

# Understanding Symmetric Smoothing Filters: A Gaussian Mixture Model Perspective

Stanley H. Chan, *Member, IEEE*, Todd Zickler, *Member, IEEE*, and Yue M. Lu, *Senior Member, IEEE*

**Abstract**—Many patch-based image denoising algorithms can be formulated as applying a smoothing filter to the noisy image. Expressed as matrices, the smoothing filters must be row normalized so that each row sums to unity. Surprisingly, if we apply a column normalization before the row normalization, the performance of the smoothing filter can often be significantly improved. Prior works showed that such performance gain is related to the Sinkhorn-Knopp balancing algorithm, an iterative procedure that symmetrizes a row-stochastic matrix to a doubly-stochastic matrix. However, a complete understanding of the performance gain phenomenon is still lacking.

In this paper, we study the performance gain phenomenon from a statistical learning perspective. We show that Sinkhorn-Knopp is equivalent to an Expectation-Maximization (EM) algorithm of learning a Gaussian mixture model of the image patches. By establishing the correspondence between the steps of Sinkhorn-Knopp and the EM algorithm, we provide a geometrical interpretation of the symmetrization process. This observation allows us to develop a new denoising algorithm called Gaussian mixture model symmetric smoothing filter (GSF). GSF is an extension of the Sinkhorn-Knopp and is a generalization of the original smoothing filters. Despite its simple formulation, GSF outperforms many existing smoothing filters and has a similar performance compared to several state-of-the-art denoising algorithms.

**Index Terms**—Non-local means, patch-based filtering, patch prior, Expectation-Maximization, doubly-stochastic matrix, symmetric smoothing filter

## I. INTRODUCTION

Smoothing filters are a class of linear and nonlinear operators that gains significant attentions in image denoising recently. The formulations of these operators are simple: Consider a noisy observation  $\mathbf{y} \in \mathbb{R}^n$  of a clean image  $\mathbf{z} \in \mathbb{R}^n$  corrupted by additive i.i.d. Gaussian noise. A smoothing filter is a matrix  $\mathbf{W} \in \mathbb{R}^{n \times n}$  that generates a denoised estimate  $\hat{\mathbf{z}}$  as

$$\hat{\mathbf{z}} = \mathbf{D}^{-1} \mathbf{W} \mathbf{y}, \quad (1)$$

where  $\mathbf{D} \stackrel{\text{def}}{=} \text{diag}\{\mathbf{W}\mathbf{1}\}$  is a diagonal matrix for normalization so that each row of  $\mathbf{D}^{-1}\mathbf{W}$  sums to unity. The formulation in (1) is very general, and many denoising algorithms are

smoothing filters, e.g., Gaussian filter [1], bilateral filter [2], non-local means (NLM) [3], locally adaptive regression kernel (LARK) [4], etc. Note that some of these filters are linear (e.g., Gaussian filter) whereas some are nonlinear (e.g. non-local means). There are interesting graph-theoretic interpretations of the smoothing filters [5]–[10], and there are also fast algorithms to compute the smoothing filters [11]–[16].

### A. Motivation: A Surprising Phenomenon

While smoothing filters work well for many denoising problems, it was observed in [17]–[19] that their performance can be further improved by modifying (1) as

$$\hat{\mathbf{z}} = \mathbf{D}_r^{-1} \mathbf{W} \mathbf{D}_c^{-1} \mathbf{y}, \quad (2)$$

where  $\mathbf{D}_c \stackrel{\text{def}}{=} \text{diag}\{\mathbf{W}^T \mathbf{1}\}$  is a diagonal matrix that normalizes the *columns* of  $\mathbf{W}$ , and  $\mathbf{D}_r \stackrel{\text{def}}{=} \text{diag}\{\mathbf{W} \mathbf{D}_c^{-1} \mathbf{1}\}$  is a diagonal matrix that normalizes the *rows* of  $\mathbf{W} \mathbf{D}_c^{-1}$ . In other words, we modify (1) by introducing a column normalization step before applying the row normalization.

Before discussing the technical properties of (1) and (2), we first provide some numerical results to demonstrate an interesting phenomenon. In Figure 1, we crop the center  $100 \times 100$  region of 10 standard clean images. We generate noisy observations by adding i.i.d. Gaussian noise of standard deviation  $\sigma = 20/255$  to each clean image. These noisy images are then denoised by (1) and (2), respectively. The weight matrix  $\mathbf{W}$  is chosen as the one defined in the non-local means (NLM) [3]. To ensure fair comparison, we choose the best parameter  $h_r$ , the range parameter in NLM, for both methods. The patch size is set as  $5 \times 5$  and the neighborhood search window is set as  $21 \times 21$ . The experiment is repeated for 20 independent Monte-Carlo trials to average out the randomness caused by different realizations of the i.i.d. Gaussian noise.

The results of this experiment are shown at the bottom of Figure 1. It is perhaps a surprise to see that (2), which is a simple modification of (1), improves the PSNR by more than 0.23 dB on average. Another puzzling observation is that if we repeatedly apply the column-row normalization, the PSNR does not always increase as more iterations are used. Figure 2 presents the result. In this experiment, we fix the NLM parameter  $h_r$  to its optimal value when using the column-row normalization, i.e., (2). For 5 out of the 10 images we tested, the PSNR values actually drop after the first column-row normalization.

The above experiment piqued our curiosity and led us to a basic question: Why would the column-row normalization improve the denoising performance? Insights gained from

S. H. Chan is with the School of Electrical and Computer Engineering, and the Department of Statistics, Purdue University, West Lafayette, IN 47907, USA. Email: stanleychan@purdue.edu.

T. Zickler and Y. M. Lu are with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA. E-mails: {zickler, yuelu}@seas.harvard.edu.

S. H. Chan completed part of this work at Harvard University in 2012-2014. This work was supported in part by the Croucher Foundation Postdoctoral Research Fellowship, and in part by the U.S. National Science Foundation under Grant CCF-1319140. Preliminary material in this paper was presented at the IEEE International Conference on Image Processing (ICIP), Quebec City, Sep 2015.



Smoothing Filter	Image No.									
	1	2	3	4	5	6	7	8	9	10
$D^{-1}\mathbf{W}$	30.54	31.42	27.41	26.81	27.89	27.01	30.54	29.16	28.41	29.02
$h_r$	$0.64\sigma$	$0.69\sigma$	$0.69\sigma$	$0.71\sigma$	$0.64\sigma$	$0.64\sigma$	$0.71\sigma$	$0.69\sigma$	$0.71\sigma$	$0.71\sigma$
$D_r^{-1}\mathbf{W}D_c^{-1}$	30.64	31.76	27.64	26.98	28.18	27.08	30.87	29.44	28.47	29.39
$h_r$	$0.71\sigma$	$0.77\sigma$	$0.77\sigma$	$0.77\sigma$	$0.71\sigma$	$0.71\sigma$	$0.79\sigma$	$0.77\sigma$	$0.77\sigma$	$0.79\sigma$
PSNR Improvement	+0.10	+0.34	+0.23	+0.17	+0.29	+0.07	+0.33	+0.28	+0.05	+0.37

Fig. 1: [Top]  $100 \times 100$  testing images. Each image is corrupted by i.i.d Gaussian noise of  $\sigma = 20/255$ . [Bottom] PSNR values of the denoised image using  $D^{-1}\mathbf{W}$  and  $D_r^{-1}\mathbf{W}D_c^{-1}$ , and the respective optimal NLM parameter  $h_r$ . All PSNR values in this table are averaged over 20 independent Monte-Carlo trials.

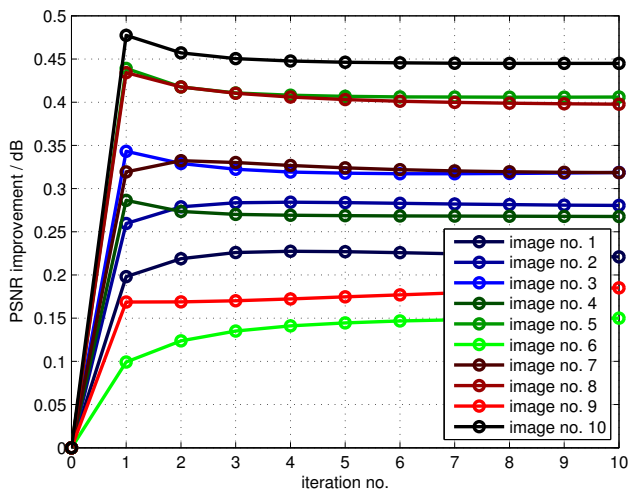


Fig. 2: Extension of the experiment shown in Figure 1. The PSNR values do not always increase as more Sinkhorn-Knopp iterations are used. The curves are averaged over 20 independent Monte-Carlo trials with different noise realizations.

a better understanding of the underlying mechanism could potentially lead to a more systematic procedure that can generalize the operations in (2) and further improve the denoising performance. The goal of this paper is to address this issue and propose a new algorithm.

### B. Sinkhorn-Knopp Balancing Algorithm

To the best of our knowledge, the above performance gain phenomenon was first discussed by Milanfar in [17], where it was shown that if we repeatedly apply the column-row normalization we would obtain an iterative procedure called the Sinkhorn-Knopp balancing algorithm [20], [21] (or Sinkhorn-Knopp, for short.) As illustrated in Algorithm 1, Sinkhorn-Knopp is a simple algorithm that repeatedly applies the column and row normalization until the smoothing filter converges. For example, the smoothing filter defined by (2) is the result of applying Sinkhorn-Knopp for one iteration.

Sinkhorn-Knopp has many interesting properties. First,

### Algorithm 1 Sinkhorn-Knopp Balancing Algorithm

---

Input:  $\mathbf{W}^{(0)}$   
**while**  $\|\mathbf{W}^{(t+1)} - \mathbf{W}^{(t)}\|_F > \text{tol}$  **do**  
 $D_c = \text{diag}\left\{(\mathbf{W}^{(t)})^T \mathbf{1}\right\}$  % Column Normalize  
 $D_r = \text{diag}\left\{\mathbf{W}^{(t)} D_c^{-1} \mathbf{1}\right\}$  % Row Normalize  
 $\mathbf{W}^{(t+1)} = D_r^{-1} \mathbf{W}^{(t)} D_c^{-1}$   
**end while**

---

when Sinkhorn-Knopp converges, the converging limit is a doubly-stochastic matrix — a symmetric non-negative matrix with unit column and row (also called a *symmetric* smoothing filter.) A doubly stochastic matrix has all of its eigenvalue’s magnitudes bounded in  $[0, 1]$  so that repeated multiplications always attenuate the eigenvalues [22]. Moreover, the estimate  $\hat{\mathbf{z}}$  formed by a doubly stochastic matrix is admissible in the sense that no other estimates are uniformly better [23].

To explain the performance gain, Milanfar [17] considered a notion called “effective degrees of freedom”, defined as

$$df = \sum_{j=1}^n \frac{\partial \hat{z}_j}{\partial y_j},$$

where  $\hat{z}_j$  is the  $j$ th pixel of the estimate and  $y_j$  is the  $j$ th pixel of the input.  $df$  measures how an estimator trades bias against variance. Larger values of  $df$  imply a lower bias but higher variance. Milanfar argued that the overall mean squared error, which is the sum of the bias and the variance, is reduced because one can prove that symmetric smoothing filters have high effective degrees of freedom. However, effective degrees of freedom is not easy to interpret. It will be more useful if we can geometrically describe the actions to which the column-row normalization are applying.

