UNDERSTANDING SYMMETRIC SMOOTHING FILTERS VIA GAUSSIAN MIXTURES

Stanley H. Chan[1]*, Todd Zickler*[2]*, and Yue M. Lu*[2]

[1]School of ECE and Dept of Statistics, Purdue University, West Lafayette, IN 47907.
[2]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138.

ABSTRACT

We study a class of smoothing filters for image denoising. Expressed as matrices, these smoothing filters must be row normalized so that each row sums to unity. Surprisingly, if one applies a column normalization to the matrix before the row normalization, the denoising quality can often be significantly improved. This column-row normalization corresponds to one iteration of a symmetrization process called the Sinkhorn-Knopp balancing algorithm. However, a complete understanding of the performance gain phenomenon is lacking.

In this paper, we analyze the performance gain from a Gaussian mixture model (GMM) perspective. We show that the symmetrization is equivalent to an expectation-maximization (EM) algorithm for learning the GMM. Moreover, we make modifications to the symmetrization procedure and present a new denoising algorithm. Experimental results show that the new algorithm achieves comparable denoising results to some state-of-the-art methods.

Index Terms— Non-local means, patch-based filtering, Gaussian mixture model, Expectation-Maximization, doubly-stochastic matrix, symmetric smoothing filter

1. INTRODUCTION

1.1. Motivation: A Surprising Observation

Consider a noisy observation $\boldsymbol{y} \in \mathbb{R}^n$ of a clean image $\boldsymbol{z} \in \mathbb{R}^n$ corrupted by additive i.i.d. Gaussian noise. We like to denoise $\boldsymbol{y}$ using a non-local weighted average method with the form

$$\widehat{\boldsymbol{z}} = \boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{y}, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a matrix denoting the weights, and $\boldsymbol{D} \stackrel{\text{def}}{=} \text{diag}\{\boldsymbol{W}\boldsymbol{1}\}$ is a diagonal matrix for normalization so that each row of $\boldsymbol{D}^{-1}\boldsymbol{W}$ sums to unity. We refer to $\boldsymbol{W}$ as the *smoothing filter*.

In general, the matrix $\boldsymbol{D}^{-1}\boldsymbol{W}$ described by (1) is equivalent to a directed graph [1–3]. This notion of graph operation allows one to link the smoothing filter to a number of existing denoising methods, e.g., Gaussian filter [4], bilateral filter [5], non-local means (NLM) [6], locally adaptive regression kernel (LARK) [7], etc. Interested readers can check [8] for a comprehensive discussion.

Now, consider a simple modification of (1) as follows:

$$\widehat{\boldsymbol{z}} = \boldsymbol{D}_r^{-1}\boldsymbol{W}\boldsymbol{D}_c^{-1}\boldsymbol{y}, \tag{2}$$

where $\boldsymbol{D}_c \stackrel{\text{def}}{=} \text{diag}\left\{\boldsymbol{W}^T\boldsymbol{1}\right\}$ is a diagonal matrix that normalizes the *columns* of $\boldsymbol{W}$, and $\boldsymbol{D}_r \stackrel{\text{def}}{=} \text{diag}\left\{\boldsymbol{W}\boldsymbol{D}_c^{-1}\boldsymbol{1}\right\}$ is a diagonal matrix that normalizes the *rows* of $\boldsymbol{W}\boldsymbol{D}_c^{-1}$. In other words, we modify (1) by introducing a column normalization before applying the row normalization. Surprisingly, the column-row normalization in (2), which is a simple modification of (1), improves the denoising quality by some significant margin. Figure 1 shows some examples.

Contact author: S. Chan, e-mail: stanleychan@purdue.edu.

1.2. Sinkhorn-Knopp

To the best of our knowledge, the above performance gain phenomenon was first observed and discussed by Milanfar in [9], where it was shown that if we repeatedly apply the column-row normalization we would obtain an iterative algorithm called the Sinkhorn-Knopp balancing algorithm (See Algorithm 1.) The matrix computed by Sinkhorn-Knopp is a doubly-stochastic matrix — a symmetric non-negative matrix with unit columns and rows (also called a *symmetric* smoothing filter). Symmetric smoothing filters are admissible [10] and are stable in the sense that all their eigenvalues are bounded in the unit interval [11].

