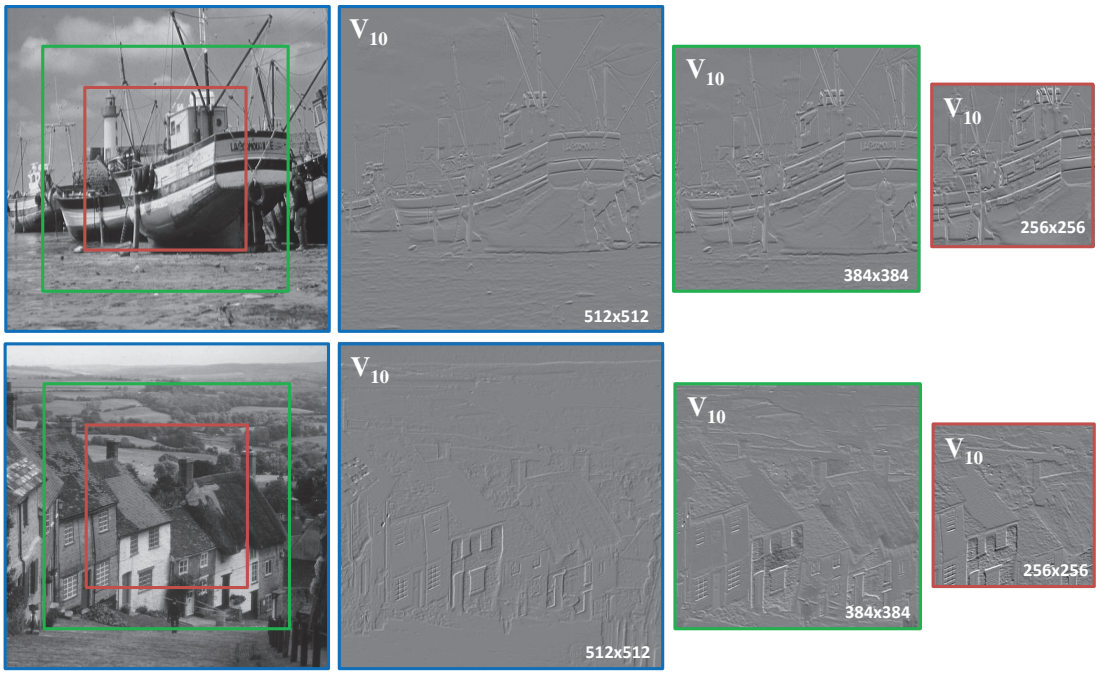


Figure 5.4: Sample eigenvectors computed from image windows of different sizes. Top: Boat image, Bottom: Goldhill image. The 10-th eigenvectors ($\mathbf{v}_{10}$) of the three subimages are illustrated.

fast. Yet, in the case of natural images, one might argue that the drop off rate could be hamstrung by image size increment. In other words, as the image size grows, the MSE bound computed for non-stationary images could be increasing. This is due to the evolving filter eigenvectors, which affect the projection coefficients b_j (or equivalently the Wiener shrinkage factors λ_j^*). We address this issue next.

5.2.2 Filter Eigenvectors

Some eigenvector examples computed from image windows of size 256×256, 384×384 and 512×512 are shown in Fig. 5.4. As the image size grows, the 256×256 eigenvector window may remain visually unchanged (such as Boat image in Fig. 5.4), or

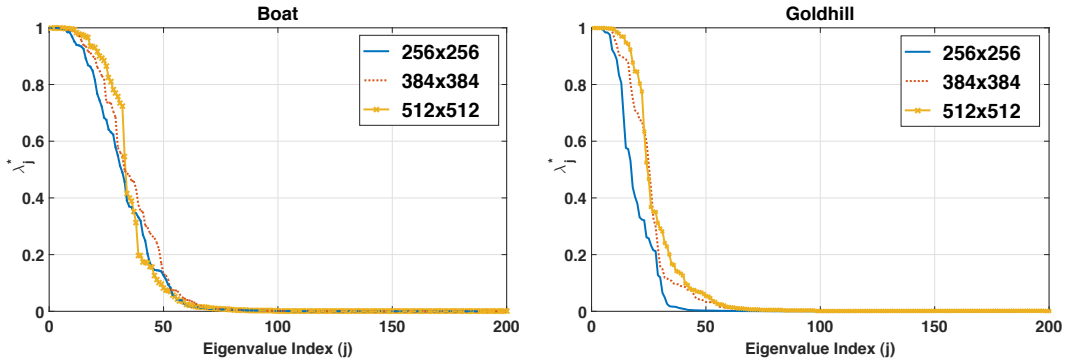


Figure 5.5: Wiener shrinkage factors (λ_j^*) of the global filter computed for image windows of different sizes shown in Fig. 5.4.

may alter across the three eigenvectors (such as Goldhill image in Fig. 5.4). Increasing the window size may introduce new content to the image, which may lead to changing eigenvectors. Fig. 5.5 shows that the image size growth could affect the drop-off rate of the Wiener shrinkage factors (λ_j^*). The shrinkage factors illustrated in Fig. 5.5 are computed for the image windows given in Fig. 5.4. This change in the decay rate of the λ_j^* coefficients has direct impact on the estimated bound. To tackle the problem of evolving eigenvectors, we propose clustering pixels, such that each cluster filter deals with relatively stationary subimages. We demonstrate this in the following section.

5.2.3 Bounding the Oracle MSE of Generic Images

For pixel grouping, the concept of diffusion maps [54] is used; wherein each pixel located at position $\mathbf{x}_i$ is mapped into a manifold defined by the weighted eigenvectors as:

$$\mathbf{\Psi}_t(\mathbf{x}_i) = (\lambda_2^t v_{i2}, \lambda_3^t v_{i3}, ..., \lambda_m^t v_{im}) \tag{5.12}$$

where v_{im} denotes the i-th entry of the m-th eigenvector and t represents the diffusion parameter. These descriptors are fed into the k-means classifier to obtain the clustering map of p clusters shown in Fig. 5.6 (in our experiments t and p are set to 1 and 5, respectively). Our descriptors are then computed from the filter eigen-decomposition, leading to clustering similar pixels together.

Now we are ready to apply a denoising filter to each cluster separately and compute the overall MSE bound. Fig. 5.6 shows the benefit of grouping similar pixels over global filter. As can be seen, in comparison to the global filter, the Wiener shrinkage factors decay more rapidly for the similar pixels in each cluster.

The overall minimum MSE bound can be expressed as:

$$\begin{aligned} \text{MMSE} &\leq \frac{\sigma}{2n}\|\mathbf{b}_1\|_1 + \cdots + \frac{\sigma}{2n}\|\mathbf{b}_p\|_1 \\ &\leq \frac{\sigma}{2n}\sum_{l=1}^{p}\|\mathbf{b}_l\|_1 \end{aligned} \tag{5.13}$$

where p denotes the number of clusters and $\mathbf{b}_l$ represents the projection coefficients of the l-th cluster onto the respective basis. With $p = 1$, the bound will be given by the expression in (5.7). As the image size grows, assuming that each newly added pixel falls at worst, into one unique cluster of size 1 (meaning that $p = n$), the MMSE will be bounded by $\frac{\sigma\ max(\|\mathbf{b}_1\|_1,\ldots,\|\mathbf{b}_n\|_1)}{2}$. In practice, however, the number of clusters is much smaller than the number of pixels ($p << n$). When each subimage is relatively stationary, the overall decay rate of the MSE bound is determined by the minimum drop-off rate of $|\mathbf{b}_l|$ for $l = 1, \cdots, p$. In other words, as the image size grows, and assuming that no new cluster is added, the overall decay rate of the MSE bound is governed by the slowest term in (5.13). Next, we illustrate these results with some experiments.

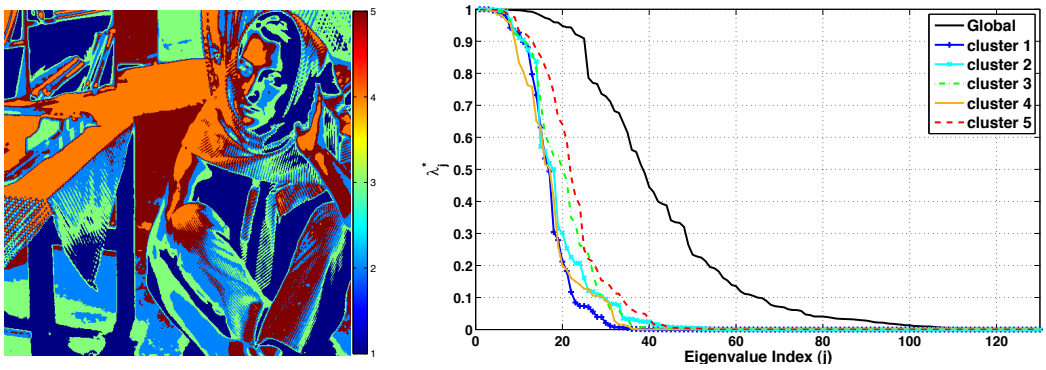
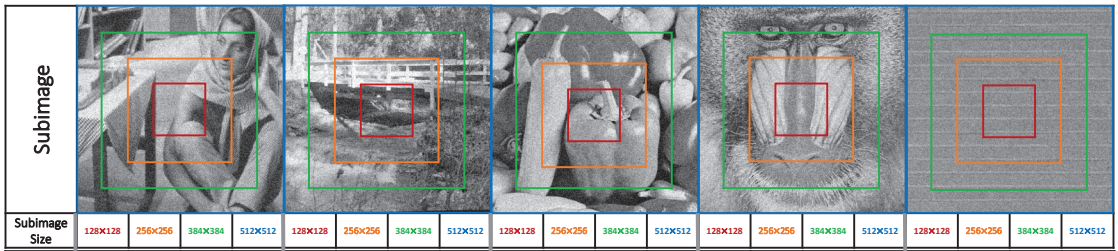


Figure 5.6: Left: clustering map, Right: Wiener shrinkage factors (λ_j^*) of the global and clustered filters. The shrinkage coefficients of the clustered pixels show faster decay rate compared to the global filter.

Figure 5.7: Some benchmark images used to evaluate performance of our denoising method.

5.3 Results and Comparisons

Some benchmark images used in this section are shown in Fig. 5.7. The effect of image size on denoising performance is explored in the first set of experiments in Fig. 5.8. For this experiment we denoised the central part of an image with increasing size. As can be seen, the increment in the number of pixels consistently leads to lower MSE.



Subimage Size	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512
$\sigma = 20$	1.82	0.57	0.27	**0.17**	1.15	0.34	0.17	**0.09**	1.05	0.73	0.14	**0.11**	1.04	0.66	0.42	**0.21**	0.93	0.25	0.18	**0.06**
$\sigma = 40$	4.19	1.49	0.76	**0.52**	9.77	1.78	0.74	**0.50**	3.44	1.24	0.42	**0.29**	5.13	2.49	0.88	**0.59**	2.74	0.73	0.42	**0.24**
$\sigma = 60$	7.18	2.73	1.44	**1.01**	11.40	3.38	1.35	**0.93**	6.80	2.12	0.81	**0.67**	7.87	5.08	1.57	**1.13**	5.42	1.43	0.87	**0.54**
Subimage Size	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512	128×128	256×256	384×384	512×512
$\sigma = 20$	1.39	0.56	0.33	**0.22**	6.49	0.70	0.28	**0.12**	3.32	0.64	0.29	**0.17**	1.57	0.69	0.23	**0.09**	4.28	1.73	0.98	**0.71**
$\sigma = 40$	3.79	1.58	0.91	**0.47**	5.46	1.51	0.77	**0.33**	4.64	0.99	0.49	**0.28**	3.85	1.72	0.72	**0.21**	8.19	3.25	1.66	**1.41**
$\sigma = 60$	6.84	2.94	1.70	**1.04**	9.90	2.56	1.43	**0.65**	5.12	1.93	0.93	**0.48**	6.56	3.32	1.22	**0.51**	14.24	4.22	2.73	**2.61**

Figure 5.8: Oracle performance of the global denoising scheme for different window sizes. MSE values are averaged over 20 independent WGN realizations and k-means initialization points.

It is also important to highlight that the MSE values of the full size image (512×512) are very small and below round off error.

The oracle MSE for the images in Fig. 5.8 are shown in Fig. 5.9 where for each noise level, the bound in (5.13) and the oracle MSE values are computed and then averaged across images given in Fig.[2] 5.7. Each experiment is repeated and averaged for 20 independent noise realizations and 20 different initializations of the k-means clustering.

[2]We note that for practical purposes, our experiments are carried out using a truncated filter with only a small percentage of the leading eigenvectors of $\mathbf{W}$. Still, the averaged MSEs in Fig. 5.9 capture the decay rate for the tested images as hypothesized. See Appendix 5.A.

Table 5.1: Estimated decay rate of the oracle MSE and estimated bound obtained from test images corrupted by WGN with $\sigma = 30$. By using a least square approach, $\frac{\gamma}{n^{\alpha}}$ is fitted on the data points of the MSE and estimated Bound.

Test Images	**Peppers**	**Man**	**Stream**	**Lena**	**Wall**
α_{MSE}	1.32 ($\pm$0.19)	0.98 ($\pm$0.15)	0.96 ($\pm$0.11)	0.94 ($\pm$0.08)	0.89 ($\pm$0.12)
α_{Bound}	0.57 ($\pm$0.05)	0.55 ($\pm$0.04)	0.52 ($\pm$0.05)	0.51 ($\pm$0.03)	0.50 ($\pm$0.01)
Test Images	**Barbara**	**Goldhill**	**Boat**	**Mandrill**	**Grass**
α_{MSE}	0.79 ($\pm$0.04)	0.75 ($\pm$0.07)	0.71 ($\pm$0.13)	0.58 ($\pm$0.10)	0.55 ($\pm$0.16)
α_{Bound}	0.50 ($\pm$0.05)	0.48 ($\pm$0.03)	0.49 ($\pm$0.05)	0.44 ($\pm$0.02)	0.45 ($\pm$0.07)

In this example, the function $\frac{\gamma}{n^{\alpha}}$ is fitted on both bound and MSE data points using a least square approach where n denotes the image size and γ is a constant. Table 5.1 represents the estimated α values for our test images. The estimated α_{MSE} values indicate the decay rate of MSE with respect to image size. As can be seen, stochastic textures such as Mandrill and Grass show smaller α_{MSE}. Although the estimated α_{Bound} does not match α_{MSE} (because (5.13) is an upper bound), relative ordering of the decay rates are preserved. The estimated bound is also depicted individually for each test image in Fig. 5.10. As can be seen, the proposed bound nearly captures the decay rate of the MSE function. The oracle performance of NLM [2], BM3D and our approach are compared in Table 5.2. BM3D is a two-stage image denoising scheme in which the first stage, as a pre-filter, provides a "pilot" estimate of the noise free image. The second stage uses the pre-filtered image to obtain the *near* optimal Wiener shrinkage using an estimate of the SNR and also to perform a more accurate patch matching. In other words, in the oracle BM3D the output of the first stage is *assumed* to be the clean image. As the noise level increases, results in Table 5.2 suggest that the gap between our oracle scheme and oracle BM3D grows monotonically.

