

Demystifying Symmetric Smoothing Filters

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 Product of Gaussians (PoG) prior 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. The new PoG model also allows us to develop a new denoising algorithm called Product of Gaussian Non-Local-Means (PoG-NLM). PoG-NLM is an extension of the Sinkhorn-Knopp and is a generalization of the classical non-local means. Despite its simple formulation, PoG-NLM outperforms many existing smoothing filters and has a similar performance compared to BM3D.

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

I. INTRODUCTION

A. Motivation: A Surprising Phenomenon

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. We would like to denoise \mathbf{y} using a linear operator $\mathbf{W} \in \mathbb{R}^{n \times n}$ and estimate the denoised image $\hat{\mathbf{z}}$ 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 linear operator \mathbf{W} is referred to as the *smoothing filter*.

The smoothing filter described by (1) is a general expression for many popular denoising algorithms, e.g., Gaussian filter [1], bilateral filter [2], non-local means (NLM) [3], locally

adaptive regression kernel (LARK) [4], etc. Recently, it has been shown that these smoothing filters are closely related to the operations of graph Laplacians formed from image patches [5]–[7]. Moreover, fast computations of smoothing filters have been proposed [8]–[13].

While smoothing filters work well for many denoising problems, it has been observed in [14], [15] and [16] that their performance can be 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 before applying the row normalization.

Before we proceed to discuss the technical properties of (1) and (2), we first provide some numerical simulations to demonstrate an interesting phenomenon. In Figure 1, we show 10 clean images (each of size 100×100) consisting of various content and textures. We generate the noisy images by adding i.i.d. Gaussian noise of standard deviation $\sigma = 20/255$. Then, we denoise the noisy images using the non-local mean [3] for (1) and (2), respectively.

The results of this experiment are shown in 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 1.6 dB on average. Another puzzling observation is that if we repeatedly apply the column-row normalization, the PSNR can drop after a certain number of iterations. Figure 2 presents this result. For 6 out of the 10 images we have 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? If we can answer this question, then perhaps we can develop a 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 [14], 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 [17], [18] (or Sinkhorn-Knopp, in short). As illustrated in Algorithm 1, Sinkhorn-Knopp is a simple algorithm that repeatedly applies the column and row normalizations until the smoothing filter converges. For example, the smoothing filter defined by (2) is the result of applying Sinkhorn-Knopp for one iteration.

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 while 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.

This paper follows the concept of reproducible research. All the results and examples presented in the paper are reproducible using the code and images available online at <http://engineering.purdue.edu/ChanGroup/>.



Smoothing Filter	Image No.									
	1	2	3	4	5	6	7	8	9	10
$D^{-1}\mathbf{W}$	34.12	31.60	32.59	25.64	26.36	25.63	22.31	24.98	26.25	22.47
$D_r^{-1}\mathbf{W}D_c^{-1}$	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 $D^{-1}\mathbf{W}$ and $D_r^{-1}\mathbf{W}D_c^{-1}$.

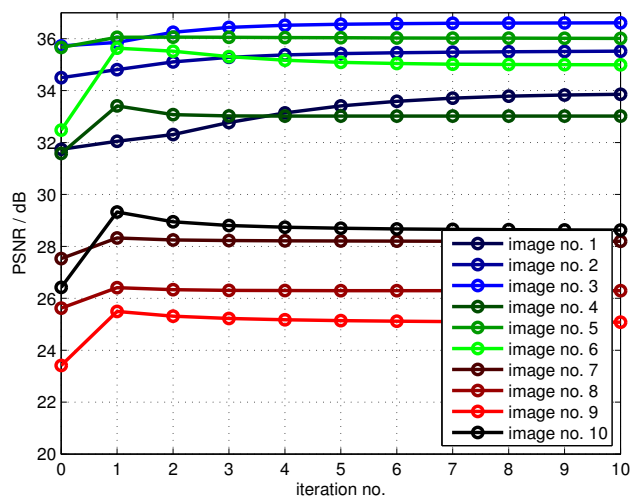


Fig. 2: Extension of the experiment in Figure 1. The PSNR values does not always increase as more Sinkhorn-Knopp iterations are used. The curves are averaged over 16 independent trials of different noise realizations.

Algorithm 1 Sinkhorn-Knopp Balancing Algorithm

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

Sinkhorn-Knopp has many interesting properties. First, when Sinkhorn-Knopp converges, the converging limit 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 [19] and are stable in the sense that all eigenvalues are bounded in the unit interval [20]. Also, symmetric smoothing filters can be used to form a graph Laplacian in some recently proposed denoising algorithms, e.g., [9].

To explain the performance gain, Milanfar showed in [14] that the symmetric smoothing filters often have higher “effec-

tive degrees of freedom”, and so the reduction in the bias of \hat{z} is greater than the gain in the variance of \hat{z} . As a result, the overall mean squared error, which is the sum of the bias and the variance, is reduced. However, this notion of “effective degrees of freedom” is not easy to interpret intuitively. In particular, it would be significantly more useful if we can geometrically describe the actions under which the column-row normalization are applying to the noisy image.

C. Contributions

The goal of this paper is to provide an explanation to the performance gain phenomenon and to propose a new denoising algorithm. Our approach is to study an Expectation-Maximization (EM) algorithm for learning a special prior which will be discussed shortly. By analyzing the E-step and the M-step of the EM algorithm, we found 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 for superior image denoising [21]–[23]. The two major contributions of the paper are summarized as follows.