### C. Contributions

The present paper is motivated by our wish to further understand the mechanism behind the performance gain phenomenon. Our approach is to study an Expectation-Maximization (EM) algorithm for learning a Gaussian mixture

model (GMM.) By analyzing the E-step and the M-step of the EM algorithm, we find that the actions of the symmetrization is a type of data *clustering*. This observation echoes with a number of recent work that shows ordering and grouping of non-local patches are key to high-quality image denoising [24]–[26]. There are two contributions of this paper, described as follows.

First, we generalize the symmetrization process by reformulating the denoising problem as a maximum-a-posteriori (MAP) estimation under a Gaussian mixture model. We show that the original smoothing filter in (1), the one-step Sinkhorn-Knopp in (2), and the full Sinkhorn-Knopp (i.e., iterative applications of Sinkhorn-Knopp until convergence) are all sub-routines of the EM algorithm to learn the GMM. By showing that each method supersedes its preceding counterpart, we provide a possible explanation for the performance gain phenomenon.

Second, based on the analysis of the GMM, we propose a new denoising algorithm called the GMM symmetric smoothing filter (GSF). We show that GSF does not only subsume a number of smoothing filters, but also has a performance similar to some state-of-the-art denoising algorithms. We will discuss implementation and parameter selections for the GSF algorithm.

This paper is an extension of a conference article presented in [19]. In [19], the linkage between the GMM and the MAP denoising step was not thoroughly discussed. Specifically, the MAP was formulated as a weighted least squares step but the weights were indirectly obtained through a by-product of the EM algorithm. In this paper, we show that the quadratic cost function in the weighted least squares is indeed a surrogate function for solving the MAP problem. Therefore, minimizing the cost function of the weighted least squares is equivalent to minimizing an upper bound of the MAP objective function.

The rest of the paper is organized as follows. First, we provide a brief introduction to GMM and the EM algorithm in Section II. In Section III we discuss the generalization of different symmetrizations using the EM algorithm. The new GSF is discussed in Section IV and experimental results are shown in Section V. We conclude in Section VI.

## II. PRELIMINARIES

### A. Notations

Throughout this paper, we use  $n$  to denote the number of pixels in the noisy image, and  $k$  to denote the number of mixture components in the GMM model. To avoid ambiguity, we call a mixture component of a GMM as a cluster. Clusters are tracked using the running index  $i \in \{1, \dots, k\}$ , whereas patches (or pixels) are tracked using the running index  $j \in \{1, \dots, n\}$ . Without loss of generality, we assume that all pixel intensity values have been normalized to the range  $[0, 1]$ .

We use bold letters to denote vectors and the symbol  $\mathbf{1}$  to denote a constant vector of all ones. The vector  $\mathbf{x}_j \in \mathbb{R}^2$  represents the two-dimensional spatial coordinate of the  $j$ th pixel, and the vector  $\mathbf{y}_j \in \mathbb{R}^d$  represents a  $d$ -dimensional patch centered at the  $j$ th pixel of the noisy image  $\mathbf{y}$ . For a clean image  $\mathbf{z}$ , the  $j$ th patch is denoted by  $\mathbf{z}_j$ . A scalar  $y_j \in \mathbb{R}$  refers

TABLE I: Popular choices of the smoothing filter  $\mathbf{W}$ .

Filter	$W_{ij}$
Gaussian Filter [1]	$\exp \left\{ -\frac{\ \mathbf{x}_j - \mathbf{x}_i\ ^2}{2h_s^2} \right\}$
Bilateral Filter [2]	$\exp \left\{ -\left( \frac{\ \mathbf{x}_j - \mathbf{x}_i\ ^2}{2h_s^2} + \frac{(y_j - y_i)^2}{2h_r^2} \right) \right\}$
NLM [3]	$\exp \left\{ -\frac{\ \mathbf{y}_j - \mathbf{y}_i\ ^2}{2h_r^2} \right\}$
Spatially Regulated NLM [17]	$\exp \left\{ -\left( \frac{\ \mathbf{x}_j - \mathbf{x}_i\ ^2}{2h_s^2} + \frac{\ \mathbf{y}_j - \mathbf{y}_i\ ^2}{2h_r^2} \right) \right\}$
LARK [4]	$\exp \left\{ -\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^T \Sigma_i^{-1} (\mathbf{x}_j - \mathbf{x}_i) \right\}$

to the intensity value of the center pixel of  $\mathbf{y}_j$ . Therefore, a  $d$ -dimensional patch  $\mathbf{y}_j$  (assume  $d$  is an odd number) is a vector  $\mathbf{y}_j = [y_{j-(d-1)/2}, \dots, y_j, \dots, y_{j+(d-1)/2}]^T$ . To extract  $\mathbf{y}_j$  from the noisy image  $\mathbf{y}$ , we use a linear operator  $\mathbf{P}_j \in \mathbb{R}^{d \times n}$  such that  $\mathbf{y}_j = \mathbf{P}_j \mathbf{y}$ . For some smoothing filters, the spatial coordinate  $\mathbf{x}_j$  is used together with a patch  $\mathbf{z}_j$  (or  $\mathbf{y}_j$ ). Therefore, for generality we define a *generalized patch*  $\mathbf{p}_j \in \mathbb{R}^p$ , which could be  $\mathbf{x}_j$ ,  $\mathbf{z}_j$  (or  $\mathbf{y}_j$ ), or a concatenation of both:  $\mathbf{p}_j = [\mathbf{x}_j^T, \mathbf{z}_j^T]^T$  (or  $\mathbf{p}_j = [\mathbf{x}_j^T, \mathbf{y}_j^T]^T$ ).

### B. Smoothing Filters

The results presented in this paper are applicable to smoothing filters  $\mathbf{W}$  taking the following form:

$$W_{ij} = \kappa_i \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (3)$$

where  $\kappa_i \stackrel{\text{def}}{=} \sqrt{(2\pi)^p |\boldsymbol{\Sigma}_i|}$  is a normalization constant, and  $\mathcal{N}(\cdot)$  denotes a  $p$ -dimensional Gaussian:

$$\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \stackrel{\text{def}}{=} \frac{1}{\kappa_i} \exp \left\{ -\frac{1}{2} \|\mathbf{p}_j - \boldsymbol{\mu}_i\|_{\boldsymbol{\Sigma}_i^{-1}}^2 \right\}, \quad (4)$$

with mean  $\boldsymbol{\mu}_i \in \mathbb{R}^p$  and covariance matrix  $\boldsymbol{\Sigma}_i \in \mathbb{R}^{p \times p}$ . We note that (3) is general and covers a number of widely used filters as shown in Table I.

*Example 1 (Standard NLM):* For the standard NLM [3], we have  $\mathbf{p}_j = \mathbf{y}_j$ . The  $i$ th mean vector is  $\boldsymbol{\mu}_i = \mathbf{y}_i$ , i.e., the noisy patch is the mean vector. The  $i$ th covariance matrix is

$$\boldsymbol{\Sigma}_i = h_r^2 \mathbf{I}, \quad \text{for all } i, \quad (5)$$

where  $h_r$  is the NLM parameter.

In practice, to reduce computational complexity, a search window  $\Omega$  is often introduced so that neighboring patches are searched within  $\Omega$ . This is equivalent to multiplying an indicator function to the weight as

$$W_{ij} = \mathbb{I} \{ (\mathbf{x}_i - \mathbf{x}_j) \in \Omega \} \exp \left\{ -\frac{\|\mathbf{y}_j - \mathbf{y}_i\|^2}{2h_r^2} \right\}, \quad (6)$$

where  $\mathbb{I} \{ \mathbf{x} \in \Omega \} = 1$  if  $\mathbf{x} \in \Omega$ , and is zero otherwise.

*Example 2 (Spatially Regulated NLM):* As an alternative to the hard thresholding introduced by the indicator function

in (6), one can also consider the spatially regulated NLM [17]. In this case, we can define  $\mathbf{p}_j = [\mathbf{x}_j^T, \mathbf{y}_j^T]^T$ . The mean vector  $\boldsymbol{\mu}_i$  and the covariance matrices will consist of two parts:

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{(s)} \\ \boldsymbol{\mu}_i^{(r)} \end{bmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} h_s^2 \mathbf{I} & 0 \\ 0 & h_r^2 \mathbf{I} \end{bmatrix} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\text{NLM}}, \quad (7)$$

where  $h_s$  and  $h_r$  are the spatial and the range parameters, respectively. This leads to the weight

$$W_{ij} = \exp \left\{ -\frac{\|\mathbf{x}_j - \mathbf{x}_i\|^2}{2h_s^2} \right\} \exp \left\{ -\frac{\|\mathbf{y}_j - \mathbf{y}_i\|^2}{2h_r^2} \right\}. \quad (8)$$

It is not difficult to see that the spatially regulated NLM coincides with the standard NLM as  $h_s \rightarrow \infty$  and  $|\Omega| \rightarrow \infty$ . In fact, the former uses a soft search window and the latter uses a hard search window. From our experience, we find that the spatially regulated NLM typically has better performance than the standard NLM when the best choices of  $h_s$  and  $\Omega$  are used in either case. Therefore, in the rest of this paper we will focus on the spatially regulated NLM.

### C. Gaussian Mixture Model

The Gaussian mixture model (GMM) plays an important role in this paper. Consider a set of  $n$  generalized patches  $\mathcal{P} \stackrel{\text{def}}{=} \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$  where  $\mathbf{p}_j \in \mathbb{R}^p$ . We say that  $\mathcal{P}$  is generated from a Gaussian mixture model of  $k$  clusters if  $\mathbf{p}_j$  is sampled from the distribution

$$f(\mathbf{p}_j | \Theta) = \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (9)$$

where  $\pi_i \in \mathbb{R}$  is the weight of the  $i$ th cluster,  $\boldsymbol{\mu}_i \in \mathbb{R}^p$  is the mean vector, and  $\boldsymbol{\Sigma}_i \in \mathbb{R}^{p \times p}$  is the covariance matrix. For the purpose of analyzing smoothing filters in this paper, we shall assume that the covariance matrices  $\boldsymbol{\Sigma}_i$  are fixed according to the underlying smoothing filter. For example, in spatially regulated NLM we fix the covariance matrices as  $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{\text{NLM}}$ . When  $\boldsymbol{\Sigma}_i$ 's are fixed, we denote  $\Theta \stackrel{\text{def}}{=} \{\pi_i, \boldsymbol{\mu}_i\}_{i=1}^k$  as the GMM model parameters.

Learning the model parameters  $\Theta$  from  $\mathcal{P}$  is typically done using the Expectation-Maximization (EM) algorithm [27]. The EM algorithm consists of two major steps: the expectation step (E-step) and the maximization step (M-step). The E-step is used to compute the conditional expected log-likelihood, often called the  $Q$ -function. The M-step is used to maximize the  $Q$ -function by seeking the optimal parameters  $\Theta$ . The algorithm iterates until the log-likelihood converges. Since the EM algorithm is widely used, we skip the introduction and refer readers to [27] for a comprehensive tutorial. The EM algorithm for learning a GMM is summarized in Algorithm 2.

## III. GENERALIZATIONS OF SYMMETRIC FILTERS

In this section, we discuss how various symmetric smoothing filters are generalized by the EM algorithm. We begin by discussing how the GMM can be used for denoising.