Summary – We emphasize that the oracle results do not correspond to practical denoising algorithms yet, and practical realization of the global scheme remain to be

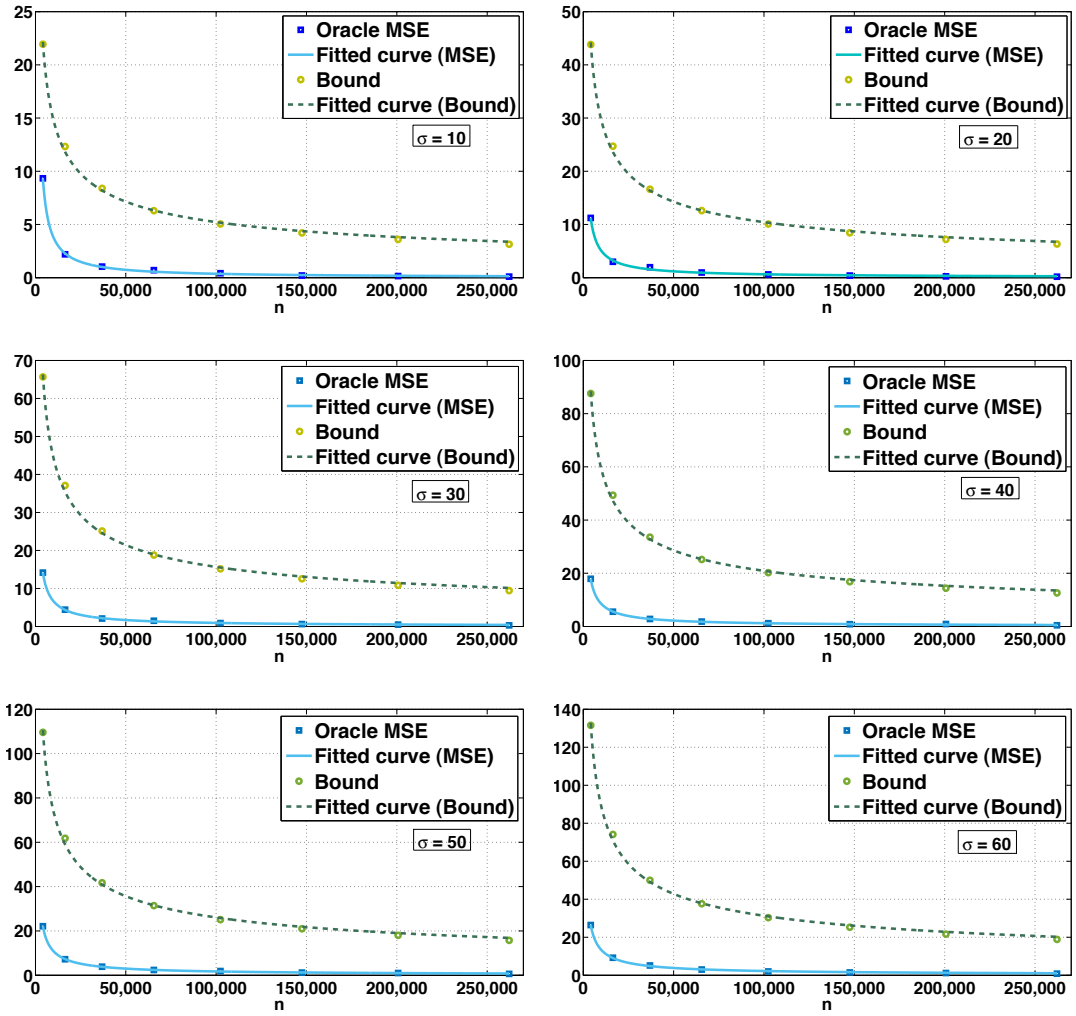


Figure 5.9: Averaged MSE of denoising images given in Fig. 5.7 for different noise levels. The estimated bound given in (5.13) is averaged across all the images.

studied. However, global filtering has an interesting asymptotic behavior that surpasses the existing patch-based bounds by a large margin. The oracle MSE values for the global filter converge to perfect reconstruction of the clean image, which is apparently impossible to achieve for oracle versions of algorithms such as BM3D. This implies that global filtering is promising as a way to see how much farther practical algorithms can be pushed.

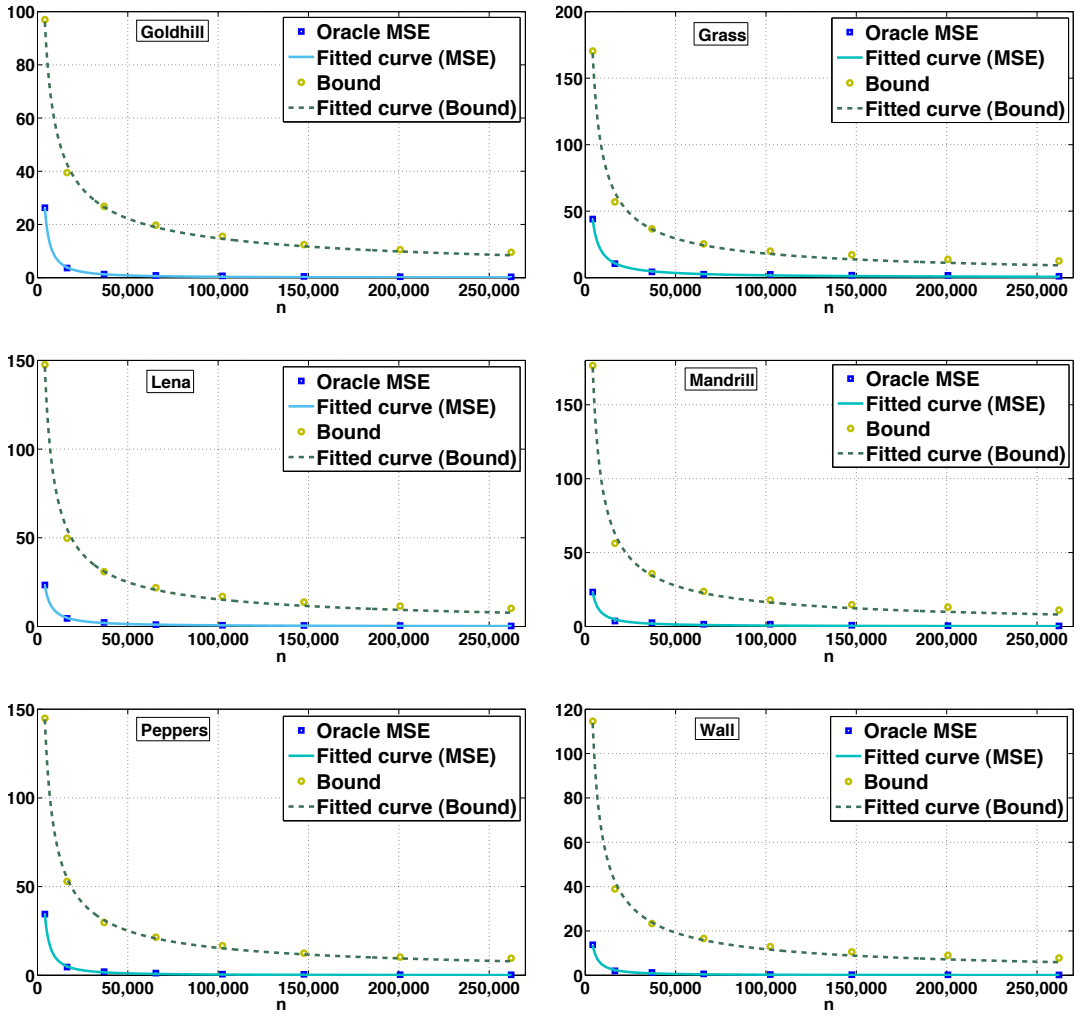


Figure 5.10: Fitted curves of MSE and the estimated bound for some test images corrupted by WGN of $\sigma = 30$. The decay rate of the fitted curves are given in Table 5.1.

5.A The Truncated Filter and Its MSE analysis

Keeping m leading eigenvectors of the filter $\mathbf{W}$, the total MSE of the truncated filter can be expressed as (proved in Chapter 4):

$$\mathrm{MSE}(m) = \frac{1}{n}\sum_{i=1}^{n} z_i^2 + \frac{1}{n}\sum_{j=1}^{m}\left((\lambda_j^2 - 2\lambda_j)b_j^2 + \sigma^2\lambda_j^2\right) \tag{5.14}$$

Table 5.2: Oracle MSE values of NLM [2] (1st column), oracle BM3D [5] (2nd column), and Ours (3rd column). The MSE values are averaged over 20 independent noise realizations for each σ.

σ	Barbara			Stream			Peppers			Mandrill			Wall		
	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours
10	29.36	11.34	**0.08**	62.95	26.93	**0.05**	21.87	11.41	**0.06**	70.97	31.32	**0.02**	18.91	8.02	**0.03**
20	42.15	22.98	**0.17**	120.77	63.42	**0.09**	33.43	20.81	**0.11**	166.23	76.30	**0.06**	28.72	16.23	**0.05**
30	61.91	34.11	**0.29**	171.91	98.39	**0.29**	45.18	28.53	**0.38**	196.61	121.54	**0.16**	39.33	24.52	**0.09**
40	91.74	45.14	**0.52**	233.03	130.44	**0.50**	58.84	35.83	**0.59**	275.49	164.95	**0.24**	56.27	31.26	**0.15**
50	126.43	59.80	**0.73**	285.12	158.55	**0.81**	75.23	48.33	**0.86**	391.81	188.16	**0.42**	70.42	41.91	**0.20**
60	163.33	71.26	**1.01**	329.89	184.09	**0.93**	93.52	56.87	**1.13**	431.36	220.80	**0.54**	87.43	48.92	**0.27**
σ	Boat			Man			Goldhill			Lena			Grass		
	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours	NLM	BM3D	Ours
10	31.87	14.30	**0.09**	36.25	16.26	**0.06**	31.16	14.93	**0.10**	19.77	9.71	**0.02**	68.35	30.04	**0.02**
20	56.96	29.42	**0.22**	67.85	35.59	**0.12**	53.47	29.72	**0.17**	31.07	18.34	**0.09**	164.01	75.72	**0.05**
30	80.44	43.16	**0.31**	94.89	53.72	**0.19**	78.67	42.06	**0.23**	44.68	26.28	**0.16**	191.16	120.56	**0.15**
40	108.72	56.16	**0.47**	125.95	70.61	**0.33**	105.51	52.91	**0.28**	60.46	34.13	**0.21**	270.90	160.11	**0.24**
50	138.42	75.05	**0.74**	158.01	93.27	**0.51**	131.17	69.78	**0.39**	76.56	45.22	**0.39**	390.17	186.94	**0.41**
60	168.69	88.32	**1.04**	188.98	109.14	**0.65**	155.07	80.54	**0.48**	92.94	53.70	**0.51**	420.11	211.56	**0.59**

where $\|\text{bias}(\widehat{\mathbf{z}})\|^2 = \sum_{i=1}^{n} z_i^2 + \sum_{j=1}^{m}(\lambda_j^2 - 2\lambda_j)b_j^2$ and $\text{var}(\widehat{\mathbf{z}}) = \sigma^2 \sum_{j=1}^{m} \lambda_j^2$. Truncation results in larger bias and smaller variance as shown by the expressions.

The optimal eigenvalues are again the Wiener shrinkage coefficients (λ_j^*) which lead to the minimum MSE as:

$$\text{MMSE}(m) = \frac{1}{n}\sum_{i=1}^{n} z_i^2 - \frac{1}{n}\sum_{j=1}^{m} b_j^2 \lambda_j^* \tag{5.15}$$

Of course this MSE will necessary be larger than the case where all the eigenvectors are used. Replacing $\frac{1}{n}\sum_{i=1}^{n} z_i^2$ with $\frac{1}{n}\sum_{j=1}^{n} b_j^2$ and after some simplifications:

$$\begin{aligned}
\text{MMSE}(m) &= \frac{1}{n}\sum_{j=1}^{n} b_j^2 - \frac{1}{n}\sum_{j=1}^{m} b_j^2 \lambda_j^* \\
&= \frac{1}{n}\sum_{j=1}^{n} b_j^2 - \frac{1}{n}\sum_{j=1}^{n} b_j^2 \lambda_j^* + \frac{1}{n}\sum_{j=m+1}^{n} b_j^2 \lambda_j^* \\
&= \underbrace{\frac{1}{n}\sum_{j=1}^{n} \frac{\sigma^2 b_j^2}{\sigma^2 + b_j^2}}_{\text{MMSE}} + \underbrace{\frac{1}{n}\sum_{j=m+1}^{n} \frac{b_j^4}{b_j^2 + \sigma^2}}_{\Delta\text{MSE}(m)}
\end{aligned} \tag{5.16}$$

where the first term is the same as the minimum MSE given in (5.4) and $\Delta\text{MSE}(m)$

denotes the filter truncation effect on the MSE. We can show that the added MSE term is bounded and consequently, the MMSE(m) will be upper bounded.

We start with an upper bound on ΔMSE(m):

$$\Delta\text{MSE}(m) = \frac{1}{n}\sum_{j=m+1}^{n}\frac{b_j^4}{b_j^2+\sigma^2} \leq \frac{1}{\sigma^2 n}\sum_{j=m+1}^{n} b_j^4 = \frac{1}{\sigma^2 n}(\|\mathbf{b}\|_4^4 - \|\mathbf{b}_m\|_4^4) \tag{5.17}$$

where [3] $\|\mathbf{b}_m\|_4^4 = \sum_{j=1}^{m} b_j^4$. Again, assuming a decay rate of $\alpha > 0$ for the coefficients $|b(t)| = c/t^\alpha$ and using the integral test for convergence [53], we obtain the following lower and upper bound:

$$\frac{1}{\sigma^2 n}\int_{m+1}^{n+1}\frac{c^4}{t^{4\alpha}}\,dt \leq \frac{1}{\sigma^2 n}\sum_{j=m+1}^{n} b_j^4 \leq \frac{1}{\sigma^2 n}(\frac{c^4}{(m+1)^{4\alpha}} + \int_{m+1}^{n}\frac{c^4}{t^{4\alpha}}\,dt) \tag{5.18}$$

For $0 < \alpha < \frac{1}{4}$ we have:

$$\begin{aligned}\frac{c^4}{\sigma^2}\left(\frac{(n+1)^{1-4\alpha}-(m+1)^{1-4\alpha}}{(1-4\alpha)n}\right) &\leq \frac{1}{\sigma^2 n}\sum_{j=m+1}^{n}\frac{c^4}{j^{4\alpha}} \\ &\leq \frac{c^4}{\sigma^2}\left(\frac{1}{n(m+1)^{4\alpha}} + (\frac{n^{1-4\alpha}-(m+1)^{1-4\alpha}}{(1-4\alpha)n})\right)\end{aligned} \tag{5.19}$$

where a convergence rate of $O(n^{-4\alpha})$ is guaranteed. For $\alpha = \frac{1}{4}$ we have:

$$\frac{c^4}{\sigma^2}(\frac{\ln(\frac{n+1}{m+1})}{n}) \leq \frac{1}{\sigma^2 n}\sum_{j=m+1}^{n}\frac{c^4}{j^4} \leq \frac{c^4}{\sigma^2}(\frac{1}{n(m+1)} + \frac{\ln(\frac{n}{m+1})}{n}) \tag{5.20}$$

which means a decay rate of $O(n^{-1}\ln(n))$. Similar to our previous analysis, for $\alpha > \frac{1}{4}$ the decay rate is $O(n^{-1})$. Overall, the minimum MSE of denoising by the truncated filter has the following bound:

$$\text{MMSE}(m) \leq \frac{\sigma}{2n}\|\mathbf{b}\|_1 + \frac{1}{\sigma^2 n}(\|\mathbf{b}\|_4^4 - \|\mathbf{b}_m\|_4^4) \tag{5.21}$$

It is not hard to see that even when m is kept fixed and n tends to infinity, a sufficiently fast decay of the coefficients $\mathbf{b}$ will still yield an asymptotic MSE of zero. In our

[3] In practice, for $j > m$ we have $b_j^2 \ll \sigma^2$. This means that $\frac{b_j^4}{b_j^2+\sigma^2} \leq \frac{b_j^4}{\sigma^2} \leq b_j^2$.

experiments the ratio between m and n is kept fixed as $\frac{m}{n} = \frac{1}{1000}$, which leads to $m \approx 250$ for a test image of size 512×512.

Chapter 6

Global Image Editing

Abstract – In this chapter we introduces a new image editing tool, based on the spectrum of a global filter computed from image affinities. In Chapter 3, it was shown that the global filter derived from a fully connected graph representing the image, can be approximated using the Nyström extension. This filter is computed by approximating the leading eigenvectors of the filter. These orthonormal eigenfunctions are highly expressive of the coarse and fine details in the underlying image, where each eigenvector can be interpreted as one scale of a data-dependent multiscale image decomposition. In this filtering scheme, each eigenvalue can boost or suppress the corresponding signal component in each scale. Our analysis shows that the mapping of the eigenvalues by an appropriate polynomial function endows the filter with a number of important capabilities, such as edge-aware sharpening, denoising, tone manipulation and abstraction, to name a few. Furthermore, the edits can be easily propagated across the image.

6.1 Introduction

Edge-aware image filtering is a key tool in image processing, computer vision and graphics. In most existing methods the underlying image is first decomposed into piecewise smooth and detail layers. Then, a variety of capabilities, such as tone mapping, edge editing and edit propagation are developed based on this type of decomposition [55–59].