First, we generalize the symmetrization process by reformulating the denoising problem as a maximum-a-posteriori (MAP) estimation with a new patch prior in the form of a product of Gaussians (PoG). We show that the original smoothing filter in (1), the one-step Sinkhorn-Knopp in (2), and the full Sinkhorn-Knopp (i.e., iterate Sinkhorn-Knopp until convergence) are all sub-routines of the EM algorithm to learn the PoG prior. By showing that each of methods supersedes its previous counterpart, we explain the performance gain phenomenon.

Second, based on the analysis of the PoG patch prior, we propose a new denoising algorithm called the PoG Non-local Means (PoG-NLM). We show that PoG-NLM does not only subsumes 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 PoG-NLM algorithm.

The PoG model discussed in this paper is an improved version of our earlier work in [16] which used a classical Gaussian mixture model (GMM). The problem of the GMM is that the mixture weights are not used for the actual denoising. This causes a gap between the clustering model and the

denoising step. The new PoG model resolves the issue by ensuring that the clustering model is used in the denoising step. As a result, clustering is now an integral part of the algorithm.

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

II. PRODUCT OF GAUSSIANS PATCH PRIOR

A. Notations

Throughout this paper, we use n to denote the number of pixels to be denoised, and k to denote the number of components of the PoG model. To avoid ambiguity, we shall call each component of the PoG a cluster. The running index $j \in \{1, \dots, n\}$ tracks the pixels, and $i \in \{1, \dots, k\}$ tracks the clusters. Vectors are represented by bold letters, and $\mathbf{1}$ denotes a constant vector of which all entries are one. We designate the vector $\mathbf{x}_j \in \mathbb{R}^2$ to represent the two-dimensional spatial coordinate of the j th pixel, the vector $\mathbf{y}_j \in \mathbb{R}^d$ to represent a d -dimensional patch centered at the j th pixel of the noisy image, and the scalar $y_j \in \mathbb{R}$ to represent the value of the center pixel of j th patch. Without loss of generality, we assume that all pixel intensity values have been normalized to the range $[0, 1]$.

The results presented in this paper are applicable to a wide range of smoothing filters. However, for simplicity we shall focus on smoothing filters \mathbf{W} of the following general form (See Table I for examples):

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

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

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

with mean $\mathbf{q}_i \in \mathbb{R}^p$ and covariance matrix $\Sigma_i \in \mathbb{R}^{p \times p}$. The vector $\mathbf{p}_j \in \mathbb{R}^p$ is a p -dimensional feature vector of the j th pixel of the noisy image.

Example 1: For non-local means [3], the feature vector \mathbf{p}_j is a concatenation of the spatial coordinates \mathbf{x}_j and the intensity values of the j th patch \mathbf{y}_j :

$$\mathbf{p}_j = [\mathbf{x}_j^T, \mathbf{y}_j^T]^T. \quad (5)$$

The i th mean vector and the i th covariance matrix are

$$\mathbf{q}_i = \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix}, \quad \text{and} \quad \Sigma_i = \begin{bmatrix} h_s^2 \mathbf{I} & 0 \\ 0 & h_r^2 \mathbf{I} \end{bmatrix} \stackrel{\text{def}}{=} \Sigma_{\text{NLM}}, \quad (6)$$

respectively, where h_s and h_r are parameters. Note that for NLM, the covariance matrices are identical.

B. Product of Gaussians (PoG) Model

Consider a set of n data points $\mathcal{P} \stackrel{\text{def}}{=} \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ where $\mathbf{p}_j \in \mathbb{R}^p$, we say that \mathcal{P} is generated from a PoG of k

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]	$\exp \left\{ -\left(\frac{\ \mathbf{x}_j - \mathbf{x}_i\ ^2}{2h_s^2} + \frac{(y_j - y_i)^2}{2h_r^2} \right) \right\}$
Non-local Means [3]	$\exp \left\{ -\left(\frac{\ \mathbf{x}_j - \mathbf{x}_i\ ^2}{2h_s^2} + \frac{\ \mathbf{y}_j - \mathbf{y}_i\ ^2}{2h_r^2} \right) \right\}$
LARK [4]	$\exp \left\{ -(\mathbf{x}_j - \mathbf{x}_i)^T \Sigma_i^{-1} (\mathbf{x}_j - \mathbf{x}_i) \right\}$

clusters if the distribution of \mathbf{p}_j is a product of k Gaussians with means $\{\boldsymbol{\mu}_i\}_{i=1}^k$ and covariance matrices $\{\mathbf{C}_{ij}\}_{i=1}^k$. Mathematically, by denoting the PoG parameters collectively as $\Theta = \{(\boldsymbol{\mu}_i, \mathbf{C}_{ij}) \mid i = 1, \dots, k, j = 1, \dots, n\}$, the conditional distribution of \mathbf{p}_j given Θ is

$$f(\mathbf{p}_j | \Theta) \propto \prod_{i=1}^k \exp \left\{ -\frac{1}{2} (\mathbf{p}_j - \boldsymbol{\mu}_i)^T \mathbf{C}_{ij}^{-1} (\mathbf{p}_j - \boldsymbol{\mu}_i) \right\}, \quad (7)$$

where the proportionality accounts for the normalization constant which depends on \mathbf{C}_{ij} .