Algorithm 1 Sinkhorn-Knopp Balancing Algorithm

Input: $\boldsymbol{W}^{(0)}$
while $\|\boldsymbol{W}^{(m+1)} - \boldsymbol{W}^{(m)}\|_F > \texttt{tol}$ **do**
  $\boldsymbol{D}_c = \text{diag}\left\{(\boldsymbol{W}^{(m)})^T\boldsymbol{1}\right\}$ % Column Normalize
  $\boldsymbol{D}_r = \text{diag}\left\{\boldsymbol{W}^{(m)}\boldsymbol{D}_c^{-1}\boldsymbol{1}\right\}$ % Row Normalize
  $\boldsymbol{W}^{(m+1)} = \boldsymbol{D}_r^{-1}\boldsymbol{W}^{(m)}\boldsymbol{D}_c^{-1}$
end while

In order to explain the performance gain, Milanfar [9] showed that symmetric smoothing filters often have higher "effective degrees of freedom", and so the reduction in the bias of $\widehat{\boldsymbol{z}}$ is greater than the gain in variance of $\widehat{\boldsymbol{z}}$. As a result, the overall mean squared error, which is the sum of the bias and the variance, is reduced. However, what is still unclear is a quantitative interpretation of this "effective degrees of freedom". In particular, it would be useful if we can understand what geometrical actions are being applied to the noisy image by the column-row normalization.

1.3. Contributions

The goal of this paper is to provide an explanation to the performance gain phenomenon. Our approach is to formulate the problem using a Gaussian mixture model (GMM). The two main contributions of the paper are summarized as follows.

First, we generalize the symmetrization by reformulating the denoising problem as seeking the cluster centers of a GMM. We show that the original smoothing filter in (1), the One-step Sinkhorn-Knopp in (2), and the full Sinkhorn-Knopp in Algorithm 1 are all sub-routines of the Expectation-Maximization (EM) algorithm [12] for learning the GMM. By showing that each of methods supersedes its previous counterpart, we explain the performance gain phenomenon. (See Section 2.)

Image No.	1	2	3	4	5	6	7	8	9	10
$\boldsymbol{D}^{-1}\boldsymbol{W}$	34.12	31.60	32.59	25.64	26.36	25.63	22.31	24.98	26.25	22.47
$\boldsymbol{D}_r^{-1}\boldsymbol{W}\boldsymbol{D}_c^{-1}$	34.56	33.49	36.44	26.50	29.37	27.71	22.92	26.49	27.85	23.52
PSNR Improvement	+0.44	+1.89	+3.85	+0.86	+3.01	+2.08	+0.61	+1.51	+1.60	+1.05

Fig. 1: [Top] 100×100 testing images. Each image is corrupted by i.i.d Gaussian noise of $\sigma = 20/255$. [Bottom] PSNR values of the denoised image using $\boldsymbol{D}^{-1}\boldsymbol{W}$ and $\boldsymbol{D}_r^{-1}\boldsymbol{W}\boldsymbol{D}_c^{-1}$.

Second, based on the GMM framework, we propose in Section 3 a new denoising algorithm called Gaussian Mixture Smoothing Filters (GMSF). We demonstrate in Section 4 that GMSF does not only subsume a number of smoothing filters, but also has a performance similar to several state-of-the-art denoising algorithms.

2. GENERALIZE SYMMETRIC SMOOTHING FILTERS

2.1. Notations

Throughout this paper, we use n to denote the number of pixels, and k to denote the number of clusters of the GMM. We use the vector $\boldsymbol{x}_j \in \mathbb{R}^2$ to represent the two-dimensional spatial coordinate of the jth pixel, the vector $\boldsymbol{y}_j \in \mathbb{R}^d$ to represent a d-dimensional patch centered at the jth pixel of the noisy image, and the scalar $y_j \in \mathbb{R}$ to represent the value of the center pixel of jth patch. Without loss of generality, we assume that all pixel intensity values have been normalized to the range $[0,\ 1]$. Finally, we focus on smoothing filters of the following form:

$$W_{ij} = \exp\left\{-\frac{1}{2}(\boldsymbol{p}_i - \boldsymbol{p}_j)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{p}_i - \boldsymbol{p}_j)\right\}, \tag{3}$$

where $\boldsymbol{p}_i \in \mathbb{R}^p$ denotes a p-dimensional feature vector of the ith pixel of the noisy image, and $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ is a positive definite matrix. For example, in NLM [6], the feature vector and the matrix are

$$\boldsymbol{p}_i = \begin{bmatrix}\boldsymbol{x}_i \\ \boldsymbol{y}_i\end{bmatrix}, \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{bmatrix}h_s^2\boldsymbol{I} & 0 \\ 0 & h_r^2\boldsymbol{I}\end{bmatrix} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\text{NLM}}, \tag{4}$$

respectively, where h_s and h_r are parameters.