The ideal edge-aware filter coarsens details of the image, yet the principal edges are not altered. Given the difficulty in determining what should qualify as an edge, and which edge should be preserved or smoothed, designing such a filter is quite challenging. Many smoothing operators have been proposed in the past few years, and anisotropic diffusion is perhaps one of the most well-known methods [60]. Anisotropic diffusion tends to preserve (and even sharpen) main edges while smoothing the texture regions. However, the iterative nature of this filter can make the computational burden quite heavy. Also, the ideal iteration number for each region of the image could be different; yet, an infinite iteration number can produce a constant image.

Several non-linear (data-dependent) operators such as the bilateral filter [1,61] have been used for the same task. Chen et al. [62] used the bilateral filter, while progressively incrementing the spatial and range width of the Gaussian for building a pyramid of image layers. In a similar iterative approach [63], the bilateral filter is applied successively on the coarsened image while decreasing the range width. In all of these methods edges are preserved by the gradual change in the tuning parameters of the bilateral kernel. However, the kernel weights have to be recomputed in every iteration.

Almost all existing edge-aware methods use the same general idea: *Using a*

local operator, they decompose the image into a base layer and a detail layer, and then manipulate each layer and recombine to reach the desired edit. There are two main problems with this approach:

- Since noise is always an unavoidable part of any imaging system, boosting the detail layer usually worsens the signal-to-noise-ratio (SNR). Even with today's mega-pixel imagers, the trade off between sharpness and SNR is still a bottleneck. Increasing exposure time will result in higher SNR, but a more blurry image. On the other hand, a short exposure leads to sharper but noisier images.

- While it is often desirable to treat similar edges of an image in the same way, the existing local filters have irregular behaviors when handling edges with slightly different brightness and gradient profiles. In other words, global structure common among similar edges are usually ignored by the low-level feature vectors associated with each pixel. Even with all the edge-aware operators in hand, performing local adjustments to pixels and then *evenly* propagating the edit to the similar regions *all* across the image has proved to be challenging.

To alleviate the first problem, some methods have been proposed that build on classic linear unsharp masking. Adaptive unsharp masking [6] controls contrast enhancement in texture areas and avoids noise magnification by leaving relatively smooth regions unchanged. A hierarchical framework based on Non-Local Means (NLM) kernel [2] is proposed in [64] where noise removal is applied first as a separate step and then the detail layers are extracted. Another technique in [65] applies an offset to the bilateral filter to change its usual coarsening behavior to sharpening. This adaptive bilateral filter sharpens an image by increasing the slope of edges; however, its sharp-

ening strength is limited. Recently a new restoration method for handling mild blur and strong noise has been introduced in [66], where using the steering kernel regression technique [3] both denoising and sharpening are combined in one framework.

To mitigate the second problem, there have been some efforts to interactively propagate the edits to regions with similar appearance. Recently, the sparse optimization formulation is used to provide stroke-based editing workflows with propagative tonal adjustments [59, 67, 68]. Using an edge-aware energy minimization method, the tonal adjustment imposed by the user is interpolated to the pixels with similar luminance. Farbman et al. [58] also proposed an edit propagation method based on the concept of *diffusion distances* which can measure closeness of pixels on a manifold space. By approximating a diffusion map built upon this high-dimensional similarity measure, the input adjustments can propagate to nearby pixels on the manifold.

In our framework, the two above-mentioned shortcomings of the existing methods are tackled at the same time. Our image filter is global in the sense that all the node (pixel) pairs on the graph (image) are directly connected to each other. As shown in Chapter 3, the eigen-decomposition of the corresponding symmetric, doubly-stochastic filter matrix can be approximated using the Nyström extension. The obtained eigenvectors are very informative of the similar regions and edge information of the image[1] (Fig. 6.1). More specifically, the approximated eigenfunctions enable us to employ diffusion distance for propagating the same manipulation over pixels belonging to similar regions globally. Having the spectrum of the filter, simultaneous noise suppression and detail enhancement become much easier by mapping the spectrum of the filter using a polynomial function, with a few parameters to tune. Our experimental results show

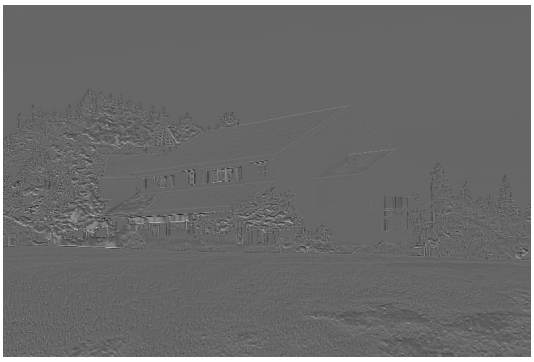
[1]Supplementary materials, including MATLAB code, additional experimental results and manuscript figures are available at the project website: http://www.soe.ucsc.edu/~htalebi/NLEditing.html

(a) House

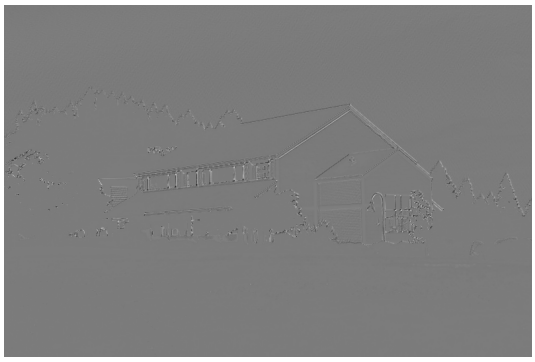
(b) $\mathbf{v}_2(\lambda_2 = 0.9917)$

(c) $\mathbf{v}_3(\lambda_3 = 0.9045)$

(d) $\mathbf{v}_{10}(\lambda_{10} = 0.2406)$



(e) $\mathbf{v}_{25}(\lambda_{23} = 0.0508)$



(f) $\mathbf{v}_{50}(\lambda_{50} = 0.0025)$

Figure 6.1: Some leading eigenvectors computed from the luminance channel of the house image using 0.01% of the pixels.

that this strategy reduces the halo artifacts around principal edges, avoids the common noise magnification problem, and can interactively propagate the user's edit across the intended similar regions with ease. In contrast to [58], our approach does not require the solution of a complex optimization problem to achieve this effect.

Our contributions are as follows: *First*, our framework handles noise naturally, because the image is projected into the data-adapted basis obtained from affinity weights. In other words, the noise is naturally separated from the underlying signal components by projecting the image into the approximated leading eigenvectors. *Second*, our global framework is better than the existing propagation approaches where the global affinity approximation is used just as a guide mask. In addition to providing such a mask, the proposed scheme is able to deliver a filtering tool with many capabilities [69, 70].

6.1.1 Non-local Affinities

As discussed in previous chapters, our filtering framework is based on non-parametric regression in which a kernel function K_{ij} measures the similarity between the samples y_i and y_j, located at $\mathbf{x}_i$ and $\mathbf{x}_j$ coordinates, respectively. Recalling the NLM kernel [2] in which the photometric similarity is captured in a patch-wise manner:

$$K_{ij} = \exp\left\{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right\} \tag{6.1}$$

The spatial and photometric smoothing parameters of the kernel (h_x and h_y in (6.1)) can affect the rank of the kernel matrix $\mathbf{K}$. Specially small h_x makes the kernel matrix $\mathbf{K}$ high-rank (more diagonally dominant), and as a result, the Nyström extension will need more samples for an accurate reconstruction of the eigenvectors of the filter matrix. However, we observed that this approximation of the filter matrix $\mathbf{W}_m$ also has

some nice properties. To illustrate, the leading eigenvectors of Fig. 6.1(a) are computed for the kernel with small spatial smoothing parameter in Fig. 6.2. Comparing these eigenvectors to the ones presented in Fig. 6.1, it can be seen that the spatial term forces the eigenvectors to become piecewise smooth. As we will discuss in Section 6.4, these eigenfunctions force the filter to become a local smoother. In this chapter, h_x is assumed to be very large (practically NLM kernel without the spatial term) for the purpose of detail manipulation, and fixed as 20 for other applications. Based on image content, we also optimize h_y to minimize the filter approximation error (See Appendix 5.A).

In what follows, our detail manipulation strategy is explained in Section 6.2. Next, our propagation mask and experimental results for different applications are described.

6.2 Eigenvalue Mapping Function

The filtered image $\widehat{\mathbf{y}}$ can be expressed as:

$$\widehat{\mathbf{y}} = f(\mathbf{W})\mathbf{y} \tag{6.2}$$

where in general, $f(\mathbf{W})$ denotes a matrix function. Analogously to scalar analytic functions, matrix functions of an $n \times n$ square matrix $\mathbf{W}$ can be defined using a power series:

$$f(\mathbf{W}) = \sum_{i=0}^{\infty} c_i \mathbf{W}^i \tag{6.3}$$

The above series exists and is finite for a given argument, if the coefficients c_i satisfy $\sum_{i=0}^{\infty} c_i x^i < \infty$ [71]. Therefore for any analytic function $f(x)$ there exists a corresponding matrix function $f(\mathbf{W})$ constructed by the power series. The analytic (differentiability) constrained can be relaxed over a *closed* interval using the Weierstrass

(a) House

(b) $\mathbf{v}_2(\lambda_2 = 0.9966)$

(c) $\mathbf{v}_3(\lambda_3 = 0.9948)$

(d) $\mathbf{v}_{10}(\lambda_{10} = 0.9667)$

(e) $\mathbf{v}_{25}(\lambda_{25} = 0.8810)$

(f) $\mathbf{v}_{50}(\lambda_{50} = 0.7387)$

Figure 6.2: Some leading eigenvectors computed from the luminance channel of the house image using less than 0.04% of the pixels. ($h_x = 20$, $h_y = 5$)

approximation theorem [72] which only needs the function $f(x)$ to be continuous on an interval $[a, b]$. Based on this theorem, for every $\epsilon > 0$, there exists a polynomial $p_k(x)$ of sufficiently high degree k, such that for all x in $[a, b]$, we have $\|f(x) - p_k(x)\| < \epsilon$. Consequently, the matrix function $f(\mathbf{W})$ given in (6.3) can be approximated by a matrix polynomial $p_k(\mathbf{W})$ of order k:

$$f(\mathbf{W}) \approx p(\mathbf{W}) = c_0\mathbf{I} + c_1\mathbf{W} + c_2\mathbf{W}^2 + \cdots + c_k\mathbf{W}^k \tag{6.4}$$

with k a function of ϵ, as long as the eigenvalues of $\mathbf{W}$ reside in a closed compact interval. Having the matrix filter eigen-decomposition as $\mathbf{W} = \mathbf{V}\mathbf{S}\mathbf{V}^T$, *any* continuous function on the eigenvalue interval $[0, 1]$ can be approximated as:

$$\begin{aligned} f(\mathbf{W}) &\approx c_0\mathbf{V}\mathbf{I}\mathbf{V}^T + c_1\mathbf{V}\mathbf{S}\mathbf{V}^T + c_2\mathbf{V}\mathbf{S}^2\mathbf{V}^T + \cdots + c_k\mathbf{V}\mathbf{S}^k\mathbf{V}^T \\ &= \mathbf{V}(c_0\mathbf{I} + c_1\mathbf{S} + c_2\mathbf{S}^2 + \cdots + c_k\mathbf{S}^k)\mathbf{V}^T \\ &= \mathbf{V}p_k(\mathbf{S})\mathbf{V}^T \end{aligned} \tag{6.5}$$

where $p_k(\mathbf{S}) = \text{diag}[p_k(\lambda_1), p_k(\lambda_2), \cdots, p_k(\lambda_n)]$ denotes the mapped eigenvalues. Instead of approximating, we chose to directly use a polynomial function for $f(x)$, because it conveniently gives the interesting interpretation of the multiscale decomposition/reconstruction explained in the following.

6.2.1 Multiscale Detail Manipulation

Fig. 6.3 depicts the multiscale decomposition and reconstruction where the input image $\mathbf{y}$ is layered to k detail layers $\mathbf{y}_{d^i}$ and one basic smooth layer $\mathbf{y}_{s^k}$ such that the exact decomposition is:

$$\mathbf{y} = \mathbf{y}_{d^1} + \ldots + \mathbf{y}_{d^k} + \mathbf{y}_{s^k} \tag{6.6}$$

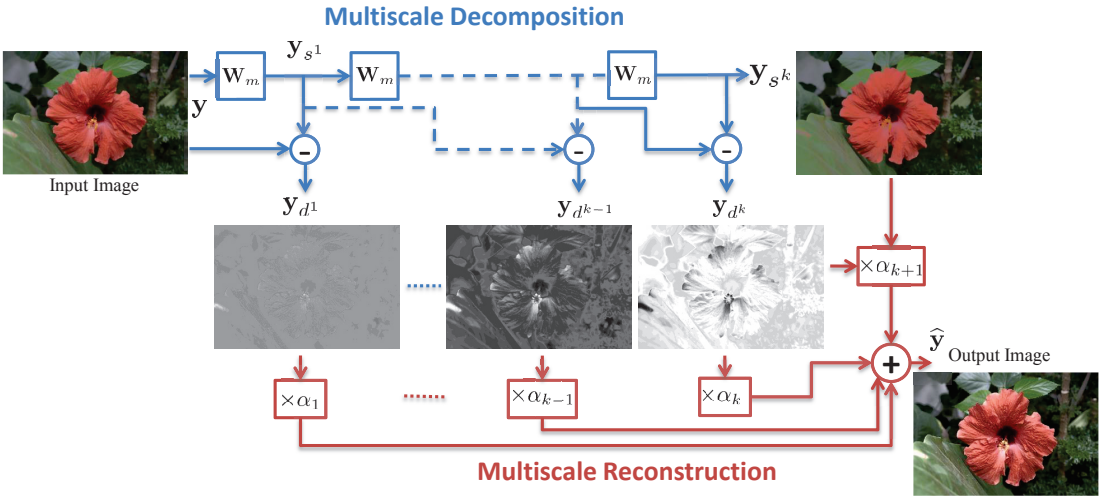


Figure 6.3: (a) Mutiscale decomposition: The low-pass filter $\mathbf{W}_m$ is used to extract detail layers $\mathbf{y}_{d^i}$. Multiscale reconstruction: Weighting each layer with α_i and adding them together.

The edited image $\widehat{\mathbf{y}}$ can be computed by weighting each layer and adding them back together:

$$\widehat{\mathbf{y}} = \alpha_1 \mathbf{y}_{d^1} + \ldots + \alpha_k \mathbf{y}_{d^k} + \alpha_{k+1} \mathbf{y}_{s^k} \tag{6.7}$$

Replacing the orthonormal eigen-decomposition of the filter $\mathbf{W}_m$ in the above, the equivalent filter is (Fig. 6.4):

$$\begin{aligned}
\widehat{\mathbf{y}} &= \alpha_1(\mathbf{I}-\mathbf{W}_m)\mathbf{y} + \alpha_2\mathbf{W}_m(\mathbf{I}-\mathbf{W}_m)\mathbf{y} + \ldots + \alpha_k\mathbf{W}_m^{k-1}(\mathbf{I}-\mathbf{W}_m)\mathbf{y} + \alpha_{k+1}\mathbf{W}_m^k\mathbf{y} \\
&\approx \mathbf{V}_m(\alpha_1(\mathbf{I}-\mathbf{S}_m) + \alpha_2\mathbf{S}_m(\mathbf{I}-\mathbf{S}_m) + \ldots + \alpha_k\mathbf{S}_m^{k-1}(\mathbf{I}-\mathbf{S}_m) + \alpha_{k+1}\mathbf{S}_m^k)\mathbf{V}_m^T\mathbf{y} \\
&= \mathbf{V}_m(\alpha_1\mathbf{I} + \mathbf{S}_m(\alpha_2-\alpha_1) + \mathbf{S}_m^2(\alpha_3-\alpha_2) + \ldots + \mathbf{S}_m^k(\alpha_{k+1}-\alpha_k))\mathbf{V}_m^T\mathbf{y} \\
&= \mathbf{V}_m f(\mathbf{S}_m)\mathbf{V}_m^T\mathbf{y}
\end{aligned} \tag{6.8}$$