In general, both $\{\boldsymbol{\mu}_i\}_{i=1}^k$ and $\{\mathbf{C}_{ij}\}_{i,j=1}^{k,n}$ have to be learned from the data \mathcal{P} . However, for the purpose of this paper, which is to analyze the performance gain phenomenon, we assume that \mathbf{C}_{ij} takes a special form as follows.

Assumption 1: For any j , we assume that $\mathbf{C}_{ij}^{-1} = \gamma_{ij} \mathbf{I}$ where

$$\gamma_{ij} \stackrel{\text{def}}{=} \frac{\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \Sigma_i)}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \Sigma_i)}, \quad (8)$$

and Σ_i is the covariance matrix defined according to the smoothing filter, e.g., (6) for non-local means.

If we apply assumption 1 to (7), it becomes clear that

$$f(\mathbf{p}_j | \Theta) \propto \prod_{i=1}^k \exp(-\gamma_{ij} \|\mathbf{p}_j - \boldsymbol{\mu}_i\|^2). \quad (9)$$

The reason of defining γ_{ij} as in (8) is to ensure that γ_{ij} is a function of $\boldsymbol{\mu}_i$. If γ_{ij} is independent of $\boldsymbol{\mu}_i$, then one can show that learning the pair $(\boldsymbol{\mu}_i, \gamma_{ij})$ in (9) is equivalent to solving a non-negative matrix factorization problem¹. Of course, solving such a matrix factorization problem is a valid step to learn Θ . However, the associated EM algorithm can no longer be used to study the symmetrization effect — which will defeat the goal of this paper. Therefore, in the rest of the paper we assume (8) holds.

Since EM algorithm is a known method, we refer interested readers to the tutorial by Gupta and Chen [24] for a gentle introduction. We now present the EM algorithm to learn $\Theta \stackrel{\text{def}}{=} \{\boldsymbol{\mu}_i\}_{i=1}^k$ from \mathcal{P} .

¹For any fixed \mathbf{p}_j , the norm-square $\|\mathbf{p}_j - \boldsymbol{\mu}_i\|^2$ is a scalar indexed by i . Let $v_i = \|\mathbf{p}_j - \boldsymbol{\mu}_i\|^2$ be such scalar. Then, the sum $\sum_i \gamma_{ij} v_i$ is a matrix-vector multiplication of $\Gamma = (\gamma_{ij})$ and $\mathbf{v} = (v_i)$. Since γ_{ij} and v_i are non-negative, finding γ_{ij} and v_i simultaneously is equivalent to solving a non-negative matrix factorization.

Proposition 1: Under Assumption 1, the E-step and the M-step of learning $\Theta = \{\boldsymbol{\mu}_i\}_{i=1}^k$ from $\mathcal{P} = \{\mathbf{p}_j\}_{j=1}^n$ are

E-Step:

$$\pi_{ij}^{(t)} = \frac{\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)}, \quad (10)$$

M-Step:

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^n \pi_{ij}^{(t)} \mathbf{p}_j}{\sum_{j=1}^n \pi_{ij}^{(t)}}, \quad (11)$$

respectively, where the superscript $(\cdot)^{(t)}$ denotes the t -th iteration of the EM algorithm.

Proof: See Appendix A. \blacksquare

C. Patch-based Denoising Algorithm

The PoG model specified by (9) allows us to derive various image denoising algorithms by solving a maximum-a-posterior (MAP) estimation problem. In particular, since (9) is a prior model of \mathbf{p}_j , we can consider the following MAP formulation:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 - \sum_{j=1}^n \log f(\mathbf{p}_j | \Theta), \quad (12)$$

where λ is a parameter, and the summation over j follows from the independence of \mathbf{p}_j . Substituting (9) into (12) yields

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \|\mathbf{P}_j \mathbf{z} - \boldsymbol{\mu}_i\|^2, \quad (13)$$

where \mathbf{P}_j is a matrix which extracts the j th feature vector from the image \mathbf{z} , i.e., $\mathbf{P}_j \mathbf{z} = \mathbf{p}_j$.

Example 2: If we use \mathbf{P}_j to extract the j th pixel of \mathbf{z} instead of extracting a feature vector, then $\mathbf{P}_j \mathbf{z} = z_j$. Moreover, if we let $\lambda = 0$ and $\boldsymbol{\mu}_i = y_i$, then (13) can be simplified as

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} (z_j - y_i)^2. \quad (14)$$

Letting $\gamma_{ij} = W_{ij}$ (with $k = n$), the i th entry of $\hat{\mathbf{z}}$ becomes

$$\hat{z}_i = \frac{\sum_{j=1}^n W_{ij} y_j}{\sum_{j=1}^n W_{ij}}, \quad (15)$$

which is exactly the non-local means.

Example 3: If we substitute $f(\mathbf{p}_j | \Theta)$ in (12) with the standard Gaussian mixture model (GMM) instead of the PoG model, then we will obtain an MAP estimation with an expected patch log-likelihood prior (EPLL) [22]:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 - \sum_{j=1}^n \log \left(\sum_{i=1}^k w_i \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i, \mathbf{C}_i) \right), \quad (16)$$

where w_i is the weight of the i th Gaussian mixture, $\boldsymbol{\mu}_i$ is the i th mean, and \mathbf{C}_i is the i th covariance matrix. We will compare the difference between EPLL and PoG in Section V.