2.2. EM Algorithm for GMM

In this paper we are interested in GMM with a known covariance $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{NLM}}$. Given a set of n data points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbb{R}^p$, we say that $\{\boldsymbol{p}_j\}_{j=1}^n$ is generated from a GMM of k clusters if the distribution of $\boldsymbol{p}_j$ is a linear combination of k Gaussian distributions with means $\{\boldsymbol{\mu}_i\}_{i=1}^k \in \mathbb{R}^p$. Mathematically, the distribution is

$$f(\boldsymbol{p}_j \,|\, \pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{\text{NLM}}) = \sum_{i=1}^k \pi_i\, \mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{\text{NLM}}). \tag{5}$$

where π_i is the weight of the ith Gaussian component, and $\mathcal{N}(\boldsymbol{p} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

Learning the GMM parameter $(\pi_i,\ \boldsymbol{\mu}_i)$ can be done using the Expectation-Maximization (EM) algorithm. EM algorithm is a well-established technique so we shall not repeat here. Interested readers can refer to [12] for detailed derivations. A summary of the EM algorithm is shown in Algorithm 2.

Algorithm 2 EM Algorithm for Learning GMM [12], with a known covariance $\boldsymbol{\Sigma}$.

Input: Data $\{\boldsymbol{p}_j\}_{j=1}^n$, number of clusters k, covariance $\boldsymbol{\Sigma}$.
Output: Parameters $\{(\pi_i, \boldsymbol{\mu}_i)\}_{i=1}^k$.
while not converge **do**
E-step: Compute, for $i = 1, \ldots, k$ and $j = 1, \ldots, n$

$$\gamma_{ij}^{(m)} = \frac{\pi_i^{(m)}\mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{\mu}_i^{(m)}, \boldsymbol{\Sigma})}{\sum_{l=1}^k \pi_l^{(m)}\mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{\mu}_l^{(m)}, \boldsymbol{\Sigma})} \tag{6}$$

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

$$\pi_i^{(m+1)} = \frac{1}{n}\sum_{j=1}^n \gamma_{ij}^{(m)} \text{ and } \boldsymbol{\mu}_i^{(m+1)} = \frac{\sum_{j=1}^n \gamma_{ij}^{(m)}\boldsymbol{p}_j}{\sum_{j=1}^n \gamma_{ij}^{(m)}} \tag{7}$$

Update Counter: $m \leftarrow m + 1$.
end while

The GMM returned by the EM algorithm is a clustering structure. In order to perform the actual denoising, it is important to specify a denoising procedure. To this end, we consider the following optimization, of which the ith denoised pixel is:

$$\widehat{z}_i = \underset{z_i}{\text{argmin}} \ \sum_{j=1}^n \gamma_{ij}(z_i - y_j)^2 = \frac{\sum_{j=1}^n \gamma_{ij} y_j}{\sum_{j=1}^n \gamma_{ij}}. \tag{8}$$

Clearly, (8) coincides with (1) if we set $\gamma_{ij} = W_{ij}$ using (3) and (4).

2.3. Generalization

We now discuss the generalization of the original filter in (1), the One-step Sinkhorn-Knopp in (2), and the full Sinkhorn-Knopp in Algorithm 1. A summary of our results is shown in Table 1.

A. Original Filter: We first study the original filter in (1). To establish its relationship with GMM, we initialize the EM algorithm with $\pi_i^{(0)} = 1/k$, $\boldsymbol{\mu}_i^{(0)} = \boldsymbol{p}_i$, and initialize $\gamma_{ij}^{(0)}$ as

$$\gamma_{ij}^{(0)} = \pi_i^{(0)}\mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{\mu}_i^{(0)}, \boldsymbol{\Sigma}_{\text{NLM}}) = \frac{1}{k}\mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{p}_i, \boldsymbol{\Sigma}_{\text{NLM}}). \tag{9}$$

Table 1: Generalization and comparisons using EM algorithm for learning GMM with known $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{NLM}}$.