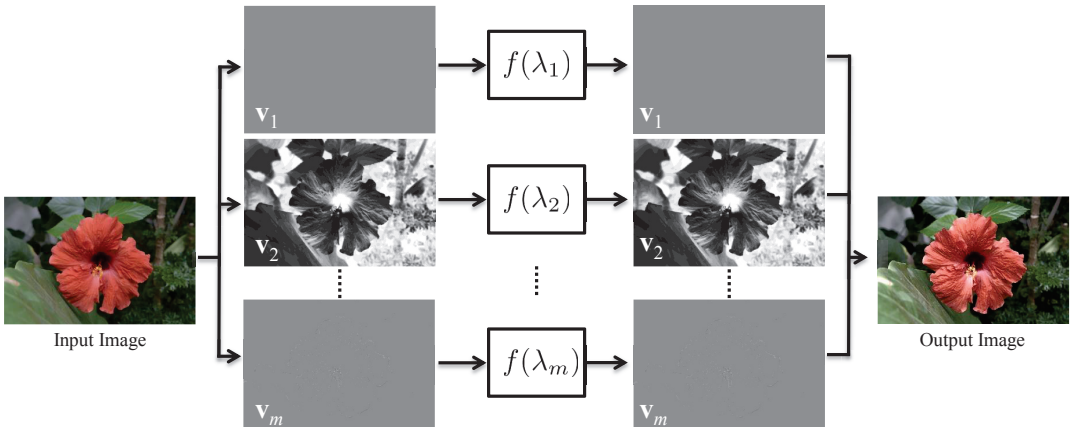


Figure 6.4: The process given in Fig. 6.3 can be interpreted as the band-pass $f(\mathbf{W}_m)$ in which the eigenvalues are a polynomial function of the low-pass filter's eigenvalues given by (6.9).

where the approximation concerns replacing $\mathbf{y}$ by $\mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}$ (see Appendix 6.B for detailed explanations). Function f has the following effect on each eigenvalue λ_j:

$$f(\lambda_j) = \alpha_1 + (\alpha_2 - \alpha_1)\lambda_j + (\alpha_3 - \alpha_2)\lambda_j^2 + ... + (\alpha_k - \alpha_{k-1})\lambda_j^{k-1} + (\alpha_{k+1} - \alpha_k)\lambda_j^k \quad (6.9)$$

This is a special polynomial with $f(0) = \alpha_1$ and $f(1) = \alpha_{k+1}$. The two coefficients α_1 and α_{k+1} correspond to the first detail layer and the basic smooth image, respectively. For example, a 3rd order polynomial is evaluated as a function of the input eigenvalues in Fig. 6.5. While the input eigenvalues act as a low-pass filter (this is equivalent to $\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$ in the new filter $f(\mathbf{W}_m)$, the function f can change the filter's behavior with α_i as the tuning parameter. The mapped eigenvalues in Fig. 6.5 (a) keep the details of the image untouched, but the leading eigenvalue manipulates the contrast of the image. Fig. 6.5 (b) is a high-pass filter with the opposite effect as compared to part (a). Fig. 6.5 (c) and (d) are two band-pass filters boosting the eigenvectors containing the main edges and possibly suppressing the existing noise.

Figs. 6.6-6.9 give a visual comparison of the effect of the filters described

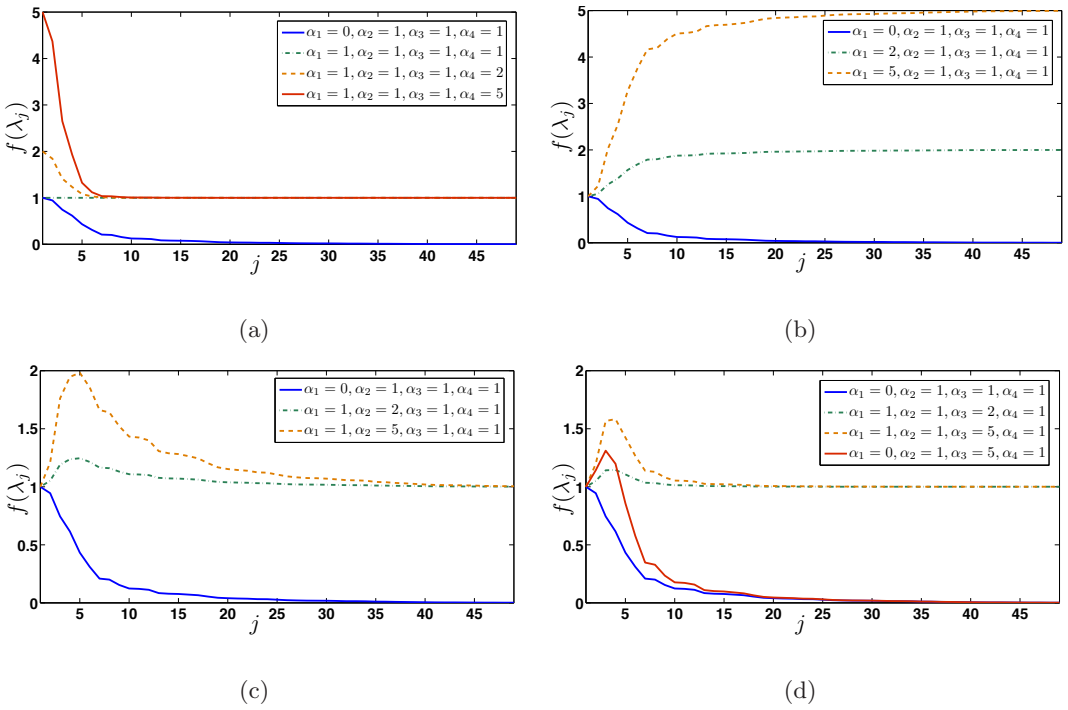


Figure 6.5: (a)-(d) The 3th order function $f(\lambda_j)$ is evaluated for different α_i weights.

above. As can be seen, applying the filter $f(\mathbf{W}_m)$ can boost the contrast and details of the image in fine, medium and coarse scales.

6.3 Globalizing Mask

As discussed earlier, the proposed global filter connects all the pixels in the image with weights commensurate to their similarity. As a result, any set of edits can be appropriately globalized to the whole image. In particular, one can automatically endow similar pixels with likewise similar editing parameters. Using the concept of diffusion maps [54], each pixel located at position $\mathbf{x}_i$ is mapped into a manifold defined

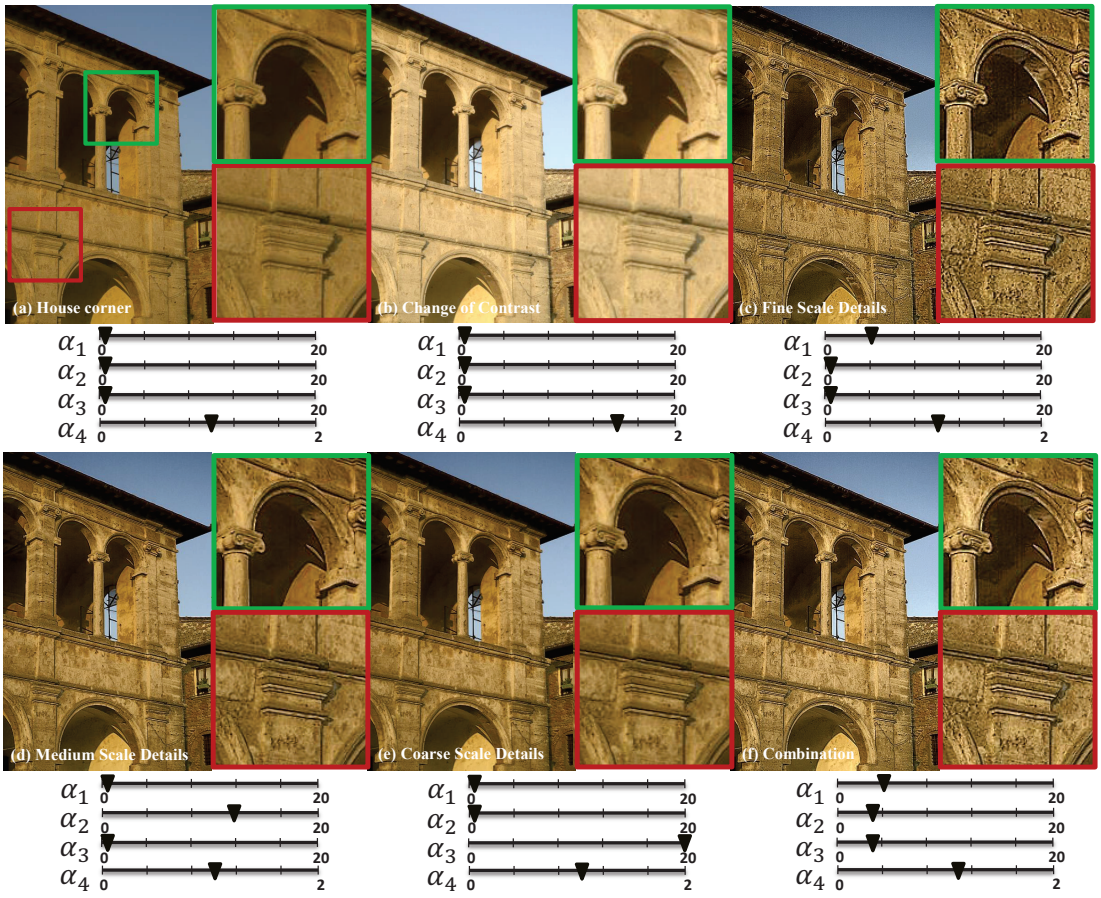


Figure 6.6: Contrast and detail manipulation of the house corner image. (a) Input image, (b) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1.4$, (c) $\alpha_1 = 4, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$, (d) $\alpha_1 = 1, \alpha_2 = 12, \alpha_3 = 1, \alpha_4 = 1$, (e) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 20, \alpha_4 = 1$, (f) $\alpha_1 = 4, \alpha_2 = 3, \alpha_3 = 3, \alpha_4 = 1.05$.

by the weighted eigenvectors as:

$$\mathbf{\Psi}_t(\mathbf{x}_i) = (\lambda_2^t v_{i2}, \lambda_3^t v_{i3}, ..., \lambda_m^t v_{im}) \tag{6.10}$$

where t denotes the diffusion parameter and v_{ik} denotes the i-th entry of the k-th eigenvector. The squared diffusion distance between two pixels located at positions $\mathbf{x}_i$

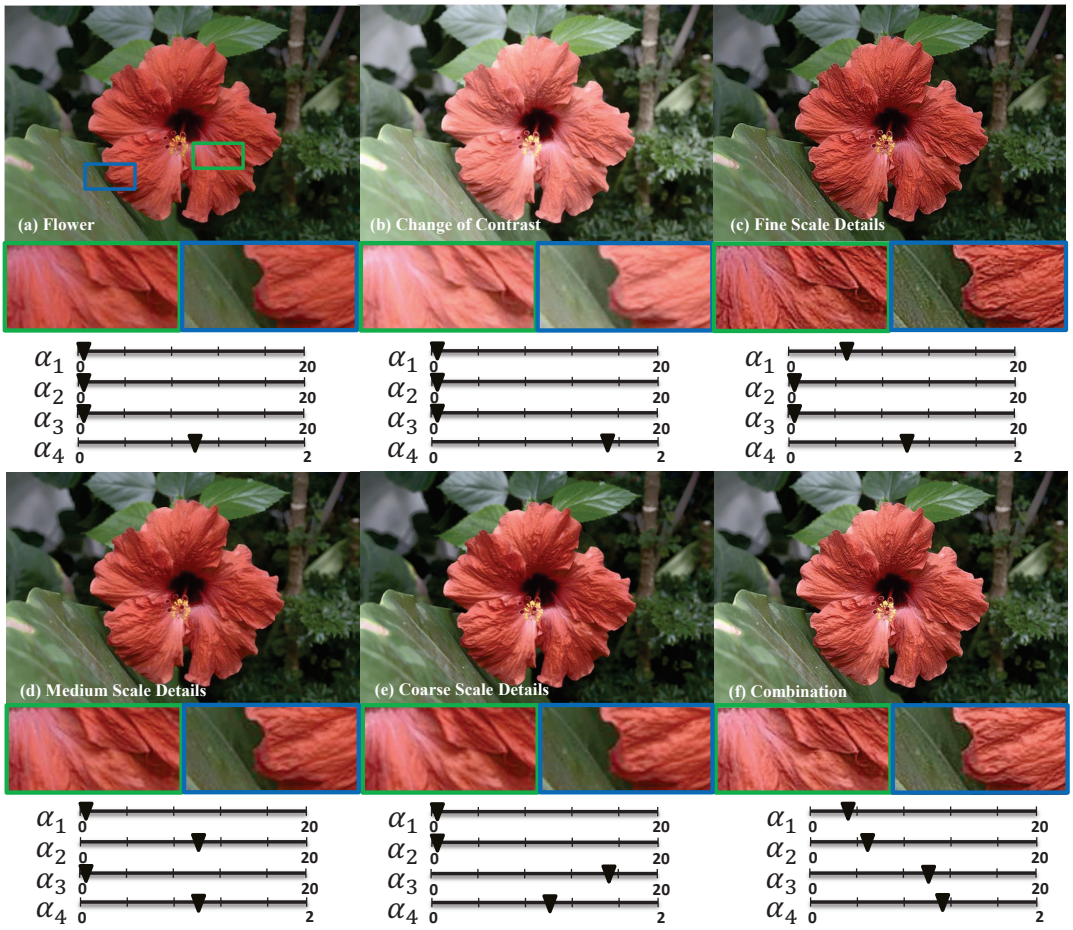


Figure 6.7: Contrast and detail manipulation of the flower image. (a) Input image, (b) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1.5$, (c) $\alpha_1 = 5, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$, (d) $\alpha_1 = 1, \alpha_2 = 10, \alpha_3 = 1, \alpha_4 = 1$, (e) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 15, \alpha_4 = 1$, (f) $\alpha_1 = 3, \alpha_2 = 5, \alpha_3 = 10, \alpha_4 = 1.1$.

and $\mathbf{x}_j$ is defined as:

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{\Psi}_t(\mathbf{x}_i) - \mathbf{\Psi}_t(\mathbf{x}_j)\|_2^2 = \sum_{l=2}^{m} \lambda_l^{2t}(v_{il} - v_{jl})^2 \tag{6.11}$$

As t grows, the diffusion distance between any two pixels on the manifold shrinks. We employ a simple Gaussian function to embed this squared distance into $[0, 1]$ interval

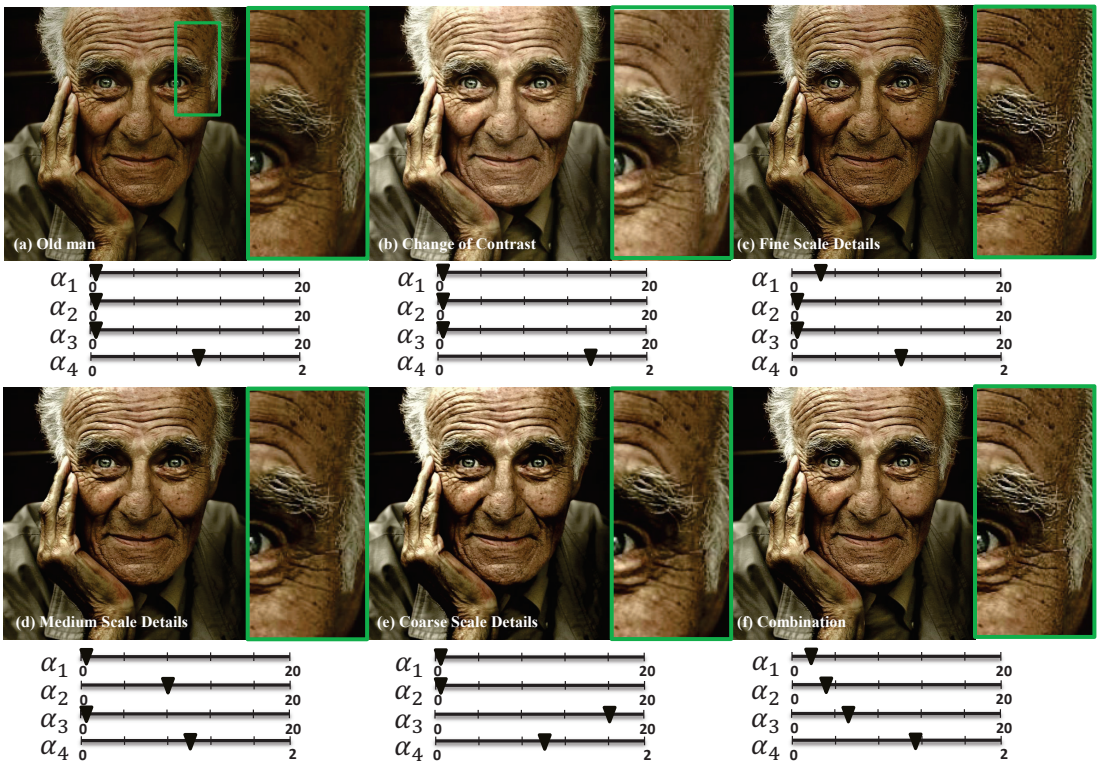


Figure 6.8: Contrast and detail manipulation of the old man image. (a) Input image, (b) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1.4$, (c) $\alpha_1 = 3, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$, (d) $\alpha_1 = 1, \alpha_2 = 8, \alpha_3 = 1, \alpha_4 = 1$, (e) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 16, \alpha_4 = 1$, (f) $\alpha_1 = 2, \alpha_2 = 3, \alpha_3 = 5, \alpha_4 = 1.1$.