III. GENERALIZATION OF SYMMETRIC FILTERS

In this section, we discuss how various symmetric smoothing filters can be generalized by the EM algorithm.

A. Standard Non-Local Means

We first study the non-local means (NLM) smoothing filter. To establish the relationship between PoG and NLM, we initialize the EM algorithm with $\boldsymbol{\mu}_i^{(0)} = \mathbf{p}_i$ and choose $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{\text{NLM}}$. However, instead of defining $\pi_{ij}^{(0)}$ using (10), we initialize $\pi_{ij}^{(0)}$ as

$$\pi_{ij}^{(0)} = \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(0)}, \boldsymbol{\Sigma}_{\text{NLM}}). \quad (17)$$

Letting $\gamma_{ij} = \pi_{ij}^{(0)}$, and using (3) together with the conditions in Example 2, we observe that the denoised image of this (zero-iteration) EM algorithm is

$$\hat{z}_i = \underset{z_i}{\operatorname{argmin}} \sum_{j=1}^n \pi_{ij}^{(0)} (z_i - y_j)^2 = \frac{\sum_{j=1}^n W_{ij} y_j}{\sum_{j=1}^n W_{ij}}, \quad (18)$$

which is identical to (15), i.e., it is the NLM solution.

The PoG perspective of NLM provides a way to understand the underlying principles of NLM. The variable $\pi_{ij}^{(0)} = \mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})$ is an un-normalized conditional probability that the j th patch \mathbf{p}_j belongs to the i th Gaussian component. Note that $\mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})$ is not a legitimate probability because summing over i is not equal to 1. The action of $\pi_{ij}^{(0)}$ is to implicitly *group* patches according to their distance to the centers of the Gaussians. However, such grouping has only limited effectiveness because the center $\boldsymbol{\mu}_i^{(0)} = \mathbf{p}_i$ is the noisy data itself. Moreover, the grouping is performed only once because there is no EM iteration. Therefore, it is reasonable to expect a limited performance of NLM.

B. One-step Sinkhorn-Knopp

To analyze the one-step Sinkhorn-Knopp, we first express (2) as two consecutive steps:

$$[\mathbf{W} \mathbf{D}_c^{-1}]_{ij} = \frac{W_{ij}}{\sum_{i=1}^n W_{ij}} = \frac{\mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}{\sum_{i=1}^n \mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}, \quad (19)$$

$$[\mathbf{D}_r^{-1} \mathbf{W} \mathbf{D}_c^{-1}]_{ij} = \frac{[\mathbf{W} \mathbf{D}_c^{-1}]_{ij}}{\sum_{j=1}^n [\mathbf{W} \mathbf{D}_c^{-1}]_{ij}}. \quad (20)$$

Inspecting the equations, we observe that (19) is equivalent to the E-step with the initializations $\boldsymbol{\mu}_i^{(0)} = \mathbf{p}_i$. In this case,

$$\pi_{ij}^{(0)} = \frac{\mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})}{\sum_{i=1}^n \mathcal{N}(\mathbf{p}_j | \mathbf{p}_i, \boldsymbol{\Sigma}_{\text{NLM}})} = [\mathbf{W} \mathbf{D}_c^{-1}]_{ij}. \quad (21)$$

Substituting (21) into (14), the i th pixel of the denoised image is

$$\hat{z}_i = \frac{\sum_{j=1}^n \pi_{ij}^{(0)} y_j}{\sum_{j=1}^n \pi_{ij}^{(0)}} = \frac{\sum_{j=1}^n [\mathbf{W} \mathbf{D}_c^{-1}]_{ij} y_j}{\sum_{j=1}^n [\mathbf{W} \mathbf{D}_c^{-1}]_{ij}}. \quad (22)$$

By identifying (21) with (10), and (22) with (11), it becomes clear that the column normalization (19) is the E-step, and the row normalization (20) is the M-step.

The column normalization ensures that π_{ij} is a legitimate conditional probability, and hence the EM algorithm of learning the PoG model is valid. To demonstrate the difference between standard NLM and one-step Sinkhorn-Knopp, we consider the following example.

Example 4: Smoothing filters can equivalently be considered as graphs [5]. In Figure 3, we consider a cluster of 4 data points $\{\mathbf{p}_1, \dots, \mathbf{p}_4\}$ and two cluster centers $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2\}$. Without loss of generality let us assume that all these 4 data points belong to the first cluster. The weight of the edge linking $\boldsymbol{\mu}_i$ and \mathbf{p}_j is denoted as W_{ij} (See Figure 3(a)). In this example, we assume that

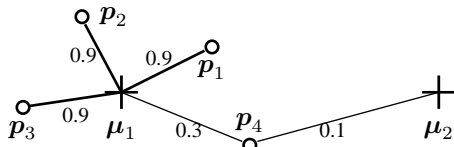
$$(W_{ij}) = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}.$$

Thus, $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ have strong connections with $\boldsymbol{\mu}_1$ but \mathbf{p}_4 only has a weak connection. For the other cluster $\boldsymbol{\mu}_2$, all $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ have a weak connection.