---

**Algorithm 2** EM Algorithm for Learning a GMM with a known covariance matrix  $\boldsymbol{\Sigma}$  [27].

---

Input: Patches  $\mathcal{P} \stackrel{\text{def}}{=} \{\mathbf{p}_j\}_{j=1}^n$ , and the number of clusters  $k$ .

Output: Parameters  $\Theta = \{(\pi_i, \boldsymbol{\mu}_i)\}_{i=1}^k$ .

Initialize  $\pi_i^{(0)}, \boldsymbol{\mu}_i^{(0)}$  for  $i = 1, \dots, k$ , and set  $t = 0$ .

**while** not converge **do**

**E-step:** Compute, for  $i = 1, \dots, k$  and  $j = 1, \dots, n$

$$\gamma_{ij}^{(t)} = \frac{\pi_i^{(t)} \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma})}{\sum_{l=1}^k \pi_l^{(t)} \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_l^{(t)}, \boldsymbol{\Sigma})} \quad (10)$$

**M-step:** Compute, for  $i = 1, \dots, k$

$$\pi_i^{(t+1)} = \frac{1}{n} \sum_{j=1}^n \gamma_{ij}^{(t)} \quad (11)$$

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^n \gamma_{ij}^{(t)} \mathbf{p}_j}{\sum_{j=1}^n \gamma_{ij}^{(t)}} \quad (12)$$

**Update Counter:**  $t \leftarrow t + 1$ .

**end while**

---

### A. MAP Denoising Using GMM

We first specify the denoising algorithm. Given the noisy image  $\mathbf{y}$ , we formulate the denoising problem by using the maximum-a-posterior (MAP) approach:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\text{argmin}} \frac{\lambda}{2} \|\mathbf{z} - \mathbf{y}\|^2 - \sum_{j=1}^n \log f(\mathbf{p}_j | \Theta) \quad (13)$$

where the first term specifies the data fidelity with  $\lambda$  as a parameter. The second term, which is a sum of overlapping patches (thus are dependent), is called the expected patch log-likelihood (EPLL) [25]. Note that EPLL is a general prior that uses the expectation of the log-likelihood of overlapping patches. It is not limited to a particular distribution for  $f(\mathbf{p}_j | \Theta)$ , although in [25] the GMM was used. Note also that the patch  $\mathbf{p}_j$  in (13) is extracted from the optimization variable  $\mathbf{z}$ . Thus, by minimizing over  $\mathbf{z}$  we also minimize over  $\mathbf{p}_j$ .

Substituting (9) into (13), we obtain a GMM-based MAP denoising formulation

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\text{argmin}} \frac{\lambda}{2} \|\mathbf{z} - \mathbf{y}\|^2 - \sum_{j=1}^n \log \left( \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}) \right). \quad (14)$$

The computational challenge of (14) is the sum of exponentials inside the logarithm, which hinders closed-form solutions. In [25], Zoran and Weiss proposed to use a half quadratic splitting method to alternately minimize a sequence of subproblems, and select the mode of the GMM in each subproblem.

Our solution to handle the optimization problem in (14) is to use a surrogate function under the Majorization-Maximization framework [28]. The idea is to find a surrogate function

$h(\mathbf{p}_j, \mathbf{p}'_j | \Theta)$  such that

$$h(\mathbf{p}_j, \mathbf{p}'_j | \Theta) \geq -\log f(\mathbf{p}_j | \Theta), \quad \forall (\mathbf{p}_j, \mathbf{p}'_j), \quad (15)$$

$$h(\mathbf{p}_j, \mathbf{p}_j | \Theta) = -\log f(\mathbf{p}_j | \Theta). \quad (16)$$

If we can find such function  $h(\mathbf{p}_j, \mathbf{p}'_j | \Theta)$ , then the minimization in (13) can be relaxed to minimizing an upper bound

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \frac{\lambda}{2} \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n h(\mathbf{p}_j, \mathbf{p}'_j | \Theta). \quad (17)$$

For the GMM in (9), Zhang *et al.* [29] proposed one convenient surrogate function  $h(\mathbf{p}_j, \mathbf{p}'_j | \Theta)$  whose expression is given in Lemma 1.

*Lemma 1 (Surrogate function for GMM):* The function

$$h(\mathbf{p}_j, \mathbf{p}'_j | \Theta) \stackrel{\text{def}}{=} -\log f(\mathbf{p}'_j | \Theta) + \sum_{i=1}^k \gamma_{ij} \left( -\frac{1}{2} \|\mathbf{p}_j - \boldsymbol{\mu}_i\|_{\Sigma^{-1}}^2 + \frac{1}{2} \|\mathbf{p}'_j - \boldsymbol{\mu}_i\|_{\Sigma^{-1}}^2 \right), \quad (18)$$

where

$$\gamma_{ij} = \frac{\pi_i \mathcal{N}(\mathbf{p}'_j | \boldsymbol{\mu}_i, \Sigma)}{\sum_{l=1}^k \pi_l \mathcal{N}(\mathbf{p}'_j | \boldsymbol{\mu}_l, \Sigma)}, \quad (19)$$

is a surrogate function that satisfies (15) and (16).

*Proof:* See Appendix A of [29].  $\blacksquare$

*Remark 1:* The surrogate function  $h(\mathbf{p}_j, \mathbf{p}'_j | \Theta)$  requires an intermediate variable  $\mathbf{p}'_j$ . This variable can be chosen as  $\mathbf{p}'_j = [\mathbf{x}_j^T, \mathbf{y}_j^T]^T$ , i.e., the generalized patch using the noisy image  $\mathbf{y}$ . Thus,  $\gamma_{ij}$  is independent of  $\mathbf{p}_j$ .

Substituting the result of Lemma 1 into (17), we observe that (17) can be rewritten as

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \|\mathbf{p}_j - \boldsymbol{\mu}_i\|_{\Sigma^{-1}}^2, \quad (20)$$

where we dropped terms involving  $\mathbf{p}'_j$  as they are independent to the optimization variable  $\mathbf{z}$ . Note that (20) is a quadratic minimization, which is significantly easier than (14).

For the case of spatially regulated NLM, it is possible to further simplify (20) by recognizing that  $\mathbf{p}_j$  involves both spatial coordinate  $\mathbf{x}_j$  and patch  $\mathbf{z}_j$ . Therefore, by expressing  $\mathbf{p}_j = [\mathbf{x}_j^T, \mathbf{z}_j^T]^T$  and by defining  $\boldsymbol{\mu}_i$  using (7), (20) becomes

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \left\| \begin{bmatrix} \mathbf{x}_j \\ \mathbf{z}_j \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_i^{(s)} \\ \boldsymbol{\mu}_i^{(r)} \end{bmatrix} \right\|_{\Sigma^{-1}}^2.$$

In this minimization, we observe that since  $\mathbf{x}_j$  is not an optimization variable, it can be eliminated without changing the objective function. By using a patch extract operator  $\mathbf{P}_j$ , i.e.,  $\mathbf{z}_j = \mathbf{P}_j \mathbf{z}$ , we obtain the minimization

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \left\| \mathbf{P}_j \mathbf{z} - \boldsymbol{\mu}_i^{(r)} \right\|^2, \quad (P_1)$$

where we have absorbed the NLM parameter  $2h_r^2$  (the diagonal term of  $\Sigma$ ) into  $\lambda$ . Problem  $(P_1)$  is the main optimization of interest in this paper. We call  $(P_1)$  the *GMM Symmetric Smoothing Filter (GSF)*. In the literature, there are other GMM based denoising algorithms. Their connections to GSF will be

discussed in Section IV .

### B. Original Smoothing Filter

We now discuss the role of symmetrization by studying  $(P_1)$  and the EM algorithm for learning the GMM. To keep track of the iterations of the EM algorithm, we use the running index  $t$  denotes the iteration number. For consistency, we will focus on the spatially regulated NLM.

In spatially regulated NLM, the  $i$ th pixel of the denoised image is

$$\hat{z}_i = \frac{\sum_{j=1}^n W_{ij} y_j}{\sum_{j=1}^n W_{ij}}, \quad (21)$$

with  $W_{ij}$  defined in (8). To obtain (21) from  $(P_1)$ , we consider the following choices of parameters

$$\mathbf{P}_j \mathbf{z} = z_j, \quad \boldsymbol{\mu}_i^{(r)} = y_i, \quad \pi_i = 1/k, \quad \lambda = 0, \quad k = n, \\ \gamma_{ij} = \mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \Sigma_{\text{NLM}}). \quad (22)$$

In this case, since the quadratic objective in  $(P_1)$  is separable, the  $i$ th term becomes

$$\hat{z}_i = \underset{z_i}{\operatorname{argmin}} \sum_{j=1}^n \gamma_{ij} (z_i - y_j)^2 = \frac{\sum_{j=1}^n \gamma_{ij} y_j}{\sum_{j=1}^n \gamma_{ij}}, \quad (23)$$

which coincides with (21) as  $W_{ij} = \mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \Sigma_{\text{NLM}})$  because of (8).

It is important to study the conditions in (22). First, the patch extractor  $\mathbf{P}_j$  extracts a pixel  $z_j$  from  $\mathbf{z}$ . This step is necessary because NLM is a weighted average of individual pixels, even though the weights are calculated using patches. Accordingly, the mean vector  $\boldsymbol{\mu}_i$  is also a pixel  $y_i$  so that it matches with the optimization variable  $z_i$ . From a clustering perspective, we can interpret  $\boldsymbol{\mu}_i = y_i$  as having one cluster center for one pixel, i.e., every pixel has its own cluster. Clearly, this is a suboptimal configuration, and we will address it when we present the proposed method. We also note that in (22), the parameter  $\lambda$  is 0. What it means is that the data fidelity term is not used and only the EPLL prior is required to obtain the NLM result.

The most interesting observation in (22) is the choice of  $\gamma_{ij}$ . In fact, the  $\gamma_{ij}$  in (22) is only the numerator of (19) by assuming  $\pi_i = 1/k$ . If we let  $\Theta_i = (\pi_i, \boldsymbol{\mu}_i)$  be the  $i$ th model parameter of a GMM, then the physical meaning of (22) is a *conditional probability* of observing  $\mathbf{p}_j$  given that we pick the  $i$ th cluster, i.e.,  $\mathbb{P}(\mathbf{p}_j | \Theta_i)$ . In contrast, (19) is a *posterior probability* of picking the  $i$ th cluster given that we observe  $\mathbf{p}_j$ , i.e.,  $\mathbb{P}(\Theta_i | \mathbf{p}_j)$ . We will discuss this subtle difference more carefully in the next subsection when we discuss the one-step Sinkhorn-Knopp.

### C. One-step Sinkhorn-Knopp

In one-step Sinkhorn-Knopp, we recognize that the  $i$ th pixel of the denoised image is

$$\hat{z}_i = \frac{\sum_{j=1}^n \widetilde{W}_{ij} y_j}{\sum_{j=1}^n \widetilde{W}_{ij}} \quad \text{and} \quad \widetilde{W}_{ij} = \frac{W_{ij}}{\sum_{l=1}^n W_{lj}}. \quad (24)$$

This result can be obtained from  $(P_1)$  by letting

$$\begin{aligned} \mathbf{P}_j \mathbf{z} &= z_j, \quad \boldsymbol{\mu}_i^{(r)} = y_i, \quad \pi_i = 1/k, \quad \lambda = 0, \quad k = n, \\ \gamma_{ij} &= \frac{\mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}{\sum_{l=1}^k \mathcal{N}(\mathbf{p}_j | \mathbf{p}_l, \boldsymbol{\Sigma}_{\text{NLM}})}. \end{aligned} \quad (25)$$

Unlike the conditions posted by the original smoothing filter in (22), one-step Sinkhorn-Knopp defines  $\gamma_{ij}$  according to (19), or with some substitutions that yields (25).