and define the propagation mask as:

$$\mathbf{M}_t(\mathbf{x}_s) \equiv \exp(-D_t^2(\mathbf{x}_i, \mathbf{x}_s)) \tag{6.12}$$

where $\mathbf{x}_s$ refers to the average diffusion map (given in (6.10)) over a region S selected by the user. $\mathbf{M}_t$ values close to one represent the pixel candidates to be propagated with the edit and $\mathbf{M}_t$ values close to zero stand for pixels that receive a small fraction of the edit. Fig. 6.10 illustrates the globalizing mask evaluated for different diffusion parameters obtained from the user's input. As can be seen, larger diffusion parameters lead to more aggressively propagated masks. Using these masks, it can be seen that the

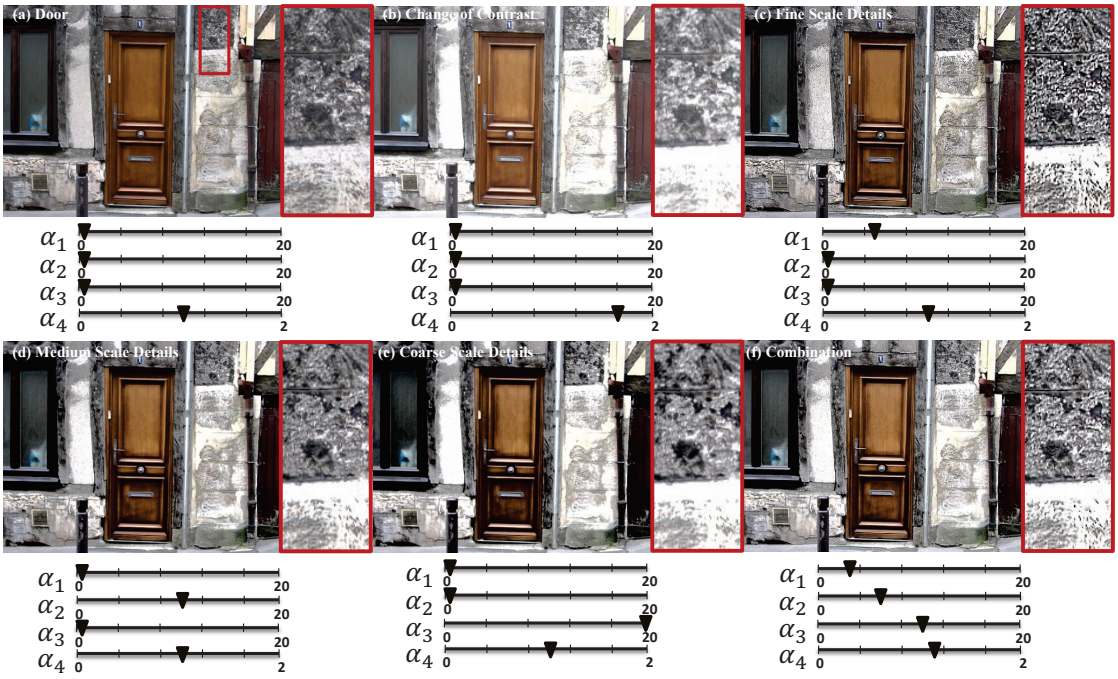


Figure 6.9: Contrast and detail manipulation of the door image. (a) Input image, (b) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1.5$, (c) $\alpha_1 = 5, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$, (d) $\alpha_1 = 1, \alpha_2 = 10, \alpha_3 = 1, \alpha_4 = 1$, (e) $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 20, \alpha_4 = 1$, (f) $\alpha_1 = 3, \alpha_2 = 6, \alpha_3 = 10, \alpha_4 = 1.1$.

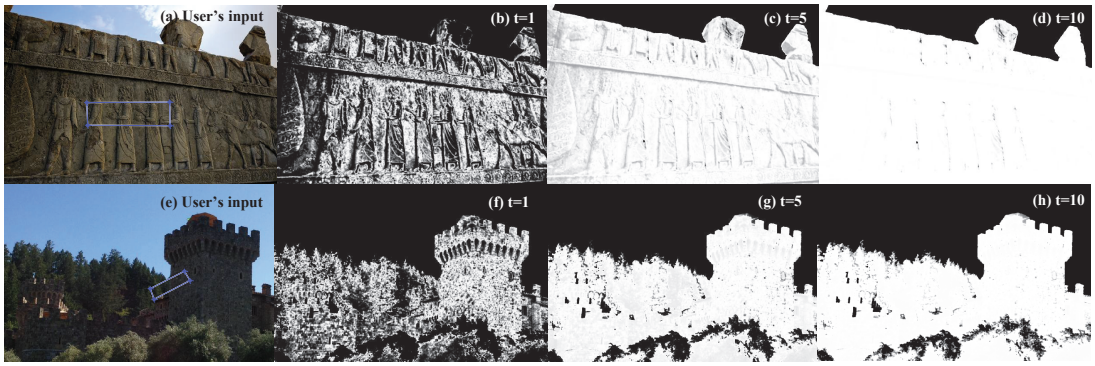


Figure 6.10: Propagation masks with different diffusion parameters for Persepolis and Castle images.

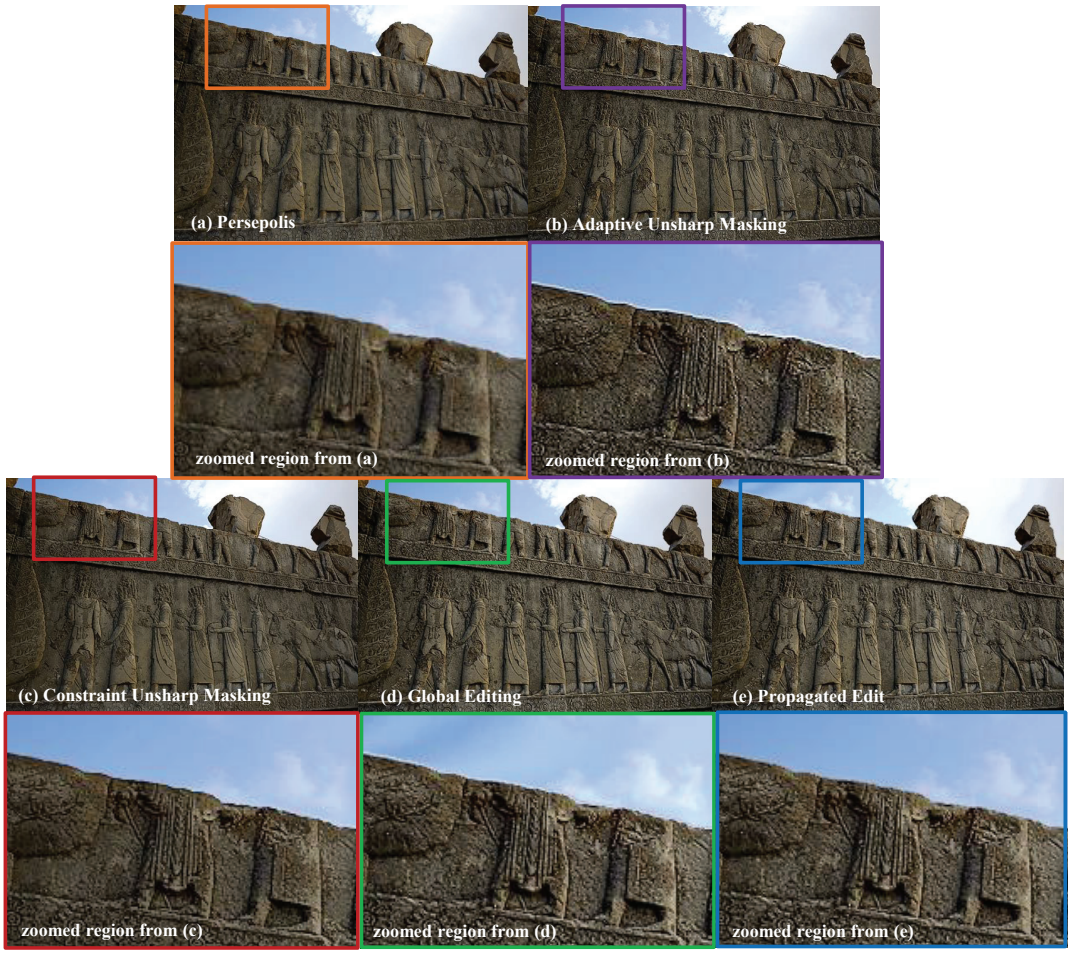


Figure 6.11: Detail propagation of the Persepolis image compared to the results from adaptive unsharp masking [6] and constrained unsharp masking [7]. Edit propagation (shown in (e)) reduces the halo artifacts compared to the global edit (shown in (d)) and adaptive unsharp masking [6].

edited region is propagated to the similar pixels in Fig. 6.11 and Fig. 6.12. Comparing results of the global editing (Fig. 6.11(d) and Fig. 6.12(d)) and propagated edit (Fig. 6.11(e) and Fig. 6.12(e)), we can conclude that: (1) The main edges of the image are preserved and there is almost no halo effect on them, (2) existing noise in image regions with low SNR is no longer boosted. Our results are also compared to the adaptive [6]

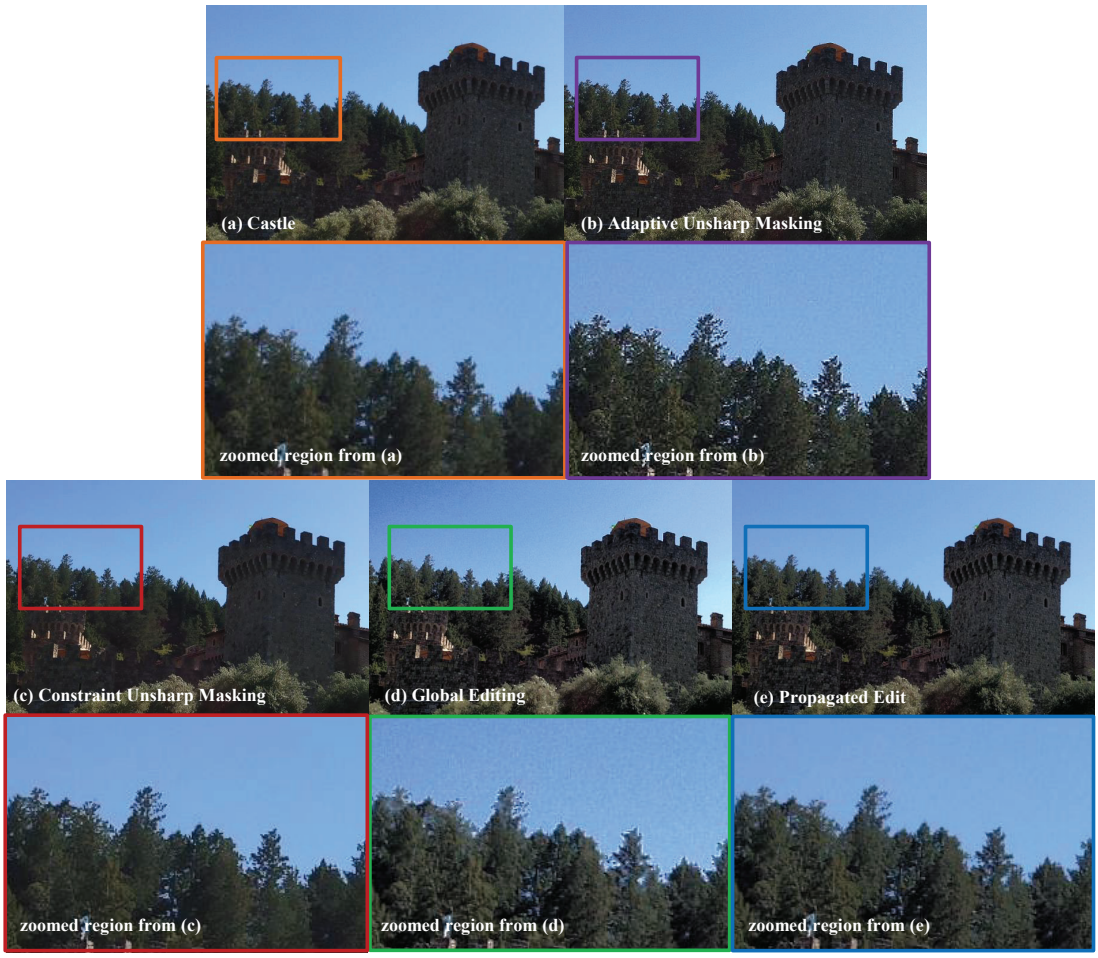


Figure 6.12: Detail propagation of the castle image compared to the results from adaptive unsharp masking [6] and constrained unsharp masking [7]. Edit propagation (shown in (e)) reduces noise and halo artifacts compared to the global edit (shown in (d)) and adaptive unsharp masking [6].

and constrained [7] unsharp masking in Fig. 6.11 and Fig. 6.12. As can be seen, while the sharpness is well enhanced, both noise and artifacts are suppressed in our results.

Depending on the application, the user can include the spatial term in the NLM kernel (6.1). This leads to a class of eigenvectors shown in Fig. 6.2 where similarity is captured more locally. Propagation masks built on these eigenvectors are used to

explore other applications of our framework in the following.

6.4 Practical Applications

The global filter studied here has many applications such as sharpening, denoising, recoloring, colorization, abstraction and fake depth of field. For the purpose of recoloring and abstraction, the kernel can be computed from the luminance channel. Color channels can be used to make the kernel for colorization and fake depth of field. In general, after computing the leading eigenvectors, an appropriate propagation mask is built to apply the user's input.

6.4.1 Recoloring

Figs. 6.13-6.14 illustrate recoloring examples where the color strokes are propagated. These color strokes are applied on the chrominance channels. As discussed previously, image pixels can be represented by their corresponding diffusion map coordinates given in (6.10). Thus, each image region selected by user strokes contains a group of pixels with the diffusion map vectors assigned to them. A binary mask can be obtained (see Figs. 6.13(c)-6.14(c)) using k-means clustering with the center of each diffusion map group as its initialization points. As shown in Figs. 6.13(d)-6.14(d), using this guide mask, the input color brushes are propagated in the chroma channels. Again, the mask can be tuned using the diffusion parameter t. Our globalizing mask for the colorization application is also obtained through the same procedure described here.