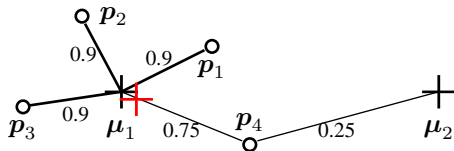
Now, if we apply column normalization to W_{ij} we can obtain a column-normalized graph $\pi_{ij} = W_{ij} / \sum_i W_{ij}$, i.e.,

$$(\pi_{ij}) = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.75 \\ 0.1 & 0.1 & 0.1 & 0.25 \end{bmatrix}.$$

The difference between (W_{ij}) and (π_{ij}) can be seen from the new empirical means $\hat{\boldsymbol{\mu}}_i \stackrel{\text{def}}{=} \sum_j W_{ij} \mathbf{p}_j / \sum_j W_{ij}$. In the case of (W_{ij}) , we observe that $\hat{\boldsymbol{\mu}}_1$ is a point located near to $\mathbf{p}_1, \mathbf{p}_2$ and \mathbf{p}_3 because of the relatively small edge of W_{14} compared to W_{11}, W_{12} and W_{13} . However, for the case of (π_{ij}) , we observe that $\hat{\boldsymbol{\mu}}_1$ is shifted towards \mathbf{p}_4 because π_{14} has a similar value compared to π_{11}, π_{12} and π_{13} . Putting this in a more qualitative description, the column normalization tries to increase the influence of less popular units in the graph, or to “balance out” the influence of the minority.



(a) Standard NLM.



(b) One-step Sinkhorn-Knopp.

Fig. 3: Comparison between standard NLM and one-step Sinkhorn-Knopp. The red cross represents the new position of the updated mean.

C. Full Sinkhorn-Knopp

The full Sinkhorn-Knopp (Algorithm 1) is an iterative algorithm that repeatedly applies the one-step Sinkhorn-Knopp until convergence. To establish its link to the EM algorithm, we first consider a modification to the EM algorithm.

In the M-step, instead of computing the actual mean $\boldsymbol{\mu}_i^{(t+1)}$, we define an intermediate variable $\beta_{ij}^{(t+1)}$ as

M-step:

$$\beta_{ij}^{(t+1)} = \frac{\pi_{ij}^{(t)}}{\sum_{j=1}^n \pi_{ij}^{(t)}}. \quad (23)$$

Consequently, the E-step becomes

E-step:

$$\pi_{ij}^{(t+1)} = \frac{\beta_{ij}^{(t+1)}}{\sum_{i=1}^n \beta_{ij}^{(t+1)}}. \quad (24)$$

Clearly, (23)-(24) is equivalent to the full Sinkhorn-Knopp iteration. However, as (23) require an intermediate variable $\beta_{ij}^{(t)}$, the modified EM steps do not correspond to a valid EM algorithm for learning the PoG model.

To understand the difference between the modified EM algorithm and the original EM algorithm, we note from (24) and (11) that $\beta_{ij}^{(t+1)}$ is an approximation of the likelihood $\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)$. In fact, if we compute the actual mean $\boldsymbol{\mu}_i^{(t+1)}$ (which is never explicitly calculated in the modified EM), we can show that

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^n \pi_{ij}^{(t)} \mathbf{p}_j}{\sum_{j=1}^n \pi_{ij}^{(t)}} \stackrel{(a)}{=} \sum_{j=1}^n \beta_{ij}^{(t+1)} \mathbf{p}_j, \quad (25)$$

where (a) follows from (23). (25) indicates that $\boldsymbol{\mu}_i^{(t+1)}$ is a linear combination of $\{\mathbf{p}_j\}_{j=1}^n$, with $\{\beta_{ij}^{(t+1)}\}_{j=1}^n$ being the combination weights. Therefore, while the modified EM algorithm does not explicitly the new means $\boldsymbol{\mu}_i^{(t+1)}$, there is an implicit grouping of pixels performed via the interactions between $\beta_{ij}^{(t)}$ and $\pi_{ij}^{(t)}$.

The fact that the full Sinkhorn-Knopp is not a valid EM algorithm for learning the PoG offers some insights into the performance gain phenomenon. Recall from the results in Figure 2, we observe that not all images have increasing PSNR when more Sinkhorn-Knopp iterations are performed. In fact, the PSNRs increase monotonically only for images 1-3, which are artificial images. These images are common in that there are only a few distinctive clusters. Therefore, the $\beta_{ij}^{(t)}$'s are good approximations to the true likelihood $\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)$. However, for other images where the clusters are not so distinctive, the $\beta_{ij}^{(t)}$'s have worse approximation and so the PSNR would drop.

D. Summary

To summarize our findings, we observe that the performance of the normalization is directly related to *how the EM algorithm is being implemented*. For standard NLM, there is no EM iteration and the weights are not the legitimate probabilities. Thus its performance is the worst. The situation improves when we apply a column normalization in Sinkhorn-Knopp, because the one-step Sinkhorn-Knopp is equivalent to one iteration of the EM. However, when extending to full Sinkhorn-Knopp, the performance drops again because the full Sinkhorn-Knopp is only an approximated EM algorithm.

We summarize our results in Table II. Besides the performance gain phenomenon, we have two additional observations.

TABLE II: Generalization and comparisons using EM algorithm for learning GMM with known Σ .