As mentioned in the previous subsection, the  $\gamma_{ij}$  defined in (22) is a conditional probability whereas that in (25) is a posterior probability. The posterior probability, if we refer to the EM algorithm (see Algorithm 2), is in fact the E-step with initializations  $\pi_i^{(0)} = 1/k$  and  $\boldsymbol{\mu}_i^{(0)} = \mathbf{p}_i$ . For the original filter, the E-step is not executed because  $\gamma_{ij}$  is defined through (22). Therefore, from a clustering perspective, it is reasonable to expect a better denoising result from one-step Sinkhorn-Knopp than the original smoothing filter because the clustering is better performed.

To further investigate the difference between the conditional probability  $\mathbb{P}(\mathbf{p}_j | \Theta_i)$  and the posterior probability  $\mathbb{P}(\Theta_i | \mathbf{p}_j)$ , we adopt a graph perspective by treating each patch  $\mathbf{p}_j$  as a node, and  $\gamma_{ij}$  as the weight on the edge linking node  $i$  and node  $j$  [5]. In the original smoothing filter, the conditional probability  $\mathbb{P}(\mathbf{p}_j | \Theta_i)$  causes a ‘‘majority vote’’ effect, meaning that the parameter  $\Theta_i$  has a direct influence to every patch  $\{\mathbf{p}_j\}$  in the image. Therefore, if  $\Theta_i$  has many ‘‘weak friends’’, the sum of these ‘‘weak friends’’ can possibly alter the denoising result which would have been better obtained from a few ‘‘good friends’’.

In contrast, the one-step Sinkhorn-Knopp uses the posterior probability  $\mathbb{P}(\Theta_i | \mathbf{p}_j)$ . From Bayes rule, the posterior probability is related to the conditional probability by

$$\mathbb{P}(\Theta_i | \mathbf{p}_j) = \frac{\mathbb{P}(\mathbf{p}_j | \Theta_i) \mathbb{P}(\Theta_i)}{\mathbb{P}(\mathbf{p}_j)}.$$

Since  $\mathbb{P}(\Theta_i) = \pi_i$  and  $\pi_i = 1/k$ , we see that  $\mathbb{P}(\Theta_i | \mathbf{p}_j)$  is the ratio of  $\mathbb{P}(\mathbf{p}_j | \Theta_i)$  and  $\mathbb{P}(\mathbf{p}_j)$ .  $\mathbb{P}(\mathbf{p}_j)$  measures the popularity of  $\mathbf{p}_j$ . Thus, if  $\mathbf{p}_j$  is a popular patch (i.e., it is a ‘‘friend’’ of many), then the normalization  $\mathbb{P}(\mathbf{p}_j | \Theta_i) / \mathbb{P}(\mathbf{p}_j)$  is a way to balance out the influence of  $\mathbf{p}_j$ . Interpreted in another way, it is equivalent to say that if  $\Theta_i$  has many ‘‘weak friends’’, the influence of these ‘‘weak friends’’ should be reduced. Such intuition is coherent to many NLM methods that attempt to limit the number of nearby patches, e.g., [30], [31].

The definite answer to whether there is performance gain due to one-step Sinkhorn-Knopp is determined by the likelihood of obtaining ‘‘weak friends’’. This, in turn, is determined by the image content and the NLM parameter  $h_r$  which controls the easiness of claiming ‘‘friends’’. For example, if an image contains many textures of a variety of content and if  $h_r$  is large, then it is quite likely to obtain ‘‘weak friends’’. In this case, one-step Sinkhorn-Knopp will improve the denoising performance. On the other hand, if an image contains only a constant foreground and a constant background, then most patches are ‘‘good friends’’ already. Applying the one-step Sinkhorn-Knopp could possibly hurt the denoising performance. Figure 3 shows an example.



Image No.	11	12	13	14	15
Original	33.48	35.15	36.39	38.85	36.81
One-Step	32.98	34.67	35.59	38.36	36.43
PSNR Gain	-0.50	-0.47	-0.80	-0.48	-0.38

Fig. 3: Repeat of the experiment in Figure 1 using images with a constant foreground and a constant background. Optimal  $h_r$  is tuned for the original standard NLM and the one-step Sinkhorn-Knopp, respectively. Note that the PSNR gain are all negative.

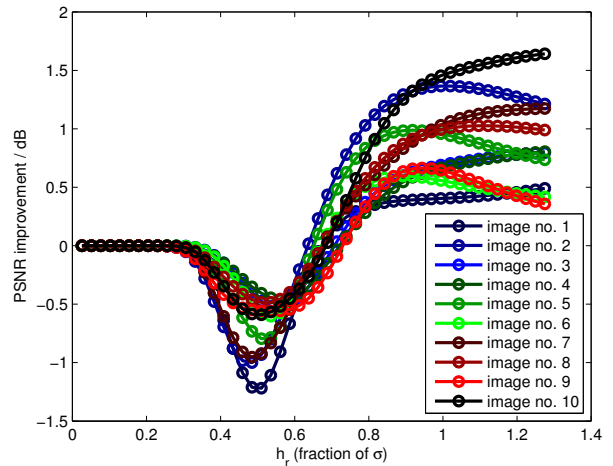


Fig. 4: Repeat the experiment in Figure 1 by plotting PSNR gain as a function of  $h_r$ . Note the consistent change of PSNR gain from negative to positive as  $h_r$  increases.

To further justify the above claim that the performance gain is caused by the likelihood of obtaining ‘‘weak friends’’, we consider the 10 images in Figure 1 by plotting the PSNR gain as a function of  $h_r$ . Our hypothesis is that for small  $h_r$ , the performance gain should be small or even negative because it is difficult for a patch to find its ‘‘friends’’. When  $h_r$  is large, the performance gain should become significant because many ‘‘weak friends’’ will be balanced out by the one-step Sinkhorn-Knopp. As shown in Figure 4, this is in fact the case: For  $h_r$  lies between 0 and certain threshold (around  $0.65\sigma$  where  $\sigma$  is the noise standard deviation), PSNR gain is always zero or negative. When  $h_r$  increases, PSNR gain becomes positive. The result is consistent for all 10 images we tested.

#### D. Full Sinkhorn-Knopp

The full Sinkhorn-Knopp (Algorithm 1) is an iterative algorithm that repeatedly applies the one-step Sinkhorn-Knopp until convergence. To analyze the full Sinkhorn-Knopp algorithm, we first recognize that the algorithm can be effectively described by two steps:

$$W_{ij} \leftarrow \frac{\widetilde{W}_{ij}}{\sum_{l=1}^n \widetilde{W}_{il}} \quad \text{and} \quad \widetilde{W}_{ij} \leftarrow \frac{W_{ij}}{\sum_{l=1}^n W_{lj}}, \quad (26)$$

where the first equation is a row normalization and the second equation is a column normalization. This pair of normalization can be linked to the EM algorithm in the following sense.

First, in the EM algorithm we fix the mixture weight  $\pi_i^{(t)}$  as  $\pi_i^{(t)} = 1/k$  for all clusters  $i = 1, \dots, k$  and all iteration numbers  $t = 1, \dots, t_{\max}$ . This step is necessary because the full Sinkhorn-Knopp does not involve mixture weights. Intuitively, setting  $\pi_i^{(t)} = 1/k$  ensures that all clusters have equal probability to be selected.

When  $\pi_i^{(t)}$  is fixed, the EM algorithm in Algorithm 2 has only two steps: Update of  $\gamma_{ij}^{(t)}$  in (10) and update of  $\mu_i^{(t)}$  in (12). Inspecting this pair of equations, we observe that (10) appears a column normalization whereas (12) appears a row normalization. However, since full Sinkhorn-Knopp does not have a mean vector  $\mu_i^{(t)}$ , we have to modify the EM algorithm in order to link the two. To this end, we modify (12) by defining a sequence  $\beta_{ij}^{(t)}$  such that

$$\mathbf{M}\text{-step} : \quad \beta_{ij}^{(t)} = \frac{\gamma_{ij}^{(t)}}{\sum_{l=1}^n \gamma_{il}^{(t)}}, \quad (27)$$

and modify (10) by updating of  $\gamma_{ij}^{(t)}$  via

$$\mathbf{E}\text{-step} : \quad \gamma_{ij}^{(t)} = \frac{\beta_{ij}^{(t)}}{\sum_{l=1}^k \beta_{lj}^{(t)}}. \quad (28)$$

Under this setting, (27)-(28) becomes exactly (26).

It is important to understand the difference between the original EM algorithm using (10)-(12) and the modified EM algorithm using (27)-(28). In the M-step of the modified EM algorithm, the mean vector  $\mu_i^{(t)}$  is absent. However,  $\mu_i^{(t)}$  is still updated, though implicitly, because

$$\mu_i^{(t)} = \frac{\sum_{j=1}^n \gamma_{ij}^{(t)} \mathbf{p}_j}{\sum_{l=1}^n \gamma_{il}^{(t)}} \stackrel{(a)}{=} \sum_{j=1}^n \beta_{ij}^{(t)} \mathbf{p}_j, \quad (29)$$

where (a) follows from (27). Therefore,  $\beta_{ij}^{(t)}$  are the coefficients that form  $\mu_i^{(t)}$  through a linear combination of  $\mathbf{p}_j$ 's.

In the E-step of the modified EM algorithm, since  $\mu_i^{(t)}$  is absent, one cannot compute the conditional probability  $\mathcal{N}(\mathbf{p}_j | \mu_i^{(t)}, \Sigma)$  and hence the posterior probability in (10). To resolve this issue, we recognize that since  $\sum_{j=1}^n \beta_{ij}^{(t)} = 1$  and  $\beta_{ij}^{(t)} \geq 0$  by definition, one possibility is to replace  $\mathcal{N}(\mathbf{p}_j | \mu_i^{(t)}, \Sigma)$  by  $\beta_{ij}^{(t)}$ . Such approximation is physically interpretable because  $\beta_{ij}^{(t)}$  is the coefficient for the mean vector as shown in (29). Thus,  $\beta_{ij}^{(t)}$  having larger values will have a larger contribution to forming  $\mu_i^{(t)}$ . In this perspective,  $\beta_{ij}^{(t)}$  is performing a similar role as  $\mathcal{N}(\mathbf{p}_j | \mu_i^{(t)}, \Sigma)$ . Practically, we observe that  $\beta_{ij}^{(t)}$  is a good approximation when there are only a few distinctive clusters. This can be understood as that while individual  $\beta_{ij}^{(t)}$  may not be accurate, averaging within a few large clusters can reduce the discrepancy between  $\beta_{ij}^{(t)}$  and  $\mathcal{N}(\mathbf{p}_j | \mu_i^{(t)}, \Sigma)$ .

The fact that the full Sinkhorn-Knopp does not resemble a complete EM algorithm offers some insights into the perfor-

mance gain phenomenon. Recall from the results in Figure 2, we observe that the first Sinkhorn-Knopp iteration always increases the PSNR except the artificial images we discussed in Section III.C. This can be interpreted as that the clustering is properly performed by the EM algorithm. However, as more Sinkhorn-Knopp iterations are performed, some images show reduction in PSNR, e.g., images 3, 4, 8, 10. A close look at these images suggests that they contain complex texture regions that are difficult to form few but distinctive clusters. In this case, the approximation of  $\mathcal{N}(\mathbf{p}_j | \mu_i^{(t)}, \Sigma)$  by  $\beta_{ij}^{(t)}$  is weak and hence the denoising performance drops.