		Original NLM [6]	One-step Sinkhorn-Knopp [13]	Full Sinkhorn-Knopp [14]	GMSF (generalization)
No. Clusters		$k = n$	$k = n$	$k = n$	$k < n$ (cross-validation)
Initialization	$\pi_i^{(0)}$	$1/k$	$1/k$	$1/k$	$1/k$
	$\boldsymbol{\mu}_i^{(0)}$	$\boldsymbol{p}_i$	$\boldsymbol{p}_i$	$\boldsymbol{p}_i$	randomly picked $\boldsymbol{p}_i$
	$\gamma_{ij}^{(0)}$	$\pi_i^{(0)}\mathcal{N}(\boldsymbol{p}_j \vert \boldsymbol{\mu}_i^{(0)}, \boldsymbol{\Sigma}_{\text{NLM}})$	$\times$	$\times$	$\times$
E-step	Update $\gamma_{ij}^{(m)}$	$\times$	✓	✓ (via $\beta_{ij}^{(m)}$)	✓
M-step	Update $\pi_i^{(m)}$	$\times$	$\times$	$\times$	✓
	Update $\boldsymbol{\mu}_i^{(m)}$	$\times$	$\times$	✓ (implicitly)	✓
No. Iterations		0	1	Many	Many
Denoising	$\widehat{z}_i$	$\frac{\sum_{j=1}^n \gamma_{ij}^{(0)} y_j}{\sum_{j=1}^n \gamma_{ij}^{(0)}}$	$\frac{\sum_{j=1}^n \gamma_{ij}^{(0)} y_j}{\sum_{j=1}^n \gamma_{ij}^{(0)}}$	$\frac{\sum_{j=1}^n \gamma_{ij}^{(m+1)} y_j}{\sum_{j=1}^n \gamma_{ij}^{(m+1)}}$	Optimization Solution

Substituting (9) into (8), we observe that the denoised image of this (zero-iteration) EM algorithm is

$$\widehat{z}_i = \underset{z_i}{\text{argmin}} \sum_{j=1}^{n} \gamma_{ij}^{(0)} (z_i - y_j)^2 = \frac{\sum_{j=1}^n W_{ij} y_j}{\sum_{j=1}^n W_{ij}}, \tag{10}$$

which is exactly (1).

B. One-step Sinkhorn-Knopp: The column normalization of the One-Step Sinkhorn-Knopp is

$$[\boldsymbol{W}\boldsymbol{D}_c^{-1}]_{ij} = \frac{W_{ij}}{\sum_{i=1}^k W_{ij}} = \frac{\mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}{\sum_{i=1}^k \mathcal{N}(\boldsymbol{p}_j \,|\, \boldsymbol{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}, \tag{11}$$

which is equivalent to the E-step in (6) with the initializations $\pi_i^{(0)} = 1/k$ and $\boldsymbol{\mu}_i^{(0)} = \boldsymbol{p}_i$. By defining $\gamma_{ij}^{(0)} = [\boldsymbol{W}\boldsymbol{D}_c^{-1}]_{ij}$, the row normalization becomes

$$[\boldsymbol{D}_r^{-1}\boldsymbol{W}\boldsymbol{D}_c^{-1}]_{ij} = \frac{[\boldsymbol{W}\boldsymbol{D}_c^{-1}]_{ij}}{\sum_{j=1}^n [\boldsymbol{W}\boldsymbol{D}_c^{-1}]_{ij}} = \frac{\gamma_{ij}^{(0)}}{\sum_{j=1}^n \gamma_{ij}^{(0)}}, \tag{12}$$

which is equivalent to updating the mean in the M-step of (7).

By comparing (9) and (11), we note that the right hand side of (11) is a *legitimate* probability whereas (9) is not. The probability specifies the likelihood that $\boldsymbol{p}_j$ belongs to the ith cluster with cluster center $\boldsymbol{\mu}_i$. In other words, the underlying mechanism of the normalization is a clustering process. Since the One-step Sinkhorn-Knopp corresponds to a legitimate probability, the corresponding clustering action is better defined than that of the original filter.