		Standard NLM [3]	One-Step Sinkhorn-Knopp [15]	Full Sinkhorn-Knopp [18]	PoG-NLM (generalization)
No. Clusters		$k = n$	$k = n$	$k = n$	$k < n$ (cross-validation)
Initialization	$\pi_{ij}^{(0)}$ $\mu_i^{(0)}$	$\mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})$	$\frac{\mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}$	$\frac{\mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}$	$\frac{\mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j \mu_i^{(0)}, \Sigma_{\text{NLM}})}$ randomly picked \mathbf{p}_i
E-step	Update $\pi_{ij}^{(t)}$	×	✓	✓ (via $\beta_{ij}^{(t)}$)	✓
M-step	Update $\mu_i^{(t)}$	×	×	✓ (implicitly)	✓
No. Iterations		0	1	Many	Many
Denoising Parameters	λ $\mathbf{P}_j \mathbf{z}$ γ_{ij}	0 z_j $\pi_{ij}^{(0)}$	0 z_j $\pi_{ij}^{(0)}$	0 z_j $\pi_{ij}^{(t)}$	by SURE \mathbf{p}_j $\pi_{ij}^{(t)}$

First, in all three symmetrization schemes, the number of clusters k is equal to the number of data points n . This setting is suboptimal because ideally we would like to group multiple pixels into one cluster, and hence we like to have $k < n$. Second, since none of the three symmetrization schemes is the complete EM algorithm, it is natural that a new denoising algorithm can be designed based on the complete EM algorithm. As a preview we show and compare the proposed method in the right most column of Table II.

IV. PRODUCT OF GAUSSIAN NON-LOCAL MEANS

The proposed denoising algorithm is called the Product of Gaussian Non-local Means (PoG-NLM). The idea of PoG-NLM is to run the full EM algorithm (10)-(11) for the NLM filter defined in Example 2 and solve the MAP optimization (13). In the following subsections we will discuss how the PoG-NLM is designed.

A. PoG-NLM

First of all, because the non-local means feature vector contains a spatial component and an intensity component, (13) can be written as

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \left\| \begin{bmatrix} \mathbf{x}_j \\ \mathbf{z}_j \end{bmatrix} - \begin{bmatrix} \mu_i^{(s)} \\ \mu_i^{(r)} \end{bmatrix} \right\|^2, \quad (26)$$

where λ is a parameter, \mathbf{x}_j is the spatial coordinate of the j th pixel, \mathbf{z}_j is the j th patch of the denoised image, $\mu_i^{(s)}$ is the spatial component of the i th cluster center μ_i , and $\mu_i^{(r)}$ is the intensity component of μ_i . Since \mathbf{x}_j is independent of the optimization and thus can be eliminated without changing the objective function. Moreover, if we define a matrix $\mathbf{P}_j \in \mathbb{R}^{d \times n}$ as a patch-extract operator that extracts the j th patch of the image, i.e., $\mathbf{P}_j \mathbf{z} = \mathbf{z}_j$, then (26) becomes

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \lambda \|\mathbf{z} - \mathbf{y}\|^2 + \sum_{j=1}^n \sum_{i=1}^k \gamma_{ij} \left\| \mathbf{P}_j \mathbf{z} - \mu_i^{(r)} \right\|^2. \quad (27)$$

The optimization in (27) is a quadratic minimization and so closed-form solution exists. By considering the first order

optimality condition we obtain a normal equation

$$\left(\sum_{j=1}^n \mathbf{P}_j^T \mathbf{P}_j + \lambda \mathbf{I} \right) \hat{\mathbf{z}} = \sum_{j=1}^n \mathbf{P}_j^T \mathbf{w}_j + \lambda \mathbf{y}, \quad (28)$$

of which the solution is the minimizer of (27). In (28), the vector \mathbf{w}_j is defined as

$$\mathbf{w}_j \stackrel{\text{def}}{=} \sum_{i=1}^k \gamma_{ij} \mu_i^{(r)}. \quad (29)$$

To interpret (28)-(29), we note that $\hat{\mathbf{z}}$ is a weighted average of $\{\mu_i^{(r)}\}_{i=1}^k$. The operation of \mathbf{P}_j 's represents an aggregation of the spatially overlapping estimates (c.f., [25]). The addition of $\lambda \mathbf{y}$ regulates the cluster centers by adding a small amount of fine features, depending on the magnitude of λ .

In order to use (28), two technical issues must be resolved: (i) Determine λ ; (ii) Determine k .

B. Parameter Estimation

We now discuss how to choose λ . Ideally, λ 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) [26], [27]. SURE is a consistent and unbiased estimator of the MSE. That is, SURE converges to the true MSE as the number of observations. 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 first make the following assumption.

Assumption 2: We assume that the patch-extract operator $\{\mathbf{P}_j\}_{j=1}^n$ satisfies the following approximation:

$$\sum_{j=1}^n \mathbf{P}_j^T \mathbf{P}_j = d \mathbf{I}. \quad (30)$$

We note that Assumption 2 only affects the boundary pixels and not the interior pixels. Intuitively, what Assumption 2 does is to require that the boundary pixels of the image

are periodically padded instead of zero-padded. In image restoration literature, periodic boundary pad is common when analyzing deblurring methods, e.g., [28].