### E. Summary

To summarize our findings, we observe that the performance of the normalization is related to how the EM algorithm is being implemented. A summary of these findings is shown in Table II. For all the three algorithms we considered: the original filter, the one-step Sinkhorn-Knopp, and the full Sinkhorn-Knopp algorithm, the EM algorithm is not completely performed or in-properly configured. For example, setting  $k = n$  causes excessive number of clusters and should be modified to  $k < n$ ; the MAP parameter  $\lambda$  is always 0 and should be changed to a positive value to utilize the data fidelity term in  $(P_1)$ ; the E-step and the M-step are not performed as it should be. Therefore, in the following section we will propose a new algorithm that completely utilizes the EM steps for problem  $(P_1)$ .

## IV. GMM SYMMETRIC SMOOTHING FILTER

The proposed denoising algorithm is called the Gaussian Mixture Model Symmetric Smoothing Filter (GSF). The overall algorithm of GSF consists of two steps:

- Step 1: Estimate the GMM parameter  $\mu_i^{(r)}$  and  $\gamma_{ij}$  from the noisy image the by EM algorithm.
- Step 2: Solve Problem  $(P_1)$ , which has a closed form solution.

In the followings we discuss how GSF is implemented.

### A. Closed Form Solution of GSF

First of all, we recall that since  $(P_1)$  is a quadratic minimization, it is possible to derive a closed form solution by considering the first order optimality condition, which yields a normal equation

$$\left( \sum_{j=1}^n \mathbf{P}_j^T \mathbf{P}_j + \lambda \mathbf{I} \right) \hat{\mathbf{z}} = \sum_{j=1}^n \mathbf{P}_j^T \mathbf{w}_j + \lambda \mathbf{y}, \quad (30)$$

where the vector  $\mathbf{w}_j$  is defined as

$$\mathbf{w}_j \stackrel{\text{def}}{=} \sum_{i=1}^n \gamma_{ij} \mu_i^{(r)}. \quad (31)$$

Equations (30)-(31) has a simple interpretation: The intermediate vector  $\mathbf{w}_j$  is a weighted average of the mean vectors  $\{\mu_i^{(r)}\}_{i=1}^k$ . These  $\{\mathbf{w}_j\}_{j=1}^n$  represent a collection of (denoised) overlapping patches. The operation  $\mathbf{P}_j^T$  on the right hand side of (30) aggregates these overlapping patches, similar to the

TABLE II: Generalization and comparisons using EM algorithm for learning GMM with a known  $\Sigma$ .

		Original Filter [3]	One-step Sinkhorn-Knopp [18]	Full Sinkhorn-Knopp [21]	GSF (Proposed)
No. Clusters		$k = n$	$k = n$	$k = n$	$k < n$ (cross-validation)
Initialization	$\gamma_{ij}^{(0)}$ $\pi_i^{(0)}$ $\mu_i^{(0)}$	$\mathcal{N}(\mathbf{p}_j   \mu_i^{(0)}, \Sigma_{\text{NLM}})$ $1/k$ $\mathbf{p}_i$	N/A $1/k$ $\mathbf{p}_i$	N/A $1/k$ $\mathbf{p}_i$	N/A $1/k$ randomly picked $\mathbf{p}_i$
E-step	Update $\gamma_{ij}^{(t)}$	×	✓	✓	✓
M-step	Update $\pi_i^{(t)}$ Update $\mu_i^{(t)}$	×	×	×	✓
No. Iterations		0	1	Many	Many
Denoising Parameters	$\lambda$ $\mathbf{P}_j \mathbf{z}$	0 $z_j$	0 $z_j$	0 $z_j$	by SURE $\mathbf{p}_j$

aggregation step in BM3D [32]. The addition of  $\lambda \mathbf{y}$  regulates the final estimate by adding a small amount of fine features, depending on the magnitude of  $\lambda$ .

In order to use (30)-(31), we must resolve two technical issues related to the EM algorithm and Problem ( $P_1$ ): (i) How to determine  $\lambda$ ; (ii) How to determine  $k$ .

### B. Parameter $\lambda$

Ideally,  $\lambda$  should be chosen as the one that minimizes the mean squared error (MSE) of the denoised image. However, in the absence of the ground truth, MSE cannot be calculated directly. To alleviate this difficulty, we consider the Stein's Unbiased Risk Estimator (SURE) [33], [34]. SURE is a consistent and unbiased estimator of the MSE. That is, SURE converges to the true MSE as the number of observations grows. Therefore, when there are sufficient number of observed pixels (which is typically true for images), minimizing the SURE is equivalent to minimizing the true MSE.

In order to derive SURE for our problem, we make an assumption about the boundary effect of  $\mathbf{P}_j$ .

*Assumption 1:* We assume that the patch-extract operator  $\{\mathbf{P}_j\}_{j=1}^n$  satisfies the following approximation:

$$\sum_{j=1}^n \mathbf{P}_j^T \mathbf{P}_j = d\mathbf{I}. \quad (32)$$

We note that Assumption 1 only affects the boundary pixels and not the interior pixels. Intuitively, what Assumption 1 does is to require that the boundary pixels of the image are periodically padded instead of zero-padded. In the image restoration literature, periodic boundary padding is common when analyzing deblurring methods, e.g., [35].

Under Assumption 1, we can substitute (32) into (30) and take the matrix inverse. This would yield

$$\hat{\mathbf{z}}(\lambda) = \frac{d}{d+\lambda} \mathbf{u} + \frac{\lambda}{d+\lambda} \mathbf{y}, \quad (33)$$

where

$$\mathbf{u} \stackrel{\text{def}}{=} \frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \mathbf{w}_j. \quad (34)$$

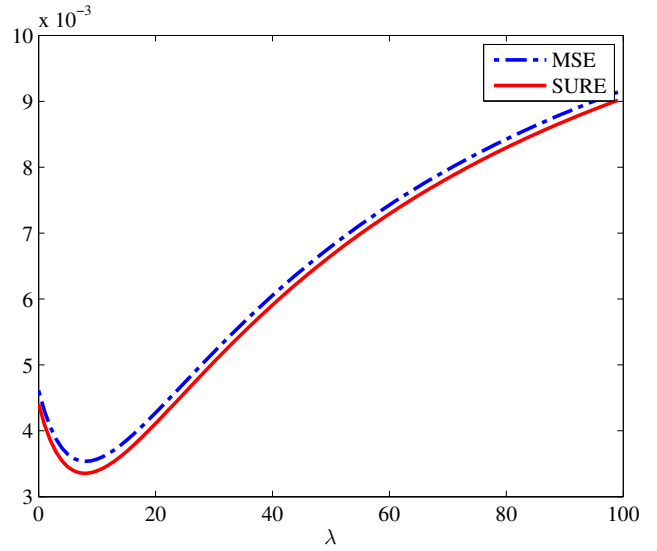


Fig. 5: Comparison between SURE and the ground truth MSE of a denoising problem.

Then, we can derive the SURE of  $\hat{\mathbf{z}}$  as follows.

*Proposition 1:* Under Assumption 1, the SURE of  $\hat{\mathbf{z}}(\lambda)$  is

$$\text{SURE}(\lambda) = -\sigma^2 + \hat{\sigma}^2 \left( \frac{d}{d+\lambda} \right)^2 + \frac{2\sigma^2}{n} \left( \frac{\text{div}(\mathbf{u})d + n\lambda}{d+\lambda} \right), \quad (35)$$

where  $\hat{\sigma}^2 \stackrel{\text{def}}{=} \frac{1}{n} \|\mathbf{u} - \mathbf{y}\|^2$ , and

$$\text{div}(\mathbf{u}) \stackrel{\text{def}}{=} \mathbf{1}_{n \times 1}^T \left( \frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \left( \sum_{i=1}^k \gamma_{ij} \left( \frac{\sum_{j=1}^n \gamma_{ij} \mathbf{e}_j}{\sum_{j=1}^n \gamma_{ij}} \right) \right) \right), \quad (36)$$

where  $\mathbf{e}_j \in \mathbb{R}^d$  is the  $j$ th standard basis.

*Proof:* See Appendix B. ■

The SURE given in (35) is a one-dimensional function in  $\lambda$ . The minimizer can be determined in closed-form.

*Corollary 1:* The optimal  $\lambda$  that minimizes  $\text{SURE}(\lambda)$  is

$$\lambda^* = \max \left( d \left( \left( \frac{\hat{\sigma}^2}{\sigma^2} \right) \left( \frac{n}{n - \text{div}(\mathbf{u})} \right) - 1 \right), 0 \right). \quad (37)$$



*Proof:* (35) is a differentiable function in  $\lambda$ . Therefore, the minimizer can be determined by considering the first order optimality and set the derivative of  $\text{SURE}(\lambda)$  to zero. The projection operator  $\max(\cdot, 0)$  is placed to ensure that  $\lambda^* \geq 0$ . ■

*Example 3:* To demonstrate the effectiveness of SURE, we show a typical MSE and a typical SURE curve of a denoising problem. In this example, we consider a  $128 \times 128$  image ‘‘Baboon’’, with noise standard deviation of  $\sigma = 30/255$ . The non-local means parameters are  $h_r = \sigma$  and  $h_s = 10$ . The number of clusters is  $k = 50$ , and the patch size is  $5 \times 5$ . The results are shown in Figure 5, where we observe that the SURE curve and the true MSE curve are very similar. In fact, the minimizer of the true MSE is  $\lambda = 8.0080$  with a PSNR of 24.5143dB whereas the minimizer of SURE is  $\lambda = 7.9145$  with a PSNR of 24.5141dB.

*Remark 2:* Careful readers may notice that in (36), we implicitly assume that  $\gamma_{ij}$  is independent of  $\mathbf{y}_j$ . This implicit assumption is generally not valid if  $\gamma_{ij}$  is learned from  $\mathbf{y}$ . However, in practice, we find that if we feed the EM algorithm with some initial estimate (e.g., by running the algorithm with  $\lambda = 0$ ), then the dependence of  $\gamma_{ij}$  from  $\mathbf{y}_j$  becomes negligible.

### C. Number of Clusters $k$

The number of clusters  $k$  is another important parameter. We estimate  $k$  based on the concept of cross validation [36].

Our proposed cross-validation method is based on comparing the estimated covariance with  $\Sigma_{\text{NLM}}$ . More specifically, we compute the estimated covariance

$$\hat{\Sigma}_i = \frac{\sum_{j=1}^n \gamma_{ij} (\mathbf{p}_j - \boldsymbol{\mu}_i) (\mathbf{p}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^n \gamma_{ij}}, \quad (38)$$

where  $\boldsymbol{\mu}_i = [\boldsymbol{\mu}_i^{(s)}, \boldsymbol{\mu}_i^{(r)}]^T$  is the mean returned by the EM algorithm, and  $\gamma_{ij} = \pi_{ij}^{(\infty)}$  is the converged weight. Then, we compute the ratio of the deviation

$$\delta_i(k) = \frac{1}{d} \text{Tr} \left\{ \Sigma_{\text{NLM}}^{-1} \hat{\Sigma}_i \right\}. \quad (39)$$

Ideally, if  $\hat{\Sigma}_i = \Sigma_{\text{NLM}}$ , then by (39) we have  $\delta_i(k) = 1$ . However, if the  $i$ th estimated Gaussian component has a radius significantly larger than  $h_r$  (or,  $h_s$  for the spatial components), then the covariance  $\hat{\Sigma}_i$  would deviate from  $\Sigma_{\text{NLM}}$  and hence  $\delta_i(k) > 1$ . Conversely, if the  $i$ th estimated Gaussian component has a radius significantly smaller than  $h_r$ , then we will have  $\delta_i(k) < 1$ . Therefore, the goal of the cross validation is to find a  $k$  such that  $\delta_i(k)$  is close to 1.