C. Full Sinkhorn-Knopp: The full Sinkhorn-Knopp is equivalent to iteratively apply the steps

$$\textbf{E-step}: \quad \gamma_{ij}^{(m)} = \frac{\beta_{ij}^{(m)}}{\sum_{i=1}^k \beta_{ij}^{(m)}}, \tag{13a}$$

$$\textbf{M-step}: \quad \beta_{ij}^{(m+1)} = \frac{\gamma_{ij}^{(m)}}{\sum_{j=1}^n \gamma_{ij}^{(m)}}, \tag{13b}$$

with initializations $\boldsymbol{\mu}_i^{(0)} = \boldsymbol{p}_i$ and $\beta_{ij}^{(0)} = \pi_i^{(0)}\mathcal{N}(\boldsymbol{p}_j \vert \boldsymbol{\mu}_i^{(0)}, \boldsymbol{\Sigma}_{\text{NLM}})$, and a *fixed* mixture weight $\pi_i^{(m)} = 1/k$ for *all* iterations.

While the full Sinkhorn-Knopp is also a clustering process, the EM steps defined by (13) do not correspond to a complete EM algorithm. In particular, the mixture weights $\pi_i^{(m)}$ are never updated. This means that Sinkhorn-Knopp forces all mixtures to be equally likely. Moreover, the M-step is only implicitly updated via an intermediate parameter $\beta_{ij}^{(m)}$. If we compare (6) with (13a), we see that $\beta_{ij}^{(m)}$ is an approximation of the actual probability $\pi_i^{(m)}\mathcal{N}(\boldsymbol{p}_j \vert \boldsymbol{\mu}_i^{(m)}, \boldsymbol{\Sigma}_{\text{NLM}})$. The cluster center $\boldsymbol{\mu}_i^{(m+1)}$, which is never explicitly calculated in (13), is

$$\boldsymbol{\mu}_i^{(m+1)} = \frac{\sum_{j=1}^n \gamma_{ij}^{(m)} \boldsymbol{p}_j}{\sum_{j=1}^n \gamma_{ij}^{(m)}} \overset{(a)}{=} \sum_{j=1}^{n} \beta_{ij}^{(m+1)} \boldsymbol{p}_j \tag{14}$$

and it is not used to update $\gamma_{ij}^{(m+1)}$, where (a) follows from (13b). Therefore, the EM steps are not completely utilized and so the effectiveness of the algorithm is limited.

Since the full Sinkhorn-Knopp is not a complete EM algorithm, one cannot guarantee a performance gain. In Figure 2 we show the PSNR values of running the full Sinkhorn-Knopp for the images shown in Figure 1. The result shows that having more Sinkhorn-Knopp iterations does not always improve the PSNR.

3. GAUSSIAN MIXTURE SMOOTHING FILTER

Based on the above observations, it is natural to ask if we can improve the denoising results by utilizing the full EM algorithm. In this section, we present a new denoising algorithm called the Gaussian Mixture Smoothing Filter (GMSF).

GMSF applies the complete EM algorithm to learn a GMM with a known covariance. Once the GMM parameters $(\pi_i, \boldsymbol{\mu}_i)$ are learned, we solve the following optimization:

$$\widehat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\text{argmin}} \ \lambda \|\boldsymbol{z} - \boldsymbol{y}\|^2 + \sum_{j=1}^{n} \sum_{i=1}^{k} \gamma_{ij} \left\| \boldsymbol{P}_j \boldsymbol{z} - \boldsymbol{\mu}_i^{(r)} \right\|^2, \tag{15}$$

where λ is a parameter, $\boldsymbol{P}_j \in \mathbb{R}^{d \times n}$ is a patch-extract operator that extracts the jth patch of the image, and $\boldsymbol{\mu}_i^{(r)}$ is the intensity component of the GMM cluster center $\boldsymbol{\mu}_i \overset{\text{def}}{=} [\boldsymbol{\mu}_i^{(s)}; \boldsymbol{\mu}_i^{(r)}]$. (See definition of $\boldsymbol{p}_i$ in (4).) If we let $\lambda = 0$, $\boldsymbol{\mu}_i^{(r)} = y_i$, and $\boldsymbol{P}_j$ be an operator that extracts the jth pixel of $\boldsymbol{z}$, then we can show that (8) is a special case of (15).