Under Assumption 2, we can substitute (30) into (28) and take the matrix inverse. This would yield

$$\hat{z}(\lambda) = \frac{d}{d+\lambda} \mathbf{u} + \frac{\lambda}{d+\lambda} \mathbf{y}, \quad (31)$$

where

$$\mathbf{u} \stackrel{\text{def}}{=} \frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \mathbf{w}_j. \quad (32)$$

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

Proposition 2: Under Assumption 2, the SURE of $\hat{z}(\lambda)$ is

$$\text{SURE}(\lambda) = -\sigma^2 + \hat{\sigma}^2 \left(\frac{d}{d+\lambda} \right)^2 + \frac{2\sigma^2}{n} \left(\frac{\text{div}(\mathbf{u})d + n\lambda}{d+\lambda} \right), \quad (33)$$

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

$$\text{div}(\mathbf{u}) \stackrel{\text{def}}{=} \mathbf{1}_{n \times 1}^T \left(\frac{1}{d} \sum_{j=1}^n \mathbf{P}_j^T \left(\sum_{i=1}^k \gamma_{ij} \left(\frac{\sum_{j=1}^n \gamma_{ij} \mathbf{e}_j}{\sum_{j=1}^n \gamma_{ij}} \right) \right) \right), \quad (34)$$

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

Proof: See Appendix B. ■

The SURE given in (33) is a one-dimensional function in λ . The minimizer can be determined in closed-form.

Corollary 1: The optimal λ that minimizes SURE(λ) is

$$\lambda^* = \max \left(d \left(\left(\frac{\hat{\sigma}^2}{\sigma^2} \right) \left(\frac{n}{n - \text{div}(\mathbf{u})} \right) - 1 \right), 0 \right). \quad (35)$$

Proof: (33) is a differentiable function in λ . Therefore, the minimizer can be determined by considering the first order optimality and set the derivative of SURE(λ) to zero. The projection operator $\max(\cdot, 0)$ is placed to ensure that $\lambda^* \geq 0$. ■

Example 5: 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 128×128 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 5×5 . The results are shown in Figure 4, 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 1: Careful readers may notice that in (34), we implicitly assume that γ_{ij} is independent of \mathbf{y}_j . This implicit assumption is generally not valid if γ_{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 γ_{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 [29].

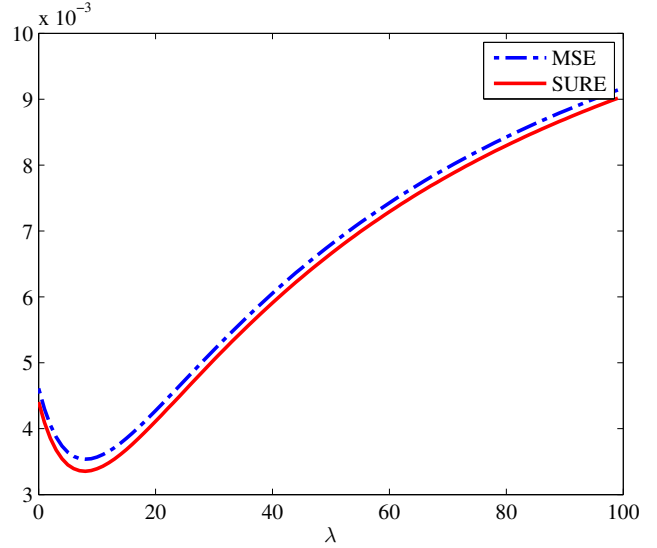


Fig. 4: Comparison between SURE and the ground truth MSE of a denoising problem.

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

$$\hat{\Sigma}_i = \frac{\sum_{j=1}^n \gamma_{ij} (\mathbf{p}_j - \boldsymbol{\mu}_i) (\mathbf{p}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^n \gamma_{ij}}, \quad (36)$$

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

$$\delta_i(k) = \frac{1}{d} \text{Tr} \left\{ \Sigma_{\text{NLM}}^{-1} \hat{\Sigma}_i \right\}. \quad (37)$$

Ideally, if $\hat{\Sigma}_i = \Sigma_{\text{NLM}}$, then by (37) 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 Σ_{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

$$\delta(k) = \frac{1}{k} \sum_{i=1}^k \delta_i(k). \quad (38)$$

The parenthesis (k) in (38) emphasizes that both $\delta(k)$ and $\delta_i(k)$ are functions of k . With (38), 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 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

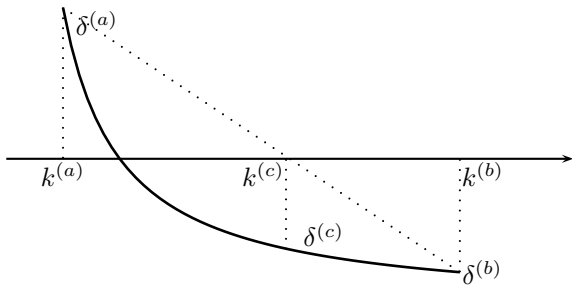


Fig. 5: Illustration of the secant method. Given $k^{(a)}$ and $k^{(b)}$, we compute $k^{(c)}$ according to the slope defined by the line linking $\delta^{(a)}$ and $\delta^{(b)}$.