To complete the cross-validation setup, we average  $\delta_i(k)$  over all  $k$  clusters to obtain an averaged ratio

$$\delta(k) = \frac{1}{k} \sum_{i=1}^k \delta_i(k). \quad (40)$$

The parenthesis ( $k$ ) in (40) emphasizes that both  $\delta(k)$  and  $\delta_i(k)$  are functions of  $k$ . With (40), we seek the root  $k$  of the equation  $\delta(k) = 1$ .

The root finding process for  $\delta(k) = 1$  can be performed using the secant method. Secant method is an extension of

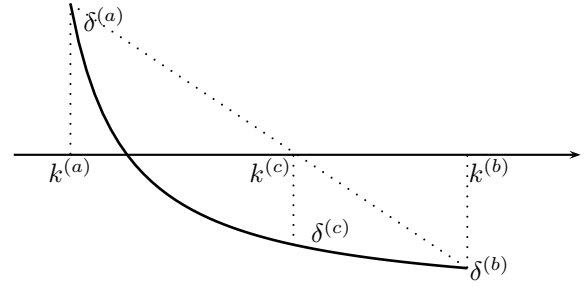


Fig. 6: Illustration of the secant method. Given  $k^{(a)}$  and  $k^{(b)}$ , we compute  $k^{(c)}$  according to the slope defined by the line linking  $\delta^{(a)}$  and  $\delta^{(b)}$ .

---

### Algorithm 3 Cross Validation to Determine $k$

---

Input:  $k^{(a)}$  and  $k^{(b)}$  such that  $\delta^{(a)} > 1$  and  $\delta^{(b)} < 1$ .  
Output:  $k^{(c)}$ .

```

while  $|k^{(a)} - k^{(c)}| > \text{tol}$  and  $|k^{(b)} - k^{(c)}| > \text{tol}$  do
    Compute  $k^{(c)}$  according to (41).
    Compute  $\delta^{(c)} \stackrel{\text{def}}{=} \delta(k^{(c)})$  according to (40).
    if  $\delta(k^{(c)}) > 1$  then
         $k^{(a)} \leftarrow k^{(c)}$ ;  $\delta^{(a)} \leftarrow \delta^{(c)}$ .
    else
         $k^{(b)} \leftarrow k^{(c)}$ ;  $\delta^{(b)} \leftarrow \delta^{(c)}$ .
    end if
end while

```

---

the bisection method in which the bisection step size (i.e.,  $1/2$ ) is now replaced by an adaptive step size determined by the local derivative of the function. Let  $k^{(a)}$  and  $k^{(b)}$  be two number of clusters, and  $\delta^{(a)}$  and  $\delta^{(b)}$  be the corresponding cross-validation scores, i.e.,  $\delta^{(a)} = \delta(k^{(a)})$ . If  $\delta^{(a)} > 1$  and  $\delta^{(b)} < 1$ , the secant method computes the new  $k$  as

$$k^{(c)} = \frac{k^{(a)}(\delta^{(b)} - 1) - k^{(b)}(\delta^{(a)} - 1)}{\delta^{(b)} - \delta^{(a)}}. \quad (41)$$

If  $\delta(k^{(c)}) > 1$ , then we replace  $k^{(a)}$  by  $k^{(c)}$ ; Otherwise, we replace  $k^{(b)}$  by  $k^{(c)}$ . The process repeats until the  $|k^{(a)} - k^{(c)}| < \text{tol}$  and  $|k^{(b)} - k^{(c)}| < \text{tol}$ . A pictorial illustration of the secant method is shown in Figure 6. A pseudo code is given in Algorithm 3.

*Example 4:* To verify the effectiveness of the proposed cross validation scheme, we consider a  $128 \times 128$  ‘‘House’’ image with noise  $\sigma = 60/255$ . The patch size is  $5 \times 5$ ,  $h_r = \sigma$ , and  $h_s = 10$ . Figure 7 shows the PSNR value of the denoised image and the corresponding cross validation score  $\delta(k)$  as a function of  $k$ . For this experiment, the maximum PSNR is achieved at  $k = 144$ , where PSNR = 26.0257dB. Using the cross-validation score  $\delta(k)$ , we find that  $\delta(k)$  is closest to 1 when  $k = 130$ . The corresponding PSNR value is 25.9896dB, which is very similar to the true maximum PSNR.

## V. EXPERIMENTS

In this section, we present additional simulation results to evaluate the proposed GSF.

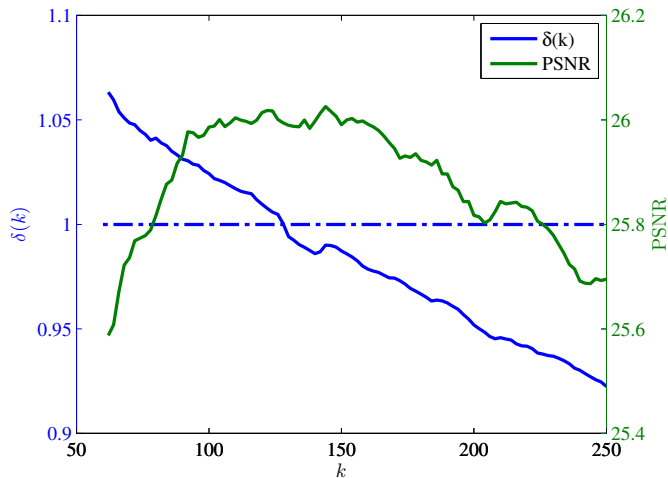


Fig. 7: Comparison between the cross validation score  $\delta(k)$  and the true PSNR value as a function of  $k$ . The horizontal dashed line indicates the intersection at  $\delta(k) = 1$ .

#### A. Experiment Settings

We consider 10 testing images, each of which is resized to  $128 \times 128$  (so  $n = 16384$ ) for computational efficiency. The noise standard deviations are set as  $\sigma \in \{20/255, 40/255, 60/255, 80/255, 100/255\}$ . Several existing denoising algorithms are studied, namely the NLM [3], One-step Sinkhorn-Knopp [18], BM3D [32], EPLL [25], Global image denoising (GLIDE) [12], NL-Bayes [37], and PLE [38]. The parameters of the methods are configured as shown in Table III.

For NLM and One-step Sinkhorn-Knopp (One-step, in short), we use the spatially regulated version due to its better performance over the standard NLM. We implement the algorithms by setting the patch size as  $5 \times 5$  (i.e.,  $d = 25$ ). The parameters are  $h_s = 10$  and  $h_r = \sigma\sqrt{d}$ . The full Sinkhorn-Knopp algorithm is implemented using GLIDE [12], where the source code is downloaded from the author’s website<sup>1</sup>. Default settings of GLIDE are used in our experiment.

For the proposed GSF, we keep the same settings as NLM except for the intensity parameter  $h_r$ , where we set  $h_r = \sigma$ . The omission of the factor  $\sqrt{d}$  is due to the fact that each Gaussian component is already a  $d$ -dimensional multivariate distribution. It is therefore not necessary to normalize the distance  $\|\mathbf{y}_i - \mathbf{y}_j\|^2$  by the factor  $d$ .

For BM3D, EPLL, NL-Bayes, we downloaded the original source code from the author’s website<sup>2,3,4</sup>. For PLE, we modified an inpainting version of the source code provided by the authors. Default settings of these algorithms are used.

Among these methods, we note that EPLL is an external denoising algorithm where a Gaussian mixture is learned from a collection of 2 million clean patches. All other methods (including GSF) are single image denoising algorithms.

TABLE III: Configurations of Methods

Method	Configuration
NLM [3]	Patch size $5 \times 5$ , $h_s = 10$ , $h_r = \sigma\sqrt{d}$
One-step [18]	Patch size $5 \times 5$ , $h_s = 10$ , $h_r = \sigma\sqrt{d}$
GSF (Ours)	Patch size $5 \times 5$ , $h_s = 10$ , $h_r = \sigma$
GLIDE [12]	Default settings. Pilot estimate uses NLM.
NL-Bayes [37]	Base mode. Default settings.
PLE [38]	Default settings. Default initializations.
BM3D [32]	Default settings.
EPLL [25]	Default settings. External Database.

#### B. Comparison with NLM, One-step and Full Sinkhorn-Knopp

The overall results of the experiment are shown in Table VI. We first compare the PSNR values of GSF with NLM, One-step and full Sinkhorn-Knopp.

In Table IV we show the average PSNR over the 10 testing images. In this table, we observe that on average One-step has a higher PSNR than NLM by 0.12dB to 1.12dB, with more significant improvements at low noise levels. This implies that the “grouping” action by the column normalization becomes less influential when noise increases. Moreover, if we compare GSF with NLM and One-step, we observe that the PSNR gain is even larger. Even at a high noise level (e.g.,  $\sigma = 80/255$  or  $\sigma = 100/255$ ), the average gain from NLM is 2.5dB or more.

TABLE IV: PSNR comparison with different noise level  $\sigma$ . Results are averaged over 10 testing images.

$\sigma$	NLM (PSNR <sub>1</sub> )	One-Step (PSNR <sub>2</sub> )	Ours (PSNR <sub>3</sub> )	PSNR <sub>2</sub> - PSNR <sub>1</sub>	PSNR <sub>3</sub> - PSNR <sub>1</sub>
20	25.59	26.71	28.89	+1.12	+3.30
40	21.97	22.53	25.38	+0.56	+3.41
60	20.33	20.63	23.60	+0.30	+3.27
80	19.46	19.64	22.39	+0.18	+2.92
100	18.97	19.09	21.47	+0.12	+2.50

Besides studying the trend of PSNR as a function of  $\sigma$ , it is also interesting to compare the PSNR when we increase the spatial parameter  $h_s$ . In Table V, we show the PSNR improvement when we use different  $h_s \in \{5, 10, 20, 50, 100\}$  for a  $128 \times 128$  image. The results show that when  $h_s$  increases, the PSNR improvement also increases. One reason is that in (7), the spatial parameter  $h_s$  controls the diagonal bandwidth of the smoothing filter  $\mathbf{W}$ . That is, a small  $h_s$  leads to a banded diagonal  $\mathbf{W}$  with small bandwidth. In the limit when  $h_s \rightarrow 0$ ,  $\mathbf{W}$  will become a diagonal matrix, and hence is immune to any column normalization. Therefore, the effectiveness of the column normalization in the One-step depends on how large  $h_s$  is.