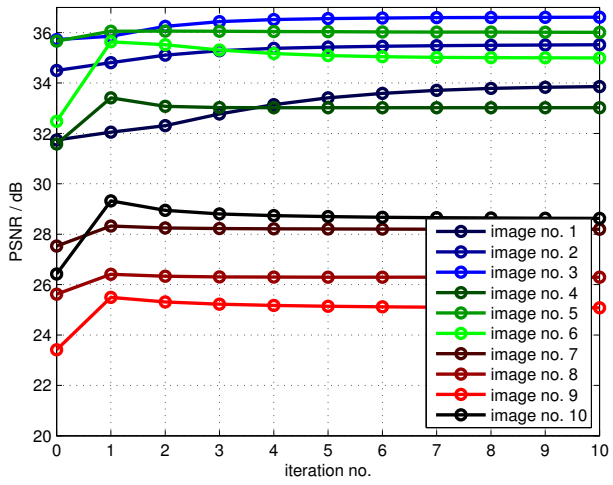

Fig. 2: Extension of the experiment in Figure 1. The PSNR values does not always increase as more Sinkhorn-Knopp iterations are used. The curves are averaged over 16 independent trials of different noise realizations.

The optimization in (15) is closely related to the expected patch log-likelihood (EPLL) framework [15]. Here, the first term $\|\boldsymbol{z}-\boldsymbol{y}\|^2$ is the data fidelity term that captures the deviation between the estimate $\boldsymbol{z}$ and the observed data $\boldsymbol{y}$. The second term $\sum_{j=1}^{n}\sum_{i=1}^{k}\gamma_{ij}\|\boldsymbol{P}_j\boldsymbol{z}-\boldsymbol{\mu}_i^{(r)}\|^2$ is the prior distribution of the jth patch $\boldsymbol{P}_j\boldsymbol{z}$. In EPLL, the prior distribution of $\boldsymbol{P}_j\boldsymbol{z}$ is a Gaussian mixture model learned from an *external* database, whereas the prior distribution in (15) is

$$f(\boldsymbol{P}_j\boldsymbol{z}) \propto \exp\left\{-\sum_{i=1}^{k}\gamma_{ij}\|\boldsymbol{P}_j\boldsymbol{z}-\boldsymbol{\mu}_i^{(r)}\|^2\right\}, \qquad (16)$$

where the parameters γ_{ij} and $\boldsymbol{\mu}_i^{(r)}$ are learned from the *noisy image*. Intuitively, the prior in (16) measures the weighted distance between $\boldsymbol{P}_j\boldsymbol{z}$ and *all* cluster centers $\{\boldsymbol{\mu}_i^{(r)}\}_{i=1}^k$. Therefore, $f(\boldsymbol{P}_j\boldsymbol{z})$ is small if $\boldsymbol{P}_j\boldsymbol{z}$ is different from most of the cluster centers, i.e., $\boldsymbol{P}_j\boldsymbol{z}$ is a rare patch.

The optimization in (15) is a quadratic problem, and so closed-form solution exists. In particular, by considering the first order optimality condition we obtain a normal equation

$$\left(\sum_{j=1}^{n}\boldsymbol{P}_j^T\boldsymbol{P}_j+\lambda\boldsymbol{I}\right)\widehat{\boldsymbol{z}}=\sum_{j=1}^{n}\boldsymbol{P}_j^T\boldsymbol{w}_j+\lambda\boldsymbol{y}, \qquad (17)$$

of which the solution is the minimizer of (15). Here in (17), the vector $\boldsymbol{w}_j$ is defined as

$$\boldsymbol{w}_j \overset{\text{def}}{=} \sum_{i=1}^{n}\gamma_{ij}\boldsymbol{\mu}_i^{(r)}. \qquad (18)$$

As far as parameters are concerned, we note that (15) involves two parameters: the regularization parameter λ and the number of clusters k. In this paper, we use Stein's Unbiased Risk Estimator (SURE) [16, 17] to optimally determine λ, and a cross-validation scheme to determine k. Due to space limit, we leave the details of the parameter selection process to a follow up article.

4. EXPERIMENT

We compare GMSF with several denoising algorithms, including the original NLM [6], One-step [13], BM3D [18], EPLL [15], Global image denoising (GLIDE) [19], and SURE-let [20]. We note the work of Luszczkiewicz-Piatek [21]. However, [21] is fundamentally different from GMSF as it only uses GMM to model the 2D histogram of the chroma-channels of a color image.