The non-local nature of our recoloring method is better illustrated in Fig. 6.13 where a small color brush on one tulip is effectively propagated to other tulips. Color replacement tool in Photoshop CC was used for the same purpose in Fig. 6.13(e). As

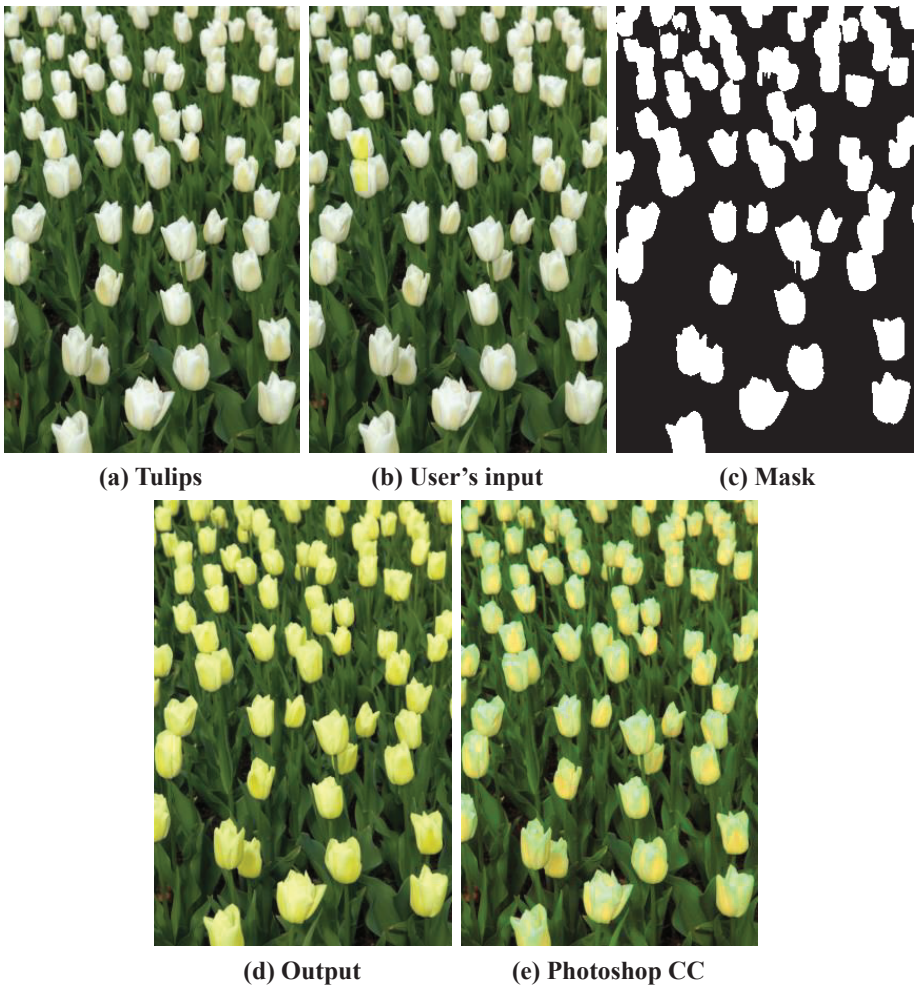


Figure 6.13: Recoloring example based on the propagation mask. Results from the proposed method (d) and color replacement tool of Photoshop CC (e) are compared.

can be seen, the result obtained from Photoshop fails to evenly propagate the color stroke. Given that the color replacement tool employs a soft propagation mask, it tends to unequally spread the input color even across the same objects. Fig. 6.14 shows recoloring example by two different color brushes. In this case the mask was tuned by the diffusion parameter (t) in a way that the right and left side flowers in the image fall into different clusters.

Figure 6.14: Recoloring example based on the propagation mask. The mask (c) is built based on the two input color brushes (b).

6.4.2 Colorization

Our colorization example is shown in Fig. 6.15 where three input color brushes are propagated on the gray input image. The mask is based on affinities computed from the luminance channel. Same as recoloring, the mask is binerized using k-means to obtain a segmented guide map Fig. 6.15(c). The edit is propagated to the chroma components and the luminance channel is not changed. We also compared our result with the colorization method of Levin et al. [8] where an optimization problem is solved to propagate the input color scribbles. For the purpose of fair comparison, we fed the same color brushes shown in Fig. 6.15(b) to both methods. This example indicates

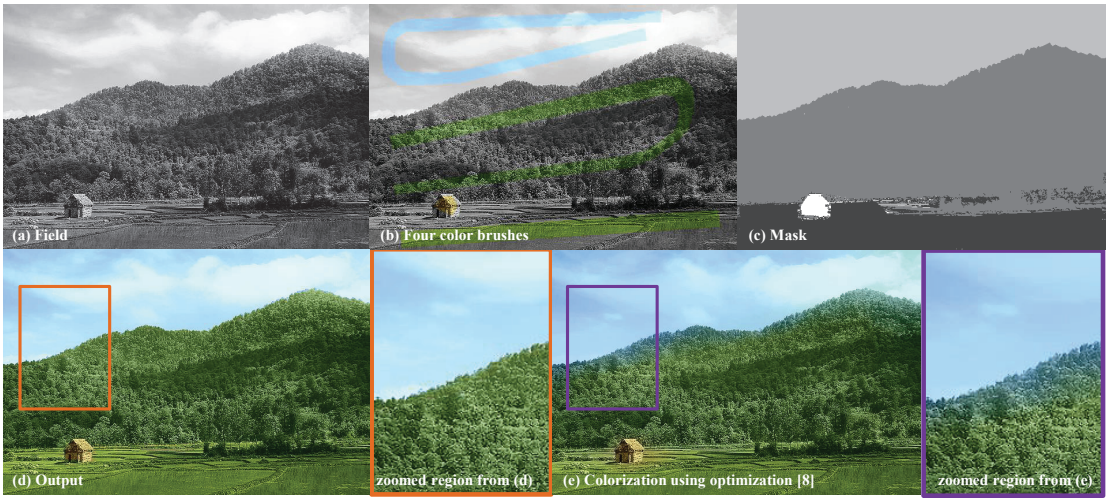


Figure 6.15: Colorization example based on the propagation mask. The gray scale image is given in (a) and our input brush colors are shown in (b). Using the mask in (c) the color brushes are propagated throughput the gray image (shown in (d)). For a better comparison, our output is shown next to the results from [8].

superiority of our method while working with only a few color brushes. As can be seen in the results of [8], blue color is leaked on the trees. While adding more color brushes especially around the object boundaries may result in a better performance from [8], our method takes advantage of the guide mask to easily distinguish the sky and forest regions. Yet, working with segmented guide masks might sometimes lead to false color propagation too. The project webpage contains more colorization examples comparing both methods and also shows some failed cases.

6.4.3 Fake Depth of Field

We can also employ the proposed propagation mask for applying fake depth of field. Although the produced mask does not represent the true depth information of the image, it still can be used to apply selective blur to each image region. Two examples are illustrated in Fig. 6.16, where the user selects the region which is supposed to stay

in focus. Then, our blur map is built using the globalizing mask explained in Sec. 6.3 and based on this, each pixel is blurred using a Gaussian function. The blur intensity of each pixel is proportional to the mask such that mask values close to 0 receive more blur. Our results are compared with the manually edited results obtained from Photoshop CC software where the same Gaussian blur was selected. The blur propagation of the results from Photoshop are manually tuned to get the best possible output. As can be seen, results obtained from our blur map are competitive with the Photoshop outputs.

Based on our observation, the kernel's spatial and photometric terms play important roles in building the blur map. We used the RGB channels in the photometric term of the NLM kernel and included the spatial term as:

$$K_{ij} = \exp\left\{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i^{RGB} - \mathbf{y}_j^{RGB}\|^2}{h_y^2}\right\} \tag{6.13}$$

The spatial term enforces the locality of the propagated blur and the RGB photometric term distinguishes object boundaries with different colors. Relatively large h_y values suit blur maps of images with distinctive color boundaries in background and foreground. On the other hand, blur maps of images with comparable color in background and foreground could be better localized by tuning h_x. Of course, any fake depth of field method has its own limitations since the real depth information is not available.

6.4.4 Abstraction

Fig. 6.17 illustrates the abstraction application where iteration of the filter leads to stylized filtered image as $\widehat{\mathbf{y}} = \mathbf{W}_m^k \mathbf{y}$ (effect of the iteration number k is illustrated in the project website where different results are shown with various iteration numbers). In practice, instead of multiplication of the filter matrix, the iterative filtering is implemented by raising the eigenvalues to a power. Fig. 6.17 shows our stylized

Figure 6.16: Fake depth of field example. Based on the depth map, pixels with lower map values are more blurred. Our results shown in (c) and (g) are competitive to the manually edited outputs from Photoshop software in (d) and (h).

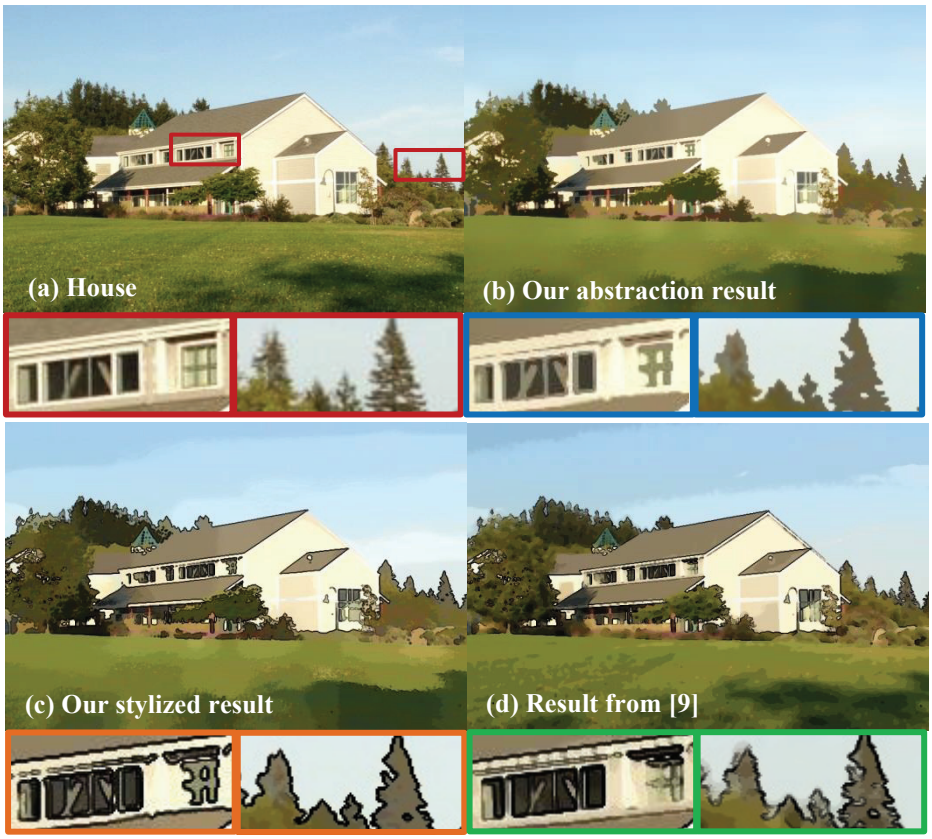


Figure 6.17: Image abstraction application. (a) Input image and zoomed regions, (b) Our abstraction result as $\widehat{\mathbf{y}} = \mathbf{W}_m^k \mathbf{y}$ with iteration number $k = 0.1$, (c) The abstracted image given in (b) is stylized using the edge-exaggeration and luminance quantization of [9], (d) Result from [9].

result compared to the one from Winnemöler et al. [9]. In this method, the input image is filtered by an iterative bilateral filter to produce a piece-wise smooth output. Then, an edge-exaggeration step followed by luminance quantization is employed to stylize the abstracted image (see Fig. 6.17(d)). To further stylize our abstracted result shown in Fig. 6.17(b) and also for a better comparison to [9], the edge-exaggeration and luminance quantization of [9] was applied on our abstracted result to produce Fig. 6.17(c). Although comparing abstracted images is very subjective, Fig. 6.17(c) and Fig. 6.17(d)

show that replacing the iterative bilateral smoother of [9] with our filter can result in visually superior results.

Summary – In the chapter, some applications of the global, affinity based filters are explored. Having eigenfunctions of these filters, complex nonlinear smoothing or sharpening operators are easily implemented by means of mapping the corresponding eigenvalues. The global nature of the eigenvectors let us propagate these edits easily throughout the image. This type of editing enables us to prevent noise magnification and also effectively reduce common artifacts such as halo. Beside filtering applications, the obtained propagation map can be used to effect many other types of edits.

6.A Parameter Tuning of the Filter

The approximated filter spans the input image onto its eigenvectors. These projected image coefficients are $b_j = \mathbf{v}_j^T \mathbf{y}$ where $j = 1, \ldots, m$. For any $m \leq n$, energy of these coefficients is less than the energy of the input image as $\sum_{j=1}^{m} b_j^2 \leq \sum_{i=1}^{n} y_i^2$ (ideally, the two energy terms are equal). The approximated filter should preserve energy of the image, which means β should be close to one, where:

$$\beta = \frac{\sum_{j=1}^{m} b_j^2}{\sum_{i=1}^{n} y_i^2} \tag{6.14}$$

Assuming that the sample number m is fixed, β is determined by the kernel smoothing parameter.

The approximation accuracy is directly affected by the smoothing parameter of the NLM kernel (h_y given in (6.1)). For the images given in Fig. 6.18, β is computed using different smoothing parameters in Table 6.1. As can be seen, β gets closer to one for larger h_y. However, for a better multiscale decomposition, it is necessary to

(a) Flower (b) Rock (c) Monastery

Figure 6.18: Test images with the following mean gradient magnitudes: (a) 3.66, (b) 7.51, (c) 12.78.

select a small smoothing parameter. We set $\beta \geq 99.9\%$ to determine the optimal kernel parameter. This test could be useful after approximation to check if the kernel parameter is not too small or large. To avoid the computation burden, a simple ad-hoc method is introduced in the following to adapt the kernel smoothing parameter into the latent image.

Image content has a direct impact on the accuracy of the approximation. Images containing more variants tend to settle for larger h_y (for a fixed β). This can be observed by comparing the mean gradient magnitude of the test images (given in Fig. 6.18) and the β values in Table 6.1. Without any need to compute β, we empirically estimate the smoothing parameter as a function of the mean gradient magnitude:

$$\widehat{h}_y = \frac{0.6}{n}\sqrt{\mathbf{g}_{\mathbf{x}_1}^T\mathbf{g}_{\mathbf{x}_1} + \mathbf{g}_{\mathbf{x}_2}^T\mathbf{g}_{\mathbf{x}_2}} \tag{6.15}$$

where $\mathbf{g}_{\mathbf{x}_1}$ and $\mathbf{g}_{\mathbf{x}_2}$ denote the gradient vectors in $\mathbf{x}_1$ and $\mathbf{x}_2$ directions.

6.B Approximation of Eq. 6.8

The approximation made in (6.8) is $\mathbf{y} \approx \mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}$, or in effect replacing the identity matrix with $\mathbf{V}_m\mathbf{V}_m^T$. This assumption means that the underlying image $\mathbf{y}$

Table 6.1: β percentage values for images given in Figure 6.18

h_y	2	4	6	8	10
Flower	**99.95**	99.99	100	100	100
Rock	99.83	**99.95**	99.97	99.98	99.99
Monastery	99.63	99.80	99.88	**99.93**	99.96

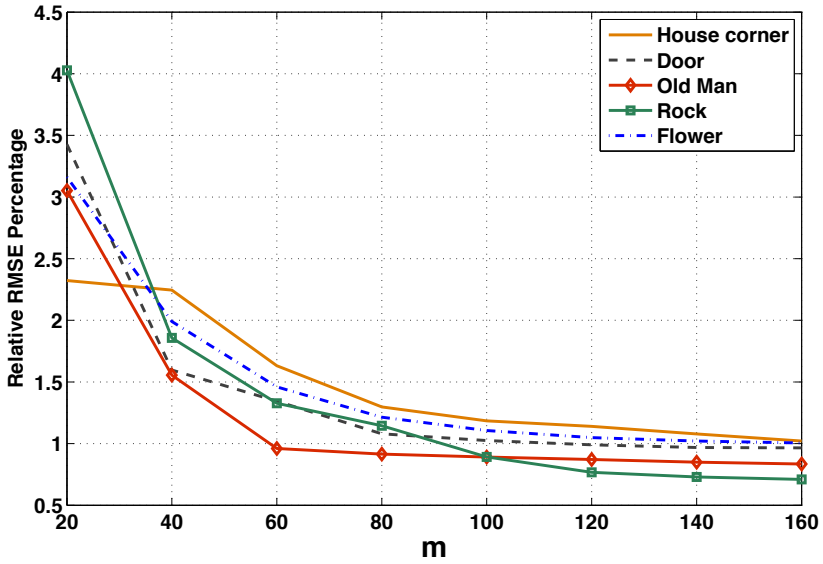


Figure 6.19: Approximation error of (6.8) computed as relative RMSE $= \frac{\|\mathbf{y}-\mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}\|}{\|\mathbf{y}\|} \times$ 100% for different numbers of retained eigenvectors m.

should be "well represented" (or spanned) by the retained eigenvectors. For a better validation of this approximation, Fig. 6.19 shows relative root-mean-square-error[2] for some of the test images. The relative RMSE values are computed for images in the range [0,255] and various values of m. Unsurprisingly, the approximation error decays quickly as the number of retained eigenvectors grows. Overall, keeping $m > 75$ in our framework leads to around 1% approximation error.

[2]relative RMSE $= \frac{\|\mathbf{y}-\mathbf{V}_m\mathbf{I}\mathbf{V}_m^T\mathbf{y}\|}{\|\mathbf{y}\|} \times 100\%$

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This dissertation introduces an innovative image-dependent filtering paradigm capable of realizing state-of-the-art image enhancement and editing results. Our filtering process is a simple weighted averaging with image-dependent weights representing local or global pixel similarities. The pixel averaging operation can be interpreted as shrinking or boosting the spectrum of the corresponding filter weight matrix. Depending on the filtering application, a proper mapping function can be used to manipulate the corresponding eigenvalues in the spectrum.