The full Sinkhorn-Knopp algorithm is implemented using GLIDE [12]. GLIDE consists of multiple steps: It first determines the weight matrix, followed by a full Sinkhorn-Knopp algorithm that symmetrizes the weight matrix. Then, it incorporates an estimator to optimally determine the number of non-zero eigenvalues and the power of eigenvalues of the smoothing filter. GLIDE can use any denoising result as its pilot estimate. For the fairness of the experiment we follow the default setting of GLIDE and use the standard NLM as the

<sup>1</sup>GLIDE: <https://users.soe.ucsc.edu/~htalebi/GLIDE.php>

<sup>2</sup>BM3D: <http://www.cs.tut.fi/~foi/GCF-BM3D/>

<sup>3</sup>EPLL: <http://people.csail.mit.edu/danielzoran/>

<sup>4</sup>NL-Bayes: <http://www.ipol.im/pub/art/2013/16/>

TABLE VI: Denoising results of Standard NLM [3], One-step Sinkhorn-Knopp [18], BM3D [32], EPLL [25], Global image denoising [12], and the proposed GSF.

	NLM [3]	OneStep [18]	GSF (ours)	GLIDE [12]	Bayes [37]	PLE [38]	BM3D [32]	EPLL [25]	NLM [3]	OneStep [18]	GSF (ours)	GLIDE [12]	Bayes [37]	PLE [38]	BM3D [32]	EPLL [25]
$\sigma$	<i>Baboon</i>								<i>Barbara</i>							
20	24.53	25.01	26.84	26.51	27.22	26.14	26.96	27.19	26.05	27.02	29.43	28.64	29.52	28.82	29.42	29.40
40	22.32	22.55	24.49	24.04	24.55	23.75	24.57	24.56	21.48	21.91	25.41	24.83	25.60	24.74	25.35	25.79
60	21.31	21.46	23.32	22.87	23.03	22.67	23.53	23.44	19.31	19.55	23.28	22.33	23.61	22.54	23.55	23.64
80	20.76	20.87	22.51	22.22	22.21	21.72	22.77	22.66	18.25	18.38	21.83	20.92	22.11	21.05	22.30	22.11
100	20.43	20.52	21.98	20.68	21.80	20.49	22.14	22.09	17.68	17.76	20.77	19.82	20.84	19.83	21.30	21.00
$\sigma$	<i>Boat</i>								<i>Bridge</i>							
20	24.88	26.05	28.43	27.55	28.59	27.53	28.58	28.76	23.99	24.95	26.90	26.34	27.26	26.83	27.09	27.25
40	21.97	22.39	24.96	24.35	25.04	24.26	25.12	25.32	20.87	21.35	23.85	23.18	24.02	23.29	23.88	24.19
60	20.46	20.70	23.19	22.59	23.60	22.21	23.47	23.56	19.58	19.85	22.24	21.47	22.45	21.37	22.44	22.48
80	19.60	19.74	22.14	21.42	22.21	21.19	22.43	22.41	18.83	19.01	21.20	20.44	21.25	20.35	21.45	21.42
100	19.09	19.18	21.34	20.50	21.43	19.81	21.74	21.53	18.35	18.48	20.46	19.75	20.16	19.79	20.67	20.66
$\sigma$	<i>Couple</i>								<i>Hill</i>							
20	24.54	25.62	28.20	27.25	28.33	27.43	28.42	28.60	25.51	26.38	28.68	27.98	28.99	28.12	28.82	28.97
40	21.67	22.10	24.64	23.95	24.96	23.86	25.00	25.11	22.58	23.11	25.55	24.79	25.61	24.92	25.70	25.85
60	20.35	20.60	23.07	22.32	23.11	22.40	23.36	23.37	21.33	21.69	23.95	23.26	23.91	23.25	24.21	24.16
80	19.64	19.81	22.02	21.40	21.76	20.76	22.32	22.30	20.68	20.93	22.90	22.42	22.60	21.78	23.19	23.12
100	19.24	19.35	21.23	19.80	20.97	18.15	21.56	21.52	20.29	20.49	22.07	21.85	21.75	20.83	22.37	22.39
$\sigma$	<i>House</i>								<i>Lena</i>							
20	28.20	30.02	32.92	31.82	32.54	31.57	32.73	32.47	26.90	28.03	29.83	29.19	30.13	28.91	29.93	30.06
40	23.26	24.27	28.31	27.31	28.49	26.75	28.91	28.77	22.40	23.11	26.40	25.96	26.40	25.46	26.23	26.70
60	21.40	21.79	26.05	24.72	26.10	24.11	26.68	26.58	20.22	20.60	24.49	23.51	24.57	22.73	24.49	24.80
80	20.52	20.70	24.46	22.96	23.88	22.41	25.20	25.04	19.09	19.32	23.10	22.00	22.61	21.54	23.22	23.45
100	20.04	20.13	23.21	20.80	22.84	21.07	23.96	23.83	18.47	18.62	22.03	20.98	21.05	20.56	22.25	22.41
$\sigma$	<i>Man</i>								<i>Pepper</i>							
20	25.14	26.09	28.12	27.37	28.37	27.56	28.13	28.43	26.17	27.89	29.58	28.86	29.61	28.80	29.61	29.76
40	21.93	22.26	24.78	24.29	25.05	24.53	24.91	25.19	21.19	22.23	25.43	24.61	25.68	24.25	25.44	26.02
60	20.26	20.45	23.12	22.24	23.20	22.75	23.26	23.48	19.05	19.61	23.28	22.24	23.43	21.24	23.35	23.77
80	19.33	19.46	22.01	20.72	22.09	21.30	22.26	22.27	17.92	18.22	21.71	20.58	21.52	20.79	21.93	22.16
100	18.78	18.87	21.07	20.42	20.94	20.70	21.48	21.38	17.27	17.45	20.51	19.54	20.60	19.05	20.86	20.93

TABLE V: PSNR comparison with different parameter  $h_s$ . The testing image is “Man”.  $\sigma = 40/255$ .

$h_s$	NLM (PSNR <sub>1</sub> )	One-Step (PSNR <sub>2</sub> )	Ours (PSNR <sub>3</sub> )	PSNR <sub>2</sub> - PSNR <sub>1</sub>	PSNR <sub>3</sub> - PSNR <sub>2</sub>
5	22.82	23.08	24.76	+0.26	+1.68
10	21.83	22.24	24.83	+0.41	+2.60
20	21.25	21.66	24.79	+0.41	+3.13
50	20.92	21.59	24.74	+0.68	+3.15
100	20.53	21.38	24.73	+0.85	+3.36

TABLE VII: PSNR comparison between GLIDE and GSF. Results are averaged over 10 testing images.

$\sigma$	GLIDE (PSNR <sub>1</sub> )	Ours (PSNR <sub>2</sub> )	PSNR <sub>2</sub> - PSNR <sub>1</sub>
20	28.15	28.89	+0.74
40	24.73	25.38	+0.65
60	22.75	23.60	+0.85
80	21.51	22.39	+0.88
100	20.42	21.47	+1.05

pilot estimate. The result in Table VII shows that in general GSF has at least 0.65dB improvement over GLIDE. This result is consistent with our observation that full Sinkhorn-Knopp is an incomplete EM algorithm.

C. Comparison with NL-Bayes, PLE and EPLL

Since the proposed GSF uses a Gaussian mixture model, we compare it with three other algorithms that also use Gaussian mixture models. These algorithms are the NL-Bayes [37], the piecewise linear estimator (PLE) [38] and the EPLL [25].

Methodologically, there are some important differences between GSF, NL-Bayes, PLE and EPLL. NL-Bayes has a grouping procedure that groups similar patches, a process similar to BM3D. The grouped patches are used to estimate the empirical mean and covariance of a single Gaussian. In GSF, there is no grouping. The other difference is that the denoising of NL-Bayes is performed by a conditional expectation over the *single* Gaussian. In GSF, the denoising is performed over all clusters of the GMM. Experimentally, we observe that GSF and NL-Bayes have similar performance, with NL-Bayes better in the low noise conditions and GSF better in the high noise conditions. One possible reason is that the grouping of NL-Bayes uses the standard Euclidean distance as a metric, which is not robust to noise.

PLE first estimates the model parameters using the noisy image. Then, for every patch, the algorithm selects a single Gaussian. The denoising is performed by solving a quadratic minimization and is performed for each patch individually. The algorithm iterates by improving the model parameters

and the denoised estimate until convergence. GSF does not have this iterative procedure. Once the GMM is learned, the denoising is performed in closed-form. The other difference is that PLE requires a good initialization and is very sensitive to the initialization. Experimentally, we find that GSF performs better than PLE using a MATLAB code provided the authors of PLE. In this MATLAB code, the initialization was originally designed for image inpainting at a particular image resolution. Because of the sensitivity of PLE to the initializations, its performance on denoising is not particularly good. With a better initialization, we believe that PLE would improve. However, even so the gap between GSF and PLE will unlikely be significant because PLE performs worse than BM3D and EPLL.

The closest comparison to GSF is EPLL as both algorithms solve a whole-image MAP minimization with a Gaussian mixture model. To evaluate the difference between the two algorithms, we consider an experiment by feeding the noisy patches the EM algorithm to learn a GMM. The patch size is fixed at  $5 \times 5$ , and the number of clusters is fixed as  $k = 100$ . We repeat the experiment by inputting the denoised result of BM3D and the oracle clean image into the EM algorithm.

From Table VIII, we observe that EPLL with a noisy input performs poorly. The reason is that the original EPLL trains the GMM from 2 million clean patches. When fed with noisy images, the GMM trained becomes a non-informative prior distribution. Moreover, in EPLL the GMM involves  $(\pi_i, \mu_i, \Sigma_i)$  whereas in GSF the GMM only involves  $(\pi_i, \mu_i)$ . This is a significant reduction in the number of model parameters. When fed with only a single image, there is insufficient training sample for EPLL.

Another observation from Table VIII is that the performance of EPLL depends heavily on the quality of the GMM. For example, if we use the result of BM3D as a pilot estimate for learning the GMM, the performance of EPLL is similar to the oracle case where we use the clean image. However, using BM3D as a pilot estimate is not a plausible approach because by running BM3D alone we can get an even higher PSNR (See Table VI). This result further shows the effectiveness of the proposed GSF for single image denoising.

TABLE VIII: Comparison with EPLL using different pilot estimates: “Noisy” uses the noisy image; “BM3D” uses the BM3D estimate; “Clean” uses the oracle clean image; “External” uses an external database. Testing image is “House”.

	EPLL	EPLL	EPLL	EPLL	Ours
$\sigma$	(Noisy)	(BM3D)	(Clean)	(External)	
20	25.40	32.41	32.46	32.47	32.92
40	19.75	28.32	28.31	28.77	28.31
60	16.42	25.73	25.80	26.58	26.05
80	14.29	24.05	24.07	25.04	24.46
100	12.71	22.59	22.73	23.83	23.21

#### D. Complexity and Limitations

Finally, we discuss the complexity and limitations of the proposed GSF.

TABLE IX: Number of clusters returned by cross-validation as noise level increases. Test image is “Man”. Size is  $128 \times 128$ .

$\sigma$	20	30	40	50	60	70	80	90	100
$k$	1445	667	372	243	162	125	104	83	72

TABLE X: Number of clusters returned by cross-validation as image size increases.  $\sigma = 40/255$ . Test image is “Man”.

		Ours	BM3D
Size	$k$	PSNR	PSNR
$50 \times 50$	120	22.76	22.36
$100 \times 100$	290	24.42	24.21
$150 \times 150$	501	25.21	25.32
$200 \times 200$	778	25.82	25.99
$250 \times 250$	996	26.14	26.35
$300 \times 300$	1322	26.58	26.83
$350 \times 350$	1646	26.97	27.20
$400 \times 400$	1966	27.26	27.49

GSF is a one-step denoising algorithm when  $\gamma_{ij}$  and  $\mu_i^{(r)}$  are known. However, learning the GMM using the EM algorithm is time-consuming, and the complexity depends on the number of clusters  $k$ . In addition, since  $k$  needs to be estimated through a cross-validation scheme, the actual complexity also depends on the number of cross-validation steps. To provide readers an idea of how  $k$  changes with other system parameters, we conduct two experiments.