For NLM and One-step, we implement the algorithm by setting the patch size as 5×5 (i.e., $d=25$). The parameters are $h_s=10$ and $h_r=\sigma\sqrt{d}$. For BM3D, EPLL, GLIDE and SURE-let, we downloaded the original MATLAB code from the authors' websites.

We test the algorithms on 10 images previously used in [22]. The results are shown in Table 2, where we show the average PSNR over 10 testing images. Comparing NLM and One-step Sinkhorn-Knopp, we observe that GMSF yields a significantly higher PSNR, indicating the effectiveness of the complete EM steps. GLIDE has a full Sinkhorn-Knopp iteration in its algorithm. However, experimental results show that GMSF achieves a higher PSNR. EPLL has a better performance than GMSF. However, Table 3 shows that the superiority of EPLL came from the external database. If we use the noisy image for learning the GMM, GMSF achieves a better result.

Table 2: Average PSNR over 10 testing images of size 128×128.

σ	NLM [6]	1-step [13]	Ours	GLIDE [19]	BM3D [18]	EPLL [15]	SURE [20]
20	25.59	26.71	28.89	28.15	28.97	29.09	27.54
40	21.97	22.53	25.38	24.73	25.51	25.75	24.23
60	20.33	20.63	23.60	22.75	23.83	23.93	22.46
80	19.46	19.64	22.39	21.51	22.71	22.70	21.23
100	18.97	19.09	21.47	20.42	21.83	21.77	20.25

Table 3: EPLL with different datasets for learning the GMM: "Noisy" – noisy image; "BM3D" – BM3D estimate; "Orale" – clean image; "External" – external database. Testing image is "House".

σ	EPLL (Noisy)	EPLL (BM3D)	EPLL (Oracle)	EPLL (External)	Ours
20	25.40	32.41	32.46	32.47	32.92
40	19.75	28.32	28.31	28.77	28.31

5. CONCLUSION

We studied the performance gain phenomenon of a symmetric smoothing filter from a Gaussian mixture model (GMM) perspective. We show that the original NLM, the One-step Sinkhorn-Knopp and the full Sinkhorn-Knopp are special cases of an EM algorithm for learning a GMM.

We generalized our observations and proposed a new denoising algorithm called the Gaussian Mixture Smoothing Filters (GMSF). GMSF is a simple modification of the NLM optimization framework by applying the complete EM algorithm. Our preliminary results indicate that GMSF has a consistently better denoising result than NLM, One-step Sinkhorn-Knopp and full Sinkhorn-Knopp. While GMSF 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 should be explored for future research.

6. REFERENCES

[1] 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.

[2] 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.

[3] K. M. Taylor and F. G. Meyer, "A random walk on image patches," *SIAM Journal on Imaging Sciences*, vol. 5, pp. 688–725, 2012.

[4] M. Wand and M. Jones, *Kernel Smoothing*, Chapman and Hall, London, 1995.

[5] 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.

[6] 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.

[7] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image. Process.*, vol. 16, pp. 349–366, 2007.

[8] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Processing Magazine*, vol. 30, pp. 106–128, Jan. 2013.

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

[10] A. Cohen, "All admissible linear estimates of the mean vector," *Annals of Mathematical Statistics*, vol. 37, no. 2, pp. 458–463, Apr. 1966.

[11] A. Buja, T. Hastie, and R. Tibshirani, "Linear smoothers and additive models," *Annals of Statistics*, vol. 17, pp. 453–510, 1989.

[12] 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.

[13] 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.

[14] R. Sinkhorn and P. Knopp, "Concerning non-negative matrices and doubly-stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, pp. 343 – 348, 1967.

[15] 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.

[16] C. Stein, "Estimation of the mean of a multivariation normal distribution," *Annals of Statistics*, vol. 9, pp. 1135–1151, 1981.

[17] 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.

[18] 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.

[19] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.

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

[21] M. Luszczkiewicz-Piatek, *Gaussian Mixture Model Based Non-Local Means Technique for Mixed Noise Suppression in Color Images*, pp. 75–83, Advances in Intelligent Systems and Computing. Springer International Publishing, 2015.

[22] 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.