In the process of exploring the local pixel similarities, a new spatially adaptive denoising filter (SAIF) was introduced in Chapter 2. Using the iteration mechanisms of diffusion and twicing (which are equivalent to shrinkage or boosting of the filter eigenvalues), SAIF improves performance of similarity-based filters such as NLM [2], bilateral [1] and LARK [3]. In the SAIF algorithm, the appropriate type and number of iteration in each patch is determined through the estimation of the local SNR. Ex-

periments show that this method enhances performance of the base filter both visually and quantitatively.

By exploring the global pixel similarities, a novel image filtering algorithm was described in Chapter 3. This global filter relies on the low-rank property of the weight matrix, which can be approximated with only a few leading eigenvectors. These eigen-functions provide a multiscale representation of the underlying image, while the eigenvalues denote the significance of each basis function. The eigen approximation is implemented by the Nyström technique, which operates on a few sampled pixels and effectively lowers the required memory and complexity from quadratic to linear, in the number of pixels.

Denoising application of the global filter (GLIDE) was presented in Chapter 4 where each pixel was denoised using all available pixels in the image. Our statistical analysis in Chapter 5 showed that global denoising is asymptotically optimal. Namely, incrementing the number of pixels n leads to lowering the mean-squared-error at the rate of $\frac{1}{n^{\alpha}}$ (α depends on the content of the image).

Practical applications of the global filter are not limited to image denoising. As explained in Chapter 6, a polynomial mapping of the eigenvalues enables the filter for other applications, such as tone-mapping, edge-aware sharpening and abstraction. All of these filtering results can be propagated to similar image regions using a diffusion mask developed from filter eigenvectors.

7.2 Future Work and Extensions

The filer approximation steps and the applications of the global filter can be improved as parts of our future work. Our proposed sampling approach is simple,

yet not necessarily optimal. More sophisticated and effective sampling strategies could lessen the error and computational burden of the filter approximation. The order of the filter basis functions is also an important issue to address in future studies. In this dissertation, the eigen-functions are ordered based on the magnitude of the eigenvalues, which is optimal for approximating the filter, but not optimal for representing an image. In other words, our image filtering process can benefit from other possible ordering modes of the eigenvectors. A further extension to this dissertation could be an automatic tuning parameter selection for the filter. Since the filter has only a few knobs, a possible approach to tune the filter is parameter learning. These topics are explained in the following sections.

7.2.1 Adaptive Sampling

The Nyström method is an efficient technique for obtaining a low-rank approximation of the large affinity matrix, based on a subset of its samples. The quality of this approximation depends strongly on the set of samples used, which are usually selected randomly. An efficient sampling technique can effectively reduce memory and computational complexity of the global filter by representing the latent image with even fewer basis functions. In the current work we used evenly-spaced pixel samples to take advantage of the local redundancies of natural images.

Recent research on various sampling options of low-rank approximation aim to select more informative rows (columns) from the kernel matrix $\mathbf{K}$ [43, 73–77]. The Sparse Matrix Greedy Approximation (SMGA) [73] and the Incomplete Cholesky Decomposition (ICL) [74, 75] are among the first adaptive sampling schemes suggested for the Nyström extension. SMGA is an iterative matching pursuit process which se-

lects random samples from subsets that make the least approximation error. ICL is a deterministic technique that adaptively selects rows based on potential pivots of the Incomplete Cholesky Decomposition. More recently, clustering-based sampling was introduced in [76] where the cluster centroids from k-means were selected as informative samples. Also, new adaptive random sampling methods are proposed in [43, 77] where samples are selected based on an estimated probability distribution to outperform the uniform distribution sampling. Overall, the existing sampling approaches are generic and not specialized for images. Incorporating the existing spatial redundancy of images and greedy sampling techniques can possibly improve our filter approximation step.

Beside the quality of samples in the eigen approximation of the affinity matrix, specifics of the basis functions derived from the samples are also important. Assuming that the number of samples m is fixed, the retained eigenvectors to be used in the filtering process should be highly descriptive of the latent image. This is discussed in more details next.

7.2.2 Basis Ordering

The filter approximation provides us with the m leading orthonormal eigenvectors $\mathbf{V}_m = [\mathbf{v}_1, ..., \mathbf{v}_m]$ and eigenvalues $\mathbf{S}_m = \text{diag}[\lambda_1, ..., \lambda_m]$ where $0 \leq \lambda_m \leq ... < \lambda_1 = 1$. This implies that the approximation minimizes the error when reconstructing the filter $\mathbf{W}_m$. However, the purpose of the filter approximation is to span the input image using the truncated eigenvector space $\mathbf{V}_m$. More explicitly, the quality of the filter approximation hinges on the behavior of the projected image coefficients into the leading eigenvectors:

$$|\mathbf{b}| = |\mathbf{V}_m^T \mathbf{y}| = [|\mathbf{v}_1 \mathbf{y}|, ..., |\mathbf{v}_m \mathbf{y}|]^T = [|b_1|, ..., |b_m|]^T \tag{7.1}$$

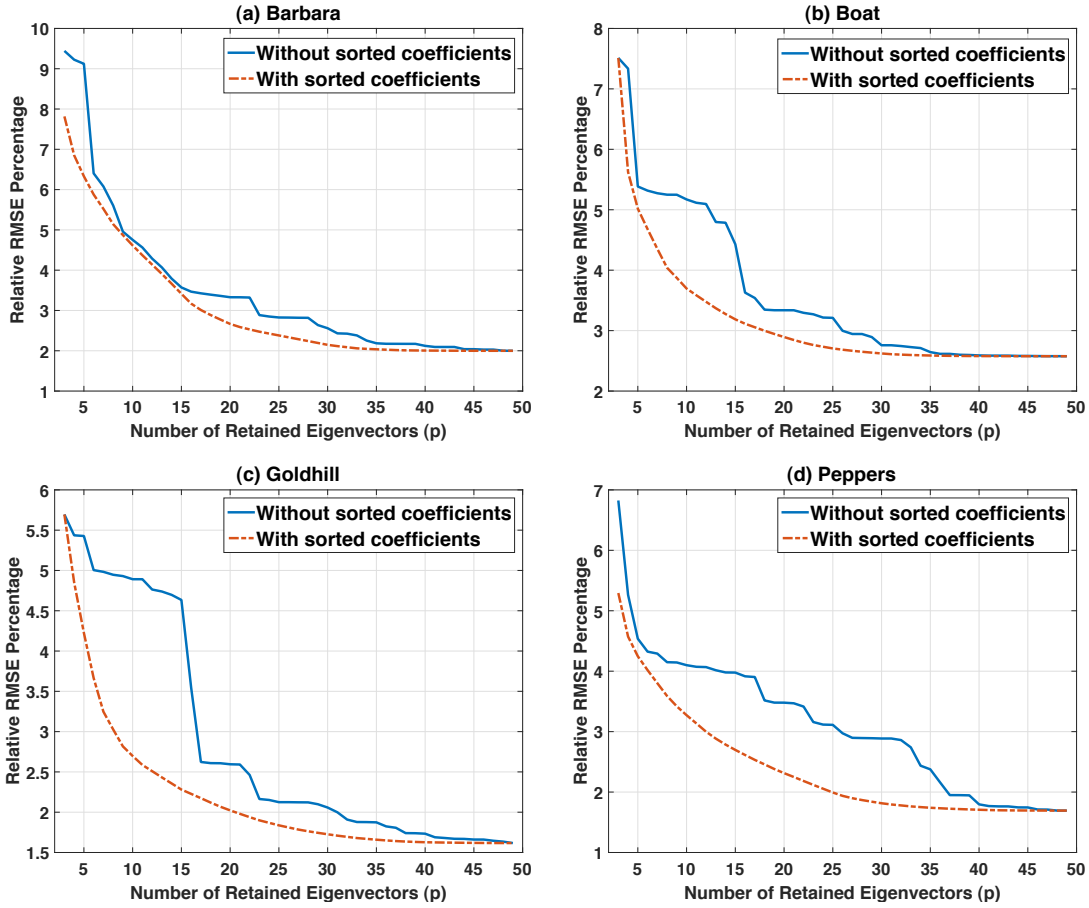


Figure 7.1: Effect of the projection coefficient sorting on the filter approximation for some test images from Chapter 5. The approximation error is computed as relative RMSE $= \frac{\|\mathbf{y}-\mathbf{V}_p\mathbf{I}\mathbf{V}_p^T\mathbf{y}\|}{\|\mathbf{y}\|} \times 100\%$ for fixed number of samples $m = 50$ and various retained eigenvectors p.

These coefficients represent the signal component in each channel. Magnitude of the projection coefficient b_j denotes significance of the j-th basis function ($\mathbf{v}_j$) in modeling the latent image $\mathbf{y}$. One might argue that sorting the leading filter eigenvectors based on the magnitude of their respective coefficients could lead to a more reasonable truncation of the filter. The experiment illustrated in Fig. 7.1 exemplifies the filter approximation with and without projection coefficient sorting. As can be seen,

after sorting the projection coefficients, fewer eigenvectors are needed to obtain a given approximation error[1]. The coefficient sorting step could be an extension to our algorithm to further improve memory-efficiency of our filtering process. Next, the process of learning the editing parameters is discussed.

7.2.3 Learning Filter Knobs

The editing operations described in Chapter 6 are achievable with only a few tuning parameters. Sharpening, tone mapping and brightness tuning were implemented by varying only four polynomial coefficients. This leads us to employ our global filter to learn unknown image editing operators. For instance, basic operators employed in commercial auto enhancing softwares are undisclosed. One might argue that proper sets of parameters in our global filter could "imitate" the editing style of an arbitrary filter.

Recently a few learning-based image enhancement and editing methods have been proposed [78–83]. Supervised learning is employed in some of these approaches to personalize the edit using before and after image pairs [79–82]. Other methods, such as [78, 83] use the image context to estimate the best restoration parameters for exposure correction and contrast enhancement operations. Together these methods assume separate filtering operations with independent parameters to be learned. In these methods, sharpening, color balancing, tone mapping and exposure correction are applied separately. On the other hand, our proposed filter provides a unified editing framework including all these operations.

[1]It is worth mentioning that computing the projection coefficients $\mathbf{b}$ will not impose any extra computation to our current filtering framework, because the filtered image is $\widehat{\mathbf{y}} = \mathbf{V}_m f(\mathbf{S}_m)\mathbf{V}_m^T\mathbf{y} = \mathbf{V}_m f(\mathbf{S}_m)\mathbf{b}$.

7.2.3.1 Estimating Filter Parameters Using Image Pairs

Our filtering scheme introduced in Chapter 6 was based on the eigenvalue mapping as:

$$\widehat{\mathbf{y}} = \mathbf{V}_m f_{\boldsymbol{\alpha}}(\mathbf{S}_m)\mathbf{V}_m^T\mathbf{y} \tag{7.2}$$

where a 3rd order polynomial function with parameters $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^T$ is applied on the eigenvector matrix $\mathbf{S}_m$. Starting with the image pairs $\widehat{\mathbf{y}}$ and $\widetilde{\mathbf{y}}$, where the latter denotes a reference image, our proposed training framework for estimating the filter parameters $\boldsymbol{\alpha}$ is:

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \ \frac{1}{n}\|\widetilde{\mathbf{y}} - \widehat{\mathbf{y}}\|^2 + \gamma\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^r\|^2 \tag{7.3}$$

where the first term minimizes the squared error between the image pairs, and the second term guarantees closeness of $\widehat{\boldsymbol{\alpha}}$ to a pre-defined reference parameters $\boldsymbol{\alpha}^r$. The positive weight γ varies the dominance of each term in our constrained least square problem. Our objective function can be rewritten by replacing $\widehat{\mathbf{y}}$ with the filtering scheme in (7.2):

$$J(\boldsymbol{\alpha}) = \frac{1}{n}\|\widetilde{\mathbf{y}} - \mathbf{V}_m f_{\boldsymbol{\alpha}}(\mathbf{S}_m)\mathbf{V}_m^T\mathbf{y}\|^2 + \gamma\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^r\|^2 \tag{7.4}$$

With a polynomial mapping $f_{\boldsymbol{\alpha}}(.)$, $J(.)$ is a quadratic function of $\boldsymbol{\alpha}$; meaning that, solving $\frac{\partial J}{\partial \boldsymbol{\alpha}} = 0$ results in the optimal parameters as:

$$\widehat{\boldsymbol{\alpha}} = \mathbf{A}^{-1}\mathbf{c} \tag{7.5}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{b}^T(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} + n\gamma & \mathbf{b}^T\mathbf{S}_m(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^2(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^3(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} \\ \mathbf{b}^T\mathbf{S}_m(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^2(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} + n\gamma & \mathbf{b}^T\mathbf{S}_m^3(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^4(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} \\ \mathbf{b}^T\mathbf{S}_m^2(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^3(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^4(\mathbf{I}_m - \mathbf{S}_m)^2\mathbf{b} + n\gamma & \mathbf{b}^T\mathbf{S}_m^5(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} \\ \mathbf{b}^T\mathbf{S}_m^3(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^4(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^5(\mathbf{I}_m - \mathbf{S}_m)\mathbf{b} & \mathbf{b}^T\mathbf{S}_m^6\mathbf{b} + n\gamma \end{bmatrix} \tag{7.6}$$

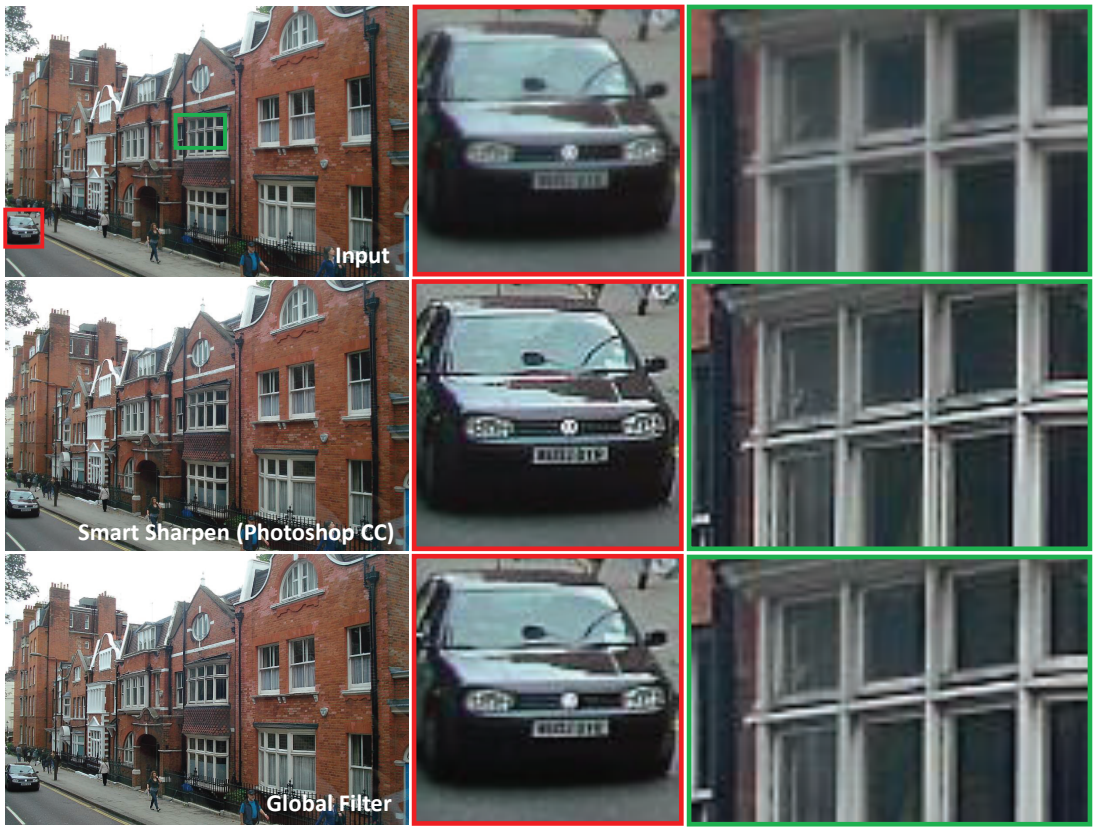


Figure 7.2: Input image is sharpened by smart sharpening tool in Photoshop CC to produce the reference image $\widetilde{\mathbf{y}}$. Then, the cost function given in (7.3) is solved to produce the global filter output. The estimated parameters are $\widehat{\boldsymbol{\alpha}} = [2.86, 0.01, 0.24, 1]^T$.

and

$$\mathbf{c} = \begin{bmatrix} \mathbf{b}^T(\mathbf{I}_m - \mathbf{S}_m)\widetilde{\mathbf{b}}_m + n\gamma\alpha_1^r \\ \mathbf{b}^T\mathbf{S}_m(\mathbf{I}_m - \mathbf{S}_m)\widetilde{\mathbf{b}}_m + n\gamma\alpha_2^r \\ \mathbf{b}^T\mathbf{S}_m^2(\mathbf{I}_m - \mathbf{S}_m)\widetilde{\mathbf{b}}_m + n\gamma\alpha_3^r \\ \mathbf{b}^T\mathbf{S}_m^3\widetilde{\mathbf{b}}_m + n\gamma\alpha_4^r \end{bmatrix} \tag{7.7}$$

in which $\mathbf{b} = \mathbf{V}_m^T\mathbf{y}$ and $\widetilde{\mathbf{b}} = \mathbf{V}_m^T\widetilde{\mathbf{y}}$. As can be seen, the positive weight γ can improve the condition number of the matrix $\mathbf{A}$.