In Table IX we show the number of clusters returned by the cross-validation scheme as we increase the noise level. As shown, the number of clusters increases when noise level reduces. This result is consistent with our intuition: As noise reduces, the grouping of patches becomes less important. In the limit when the image is noise-free, every patch will become its own cluster center. Therefore, one limitation of GSF is that for low-noise images the computing time could be very long. However, GSF is still a useful tool as its simple structure offers new insights to denoising.

Now, if we fix the noise level but change the image size, the complexity of GSF also varies. In Table X, we show the number of clusters as a function of image size. As a reference we also show the PSNR values of GSF and that of BM3D. The result in Table X indicates that the number of clusters increases with the image size. In the table, we also observe that BM3D performs worse than GSF for small images, but becomes better as image size increases.

*Remark 3 (Subtraction of mean):* It is perhaps interesting to ask whether it is possible to subtract the mean before learning the GMM, as it could reduce the number of clusters. However, from our experience, we find that this actually degrades the denoising performance. If the GMM is learned from a collection of zero-mean patches, the denoising step in  $(P_1)$  can only be used to denoise zero-mean patches. The mean values, which are also noisy, are never denoised. This phenomenon does not appear in EPLL (in which the GMM has a zero-mean) because the means are iteratively updated. We followed the same approach to iteratively update the means. However, we find that in general the denoising performance

is still worse than the original GMM with means included. Further exploration on this would likely provide more insights into the complexity reduction issue.

## VI. CONCLUSION

Motivated by the performance gain due to a column normalization step in defining the smoothing filters, we study the origin of the symmetrization process. Previous studies have shown that the symmetrization process is related to the Sinkhorn-Knopp balancing algorithm. In this paper, we further showed that the symmetrization is equivalent to an EM algorithm of learning a Gaussian mixture model (GMM). This observation allows us to generalize various symmetric smoothing filters including the Non-Local Means (NLM), the one-step Sinkhorn-Knopp and the full Sinkhorn-Knopp, and allows us to geometrically interpret the performance gain phenomenon.

Based on our findings, we proposed a new denoising algorithm called the Gaussian mixture model symmetric smoothing filters (GSF). GSF is a simple modification of the denoising framework by using the GMM prior for the maximum-a-posteriori estimation. Equipped with a cross-validation scheme which can automatically determine the number of clusters, GSF shows consistently better denoising results than NLM, One-step Sinkhorn-Knopp and full Sinkhorn-Knopp. While GSF has slightly worse performance than state-of-the-art methods such as BM3D, its simple structure highlights the importance of clustering in image denoising, which seems to be a plausible direction for future research.

## ACKNOWLEDGEMENT

We thank Dr. Guosheng Yu and Prof. Guillermo Sapiro for sharing the code of PLE [38]. We also thank the anonymous reviewers for the constructive feedback that helps to significantly improve the paper.

## APPENDIX

### A. Proof of Proposition 1

Given an estimator  $\hat{\mathbf{z}}$  of some observation  $\mathbf{y}$ , the SURE is defined as

$$\text{SURE} \stackrel{\text{def}}{=} -\sigma^2 + \frac{1}{n} \|\hat{\mathbf{z}} - \mathbf{y}\|^2 + \frac{2\sigma^2}{n} \text{div}(\hat{\mathbf{z}}). \quad (42)$$

Substituting (33) into (42) yields

$$\begin{aligned} \frac{1}{n} \|\hat{\mathbf{z}} - \mathbf{y}\|^2 &= \frac{1}{n} \left\| \frac{d}{d+\lambda} \mathbf{u} + \frac{\lambda}{d+\lambda} \mathbf{y} - \mathbf{y} \right\|^2 \\ &= \frac{1}{n} \left\| \frac{d}{d+\lambda} (\mathbf{u} - \mathbf{y}) \right\|^2 \\ &= \hat{\sigma}^2 \left( \frac{d}{d+\lambda} \right)^2, \end{aligned} \quad (43)$$

where  $\hat{\sigma}^2 \stackrel{\text{def}}{=} \frac{1}{n} \|\mathbf{u} - \mathbf{y}\|^2$ . So it remains to determine  $\text{div}(\hat{\mathbf{z}})$ .

From (33), the divergence  $\text{div}(\hat{\mathbf{z}})$  is

$$\begin{aligned} \text{div}(\hat{\mathbf{z}}) &= \frac{d}{d+\lambda} \text{div}(\mathbf{u}) + \frac{\lambda}{d+\lambda} \text{div}(\mathbf{y}) \\ &\stackrel{\text{def}}{=} \frac{d}{d+\lambda} \sum_{j=1}^n \frac{\partial u_j}{\partial y_j} + \frac{\lambda}{d+\lambda} \sum_{j=1}^n \frac{\partial y_j}{\partial y_j}. \end{aligned}$$

To determine  $\frac{\partial u_j}{\partial y_j}$ , we note from (34), (31) and (12) that

$$\mathbf{u} = \frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \left( \sum_{i=1}^k \gamma_{ij} \left( \frac{\sum_{j=1}^n \gamma_{ij} \mathbf{y}_j}{\sum_{j=1}^n \gamma_{ij}} \right) \right). \quad (44)$$

Since

$$\frac{\partial}{\partial y_j} \mathbf{y}_j = \frac{\partial}{\partial y_j} \begin{bmatrix} \vdots \\ y_{j-1} \\ y_j \\ y_{j+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} = \mathbf{e}_j,$$

it holds that

$$\text{div}(\mathbf{u}) = \mathbf{1}_{n \times 1}^T \left( \frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \left( \sum_{i=1}^k \gamma_{ij} \left( \frac{\sum_{j=1}^n \gamma_{ij} \mathbf{e}_j}{\sum_{j=1}^n \gamma_{ij}} \right) \right) \right),$$

and hence

$$\text{div}(\hat{\mathbf{z}}) = \sum_{j=1}^n \left( \frac{d}{d+\lambda} \text{div}(\mathbf{u}) + \frac{\lambda n}{d+\lambda} \right). \quad (45)$$

Substituting (45) and (43) into (42) completes the proof.

## REFERENCES

- [1] M. Wand and M. Jones, *Kernel Smoothing*, Chapman and Hall, London, 1995.
- [2] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24–52, Jan. 2009.
- [3] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *SIAM Multiscale Model and Simulation*, vol. 4, no. 2, pp. 490–530, 2005.
- [4] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, pp. 349–366, 2007.
- [5] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Processing Magazine*, vol. 30, pp. 106–128, Jan. 2013.
- [6] H. Talebi, X. Zhu, and P. Milanfar, "How to SAIF-ly boost denoising performance," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1470–1485, Apr. 2013.
- [7] F. Meyer and X. Shen, "Perturbation of the eigenvectors of the graph Laplacian: Application to image denoising," *Applied and Computational Harmonic Analysis*, 2013, In press. Available online at <http://arxiv.org/abs/1202.6666>.
- [8] G. Peyre, S. Bougleux, and L. Cohen, "Non-local regularization of inverse problems," in *Proc. European Conference on Computer Vision*, Oct. 2008, vol. 5304, pp. 57–68.
- [9] G. Peyre, "Manifold models for signals and images," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 249–260, Feb. 2009.
- [10] O. Lezoray and L. Grady, Eds., *Image Processing and Analysis with Graphs: Theory and Practice*, CRC Press, 2012.
- [11] S. H. Chan, T. Zickler, and Y. M. Lu, "Monte-Carlo non-local means: Random sampling for large-scale image filtering," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3711–3725, Aug. 2014.
- [12] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.
- [13] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005.

- [14] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," in *Proc. of ACM SIGGRAPH*, 2009, Article No. 21.
- [15] E. Gastal and M. Oliveira, "Adaptive manifolds for real-time high-dimensional filtering," *ACM Trans. Graphics*, vol. 31, no. 4, pp. 33:1–33:13, 2012.
- [16] H. Bhujle and S. Chaudhuri, "Novel speed-up strategies for non-local means denoising with patch and edge patch based dictionaries," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 356–365, Jan. 2014.
- [17] P. Milanfar, "Symmetrizing smoothing filters," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, 2013.
- [18] S. H. Chan, T. Zickler, and Y. M. Lu, "Fast non-local filtering by random sampling: it works, especially for large images," in *Proc. IEEE Intl. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2013, pp. 1603–1607.
- [19] S. H. Chan, T. Zickler, and Y. M. Lu, "Understanding symmetric smoothing filters via Gaussian mixtures," in *Proc. IEEE Intl. Conf. Image Process.*, Sep 2015, pp. 2500–2504.
- [20] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, pp. 876–879, 1964.
- [21] R. Sinkhorn and P. Knopp, "Concerning non-negative matrices and doubly-stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, pp. 343 – 348, 1967.
- [22] A. Buja, T. Hastie, and R. Tibshirani, "Linear smoothers and additive models," *Annals of Statistics*, vol. 17, pp. 453–510, 1989.
- [23] A. Cohen, "All admissible linear estimates of the mean vector," *Annals of Mathematical Statistics*, vol. 37, no. 2, pp. 458–463, Apr. 1966.
- [24] I. Ram, M. Elad, and I. Cohen, "Patch-ordering-based wavelet frame and its use in inverse problems," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 2779–2792, Jul. 2014.
- [25] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, pp. 479–486.
- [26] K. M. Taylor and F. G. Meyer, "A random walk on image patches," *SIAM Journal on Imaging Sciences*, vol. 5, pp. 688–725, 2012.
- [27] M. Gupta and Y. Chen, "Theory and use of the EM algorithm," *Foundations and Trends in Signal Processing*, vol. 4, no. 3, pp. 223–296, 2010.
- [28] D. R. Hunter and K. Lange, "A tutorial on MM algorithm," *The American Statistician*, vol. 58, no. 1, 2004.
- [29] R. Zhang, D. H. Ye, D. Pal, J.-B. Thibault, K. D. Sauer, and C. A. Bouman, "A Gaussian mixture MRF for model-based iterative reconstruction with applications to low-dose X-ray CT," *IEEE Trans. Computational Imaging*, vol. 2, no. 3, pp. 359–374, Sep. 2016.
- [30] C. Kervrann and J. Boulanger, "Local adaptivity to variable smoothness for exemplar-based image regularization and representation," *International Journal on Computer Vision*, vol. 79, no. 1, pp. 45–69, 2008.
- [31] E. Luo, S. H. Chan, S. Pan, and T. Q. Nguyen, "Adaptive non-local means for multiview image denoising: Searching for the right patches via a statistical approach," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 543–547.
- [32] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
- [33] C. Stein, "Estimation of the mean of a multivariation normal distribution," *Annals of Statistics*, vol. 9, pp. 1135–1151, 1981.
- [34] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1540–1554, 2008.
- [35] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *European Conference on Computer Vision*, 2014, pp. 783–798.
- [36] L. Wasserman, *All of Nonparametric Statistics*, Springer, 2005.
- [37] M. Lebrun, A. Buades, and J. M. Morel, "A nonlocal Bayesian image denoising algorithm," *SIAM J. Imaging Sciences*, vol. 6, no. 3, pp. 1665–1688, Sep. 2013.
- [38] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2481–2499, May 2012.