Fig. 7.2 and Fig. 7.3 demonstrate examples of minimizing the cost function in (7.3). Throughout the experiment γ is fixed as 2, and the reference parameters $\boldsymbol{\alpha}^r$

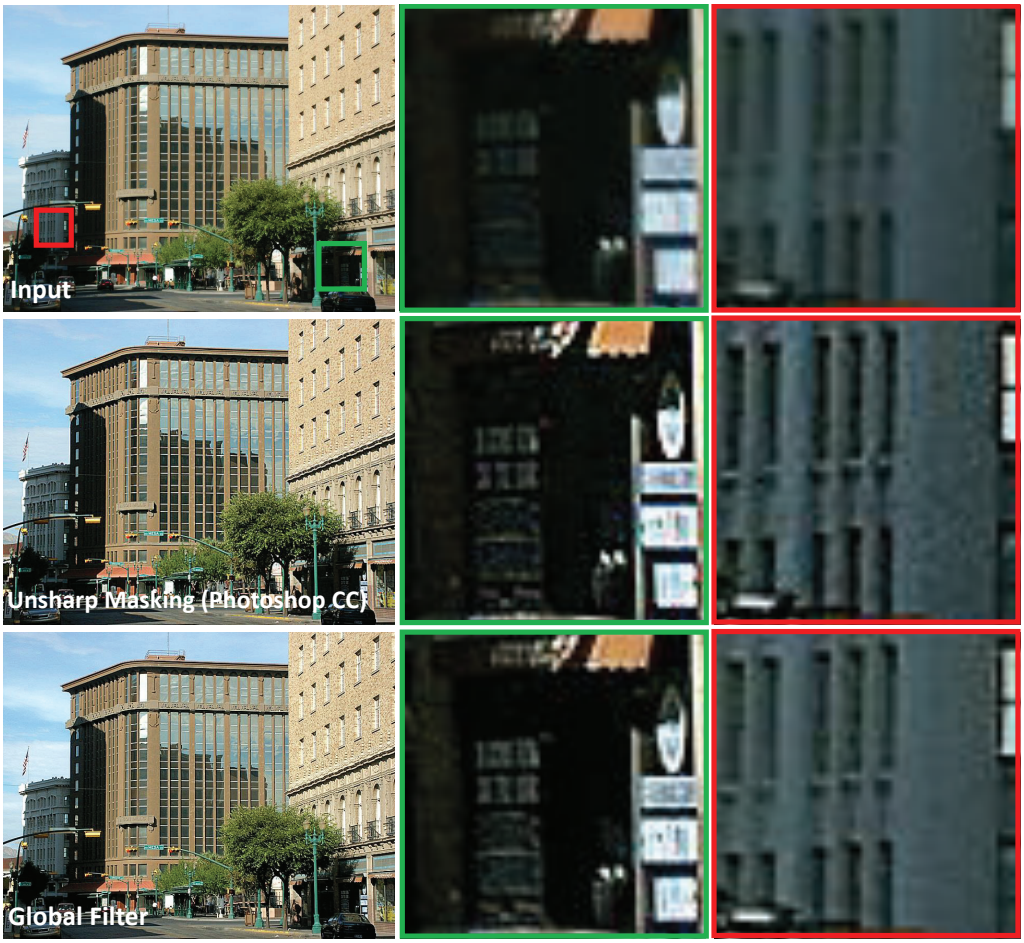


Figure 7.3: Input image is sharpened by unsharp masking tool in Photoshop CC to produce the reference image $\widetilde{\mathbf{y}}$. Then, the cost function given in (7.3) is solved to produce the global filter output. The estimated parameters are $\widehat{\boldsymbol{\alpha}} = [3.82, -9.48, 9.50, 1]^T$.

are $[3, 4, -4, 1]^T$. In this experiment, smart sharpen (Fig. 7.2) and unsharp masking (Fig. 7.3) of Photoshop CC are used to create the reference image $\widetilde{y}$. The global filter result is $\mathbf{V}_m f_{\widehat{\boldsymbol{\alpha}}}(\mathbf{S}_m)\mathbf{V}_m^T\mathbf{y}$ where $f_{\widehat{\boldsymbol{\alpha}}}(.)$ is the 3rd order polynomial with optimal coefficients $\widehat{\boldsymbol{\alpha}}$ obtained for each image pair. Visual comparison of the results from global filter and Photoshop CC shows that our filter better handles noise and JPEG artifacts. Although these examples might have no practical usage at this stage, training

the proposed framework for a large set of image pairs can be potentially very interesting. This could result in a unified filtering framework capable of imitating, and possibly improving, various filtering operators. We leave this for future work.

Bibliography

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *International Conference on Computer Vision (ICCV)*, pp. 836–846, January 1998. vi, xii, 6, 8, 24, 25, 30, 41, 92, 119

[2] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation (SIAM interdisciplinary journal)*, vol. 4, no. 2, pp. 490–530, 2005. vi, vii, viii, xii, 6, 7, 8, 24, 26, 30, 34, 35, 41, 44, 49, 64, 65, 66, 70, 74, 85, 88, 93, 96, 119

[3] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, February 2007. vi, vii, xii, 6, 7, 8, 9, 24, 26, 31, 34, 36, 40, 94, 119

[4] 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, pp. 1531–1549, Apr. 2010. vii, xii, 6, 15, 32, 34, 35, 36, 42

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*,

vol. 16, no. 8, pp. 2080–2095, August 2007. vii, viii, xii, 6, 15, 32, 34, 35, 36, 42, 65, 67, 68, 69, 70, 74, 88

[6] A. Polesel, G. Ramponi, and V. J. Mathews, "Image enhancement via adaptive unsharp masking," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 505–510, 2000. x, 93, 107, 108

[7] R. C. Bilcu and M. Vehvilainen, "Constrained unsharp masking for image enhancement," *Proceedings of International Conference on Image and Signal Processing*, pp. 10–19, 2008. x, 107, 108

[8] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 689–694, 2004. x, 111, 112

[9] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1221–1226, 2006. x, 115, 116

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004. xii, 30, 34, 65, 67

[11] Y. Tsin, V. Ramesh, and T. Kanade, "Statistical calibration of ccd imaging process," *Proceedings of IEEE International Conference on Computer Vision*, p. 480487, July 2001. 1

[12] J. R. Janesick, *Scientific Charge-Coupled Devices*. Bellingham,WA: SPIE, 2001. 3

[13] K. A. Jain, *Fundamentals of Digital Image Processing*. Cliffs, NJ: Prentice-Hall, 1989. 3

[14] F. J. Anscombe, "The transformation of Poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, December 1948. 3

[15] E. Nyström, "Über die praktische auflösung von linearn ingtegraglechungen mit anwendungen auf randwertaufgaben der potentialtheorie," *commentationes Physico-Mathematicae*, vol. 4, no. 15, pp. 1–52, April 1928. 4, 43, 45

[16] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, November 2003. 6

[17] D. D. Muresan and T. W. Parks, "Adaptive principal components and image denoising," *Proceedings of IEEE International Conference on Image Processing*, vol. 1, pp. 101–104, Sep. 2003. 6

[18] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006. 6, 42

[19] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1438–1451, July 2009. 6, 74

[20] ——, "Is denoising dead?" *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 895–911, April 2010. 6, 42, 67

[21] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2866–2878, Oct. 2006. 6

[22] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch-based image sequence restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1096–1102, Jun. 2007. 6

[23] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 106–128, 2013. 7, 8, 10, 14, 15, 16, 58, 60

[24] E. Seneta, *Non-negative Matrices and Markov Chains*. Springer Series in Statistics. Springer, 1981. 10

[25] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1991. 10

[26] P. Milanfar, "Symmetrizing smoothing filters," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, 2013. 10, 37, 50, 61

[27] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Transactions on Image Proc.*, vol. 17, no. 9, pp. 1540–1554, September 2008. 13, 24, 62

[28] X. Zhu and P. Milanfar, "Automatic parameter selection for denoising algorithms using a no-reference measure of image content," *IEEE Transactions on Image Processing*, vol. 19, pp. 3116–3132, 2010. 13, 66

[29] H. Talebi, X. Zhu, and P. Milanfar, "How to SAIF-ly boost denoising performance," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1470–1485, April 2013. 14

[30] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *The Annals of Statistics*, vol. 9, no. 6, pp. 1135–1151, November 1981. 14, 19, 62

[31] M. Van De Ville, D. Kocher, "Nonlocal means with dimensionality reduction and sure-based parameter selection," *IEEE Transactions on Image Processing*, vol. 20, no. 9, pp. 2683–2690, September 2011. 14

[32] H. Talebi and P. Milanfar, "Improving denoising filters by optimal diffusion," *Proceedings of International Conference on Image Processing (ICIP), Orlando*, 2012. 15

[33] J. Salmon and E. Le Pennec, "NL-Means and aggregation procedures," *Proceedings of International Conference on Image Processing (ICIP)*, pp. 2977–2980, 2009. 29

[34] A. S. Dalalyan and A. B. Tsybakov, "Aggregation by exponential weighting, sharp PAC-Bayesian bounds and sparsity," *Machine Learning*, vol. 72, no. 1-2, pp. 39–61, 2008. 29

[35] A. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," *European Conference on Computer Vision (ECCV)*, October 2012. 43

[36] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in NIPS 13*. MIT Press, 2001, pp. 682–688. 43, 48

[37] C. T. Baker, *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1977. 43

[38] A. Talwalkar, S. Kumar, and H. Rowley, "Large-scale manifold learning," *Computer Vision and Pattern Recognition (CVPR)*, 2008. 44, 48

[39] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the

Nyström method," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004. 44, 45, 48, 51

[40] R. Farbman, Z.and Fattal and D. Lischinski, "Diffusion maps for edge-aware image editing," *ACM Transactions on Graphics*, vol. 29, no. 6, p. 145, 2010. 44

[41] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 755–768, February 2014. 44, 56, 73

[42] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning." *Journal of Mach. Learn. Res.*, vol. 6, pp. 2153–2175, 2005. 48

[43] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *Journal of Machine Learning Research*, vol. 13, pp. 981–1006, 2012. 48, 49, 121, 122

[44] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009. 49

[45] G. Wahba, "The fast Monte-Carlo cross-validation and C_L procedures: Comments, new results and applications to image recovery problems-comments," *Comput. Stat.*, vol. 10, pp. 249–250, 1995. 62

[46] P. Chatterjee and P. Milanfar, "Practical bounds on image denoising: From estimation to information," *IEEE Transactions on Image Processing.*, vol. 20, no. 5, pp. 1221–1233, May 2011. 67, 74

[47] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948. 74

[48] E. Ordentlich, G. Seroussi, S. Verdu, M. Weinberger, and T. Weissman, "A discrete universal denoiser and its application to binary images," *Proceedings of International Conference on Image Processing (ICIP)*, 2003. 74

[49] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 5–28, 2005. 74

[50] H. Talebi and P. Milanfar, "Global denoising is asymptotically optimal," *Proceedings of International Conference on Image Processing (ICIP)*, 2014. 75

[51] ——, "Asymptotic performance of global denoising," *submitted to SIAM Journal on Imaging Sciences*, 2015. 75

[52] J. M. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. Cambridge University Press, 2004. 78

[53] K. Knopp, *Infinite Sequences and Series*. Dover Publication, 1956. 79, 89

[54] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, F. Warner, and S. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences*, vol. 102, no. 21, pp. 7426–7431, 2005. 81, 102

[55] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, August 2008. 92

[56] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decompo-

sition based on local extrema," *ACM Transactions on Graphics*, vol. 28, no. 5, December 2009. 92

[57] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters: edge-aware image processing with a Laplacian pyramid," *ACM Transactions on Graphics*, vol. 30, no. 4, July 2011. 92

[58] Z. Farbman, R. Fattal, and D. Lischinski, "Diffusion maps for edge-aware image editing," *ACM Transactions on Graphics*, vol. 29, no. 6, December 2010. 92, 94, 96

[59] X. An and F. Pellacini, "Appprop: all-pairs appearance-space edit propagation," *ACM Transactions on Graphics*, vol. 27, no. 3, August 2008. 92, 94

[60] P. Perona and J. Malik, "Scale-space and edge detection using anistropic diffusion," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 629–639, July 1990. 92

[61] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, July 2002. 92

[62] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Transactions on Graphics*, vol. 26, no. 3, July 2007. 92

[63] R. Fattal and S. Agrawala, M. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Transactions on Graphics*, vol. 26, no. 3, July 2007. 92

[64] A. Choudhury and G. G. Medioni, "Perceptually motivated automatic sharpness

enhancement using hierarchy of non-local means," *Proceedings of Internationa Conference on Computer Vision Workshops*, pp. 730–737, November 2011. 93

[65] B. Zhang and J. P. Allebach, "Adaptive bilateral filter for sharpness enhancement and noise removal," *IEEE Transactions on Image Processing*, vol. 17, pp. 664–678, 2008. 93

[66] X. Zhu and P. Milanfar, "Restoration for weakly blurred and strongly noisy images," *IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 103–109, 2011. 94

[67] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," *ACM Transactions on Graphics*, vol. 26, no. 3, July 2006. 94

[68] F. Pellacini and J. Lawrence, "Appwand: editing measured materials using appearance-driven optimization," *ACM Transactions on Graphics*, vol. 27, no. 3, August 2007. 94

[69] H. Talebi and P. Milanfar, "Non-local image editing," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4460–4473, October 2014. 96

[70] ——, "Global image editing using the spectrum of affinity matrices," *Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 771–774, December 2013. 96

[71] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU Press, 1996. 97

[72] J. A. Dieudonne, *Foundations of Modern Analysis*. New York: Academic press, 1960. 99

[73] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," *International Conference on Machine Learning*, 2000. 121

[74] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *The Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2002. 121

[75] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *The Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2003. 121

[76] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low-rank approximation and error analysis," *International Conference on Machine Learning*, pp. 1232–1239, 2008. 121, 122

[77] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang, "Matrix approximation and projective clustering via volume sampling," *Symposium on Discrete Algorithms*, 2006. 121, 122

[78] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister, "Image restoration using online photo collections," *International Conference on Computer Vision (ICCV)*, pp. 2217–2224, 2009. 124

[79] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of image enhancement," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1799–1806, 2010. 124

[80] J. C. Caicedo, A. Kapoor, and S. B. Kang, "Collaborative personalization of image enhancement," *Computer Vision and Pattern Recognition (CVPR)*, pp. 249–256, 2011. 124

[81] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," *Computer Vision and Pattern Recognition (CVPR)*, pp. 97–104, 2011. 124

[82] N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman, "Personal photo enhancement using example images," *ACM Transactions on Graphics*, vol. 29, no. 2, p. 12, 2010. 124

[83] F. Berthouzoz, W. Li, M. Dontcheva, and M. Agrawala, "A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations." *ACM Transactions on Graphics*, vol. 30, no. 5, p. 120, 2011. 124

[84] J. A. Guerrero-Colon and J. Portilla, "Deblurring-by-denoising using spatially adaptive Gaussian scale mixtures in overcomplete pyramids," *Proceedings of the International Conference on Image Processing (ICIP), Atlanta, GA*, pp. 625–628, October 2006.