Algorithm 2 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 (39).
  Compute  $\delta^{(c)} \stackrel{\text{def}}{=} \delta(k^{(c)})$  according to (38).
  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

```

$\delta^{(b)} < 1$, the secant method computes the new k as

$$k^{(c)} = \frac{k^{(a)}(\delta^{(b)} - 1) - k^{(b)}(\delta^{(a)} - 1)}{\delta^{(b)} - \delta^{(a)}}. \quad (39)$$

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 5. A pseudo code is given in Algorithm 2.

Example 6: To verify the effectiveness of the proposed cross validation scheme, we consider a 128×128 ‘‘House’’ image with noise $\sigma = 60/255$. The patch size is 5×5 , $h_r = \sigma$, and $h_s = 10$. Figure 6 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 $\text{PSNR} = 26.0257\text{dB}$. 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 PoG-NLM.

A. Experiment Settings

We consider 10 testing images, each of which has size 128×128 (so $n = 16384$). The noise standard deviations are set as $\sigma \in \{20/255, 40/255, 60/255, 80/255, 100/255\}$.

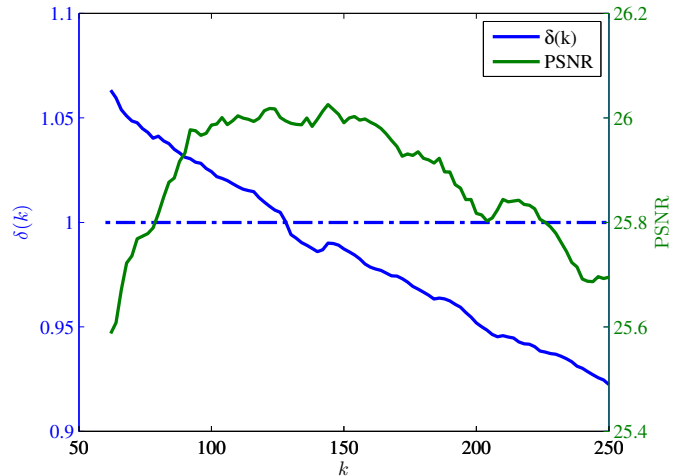


Fig. 6: Comparison between the cross validation score $\delta(k)$ and the true PSNR value as a function of k . The horizontal dashed line indicates the intersection at $\delta(k) = 1$.

TABLE III: Configurations of Methods

Method	Configuration
NLM [3]	Patch size 5×5 , $h_s = 10$, $h_r = \sigma\sqrt{d}$
One-step [15]	Patch size 5×5 , $h_s = 10$, $h_r = \sigma\sqrt{d}$
PoG-NLM	Patch size 5×5 , $h_s = 10$, $h_r = \sigma$
GLIDE [9]	Default settings. Pilot estimate uses NLM.
BM3D [25]	Default settings.
EPLL [22]	Default settings. External Database.

Several existing denoising algorithms are studied, namely the standard NLM [3], One-step Sinkhorn-Knopp [15], BM3D [25], EPLL [22], and Global image denoising (GLIDE) [9]. The parameters of the methods are configured as shown in Table III. For NLM and Sinkhorn, 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 the proposed PoG-NLM, we keep the same settings as NLM and Sinkhorn 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 and GLIDE, we downloaded the original MATLAB code from the author’s website^{2,3,4}. Default settings of these algorithms are used. Among these methods, we remark 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 PoG-NLM) are single image denoising algorithms.

B. Comparison with NLM and One-step Sinkhorn-Knopp

The overall results of the experiment are shown in Table IV. We first compare the PSNR values of the proposed method

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

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

⁴GLIDE: <https://users.soe.ucsc.edu/~htalebi/GLIDE.php>

TABLE IV: Denoising results of Standard NLM [3], One-step Sinkhorn-Knopp [15], BM3D [25], EPLL [22], Global image denoising [9], and the proposed PoG-NLM.

	NLM [3]	Sinkhorn [15]	Ours	GLIDE [9]	BM3D [25]	EPLL [22]	NLM [3]	Sinkhorn [15]	Ours	GLIDE [9]	BM3D [25]	EPLL [22]
σ	<i>Baboon</i>						<i>Barbara</i>					
20	24.53	25.01	26.84	26.51	26.96	27.19	26.05	27.02	29.43	28.64	29.42	29.40
40	22.32	22.55	24.49	24.04	24.57	24.56	21.48	21.91	25.41	24.83	25.35	25.79
60	21.31	21.46	23.32	22.87	23.53	23.44	19.31	19.55	23.28	22.33	23.55	23.64
80	20.76	20.87	22.51	22.22	22.77	22.66	18.25	18.38	21.83	20.92	22.30	22.11
100	20.43	20.52	21.98	20.68	22.14	22.09	17.68	17.76	20.77	19.82	21.30	21.00
σ	<i>Boat</i>						<i>Bridge</i>					
20	24.88	26.05	28.43	27.55	28.58	28.76	23.99	24.95	26.90	26.34	27.09	27.25
40	21.97	22.39	24.96	24.35	25.12	25.32	20.87	21.35	23.85	23.18	23.88	24.19
60	20.46	20.70	23.19	22.59	23.47	23.56	19.58	19.85	22.24	21.47	22.44	22.48
80	19.60	19.74	22.14	21.42	22.43	22.41	18.83	19.01	21.20	20.44	21.45	21.42
100	19.09	19.18	21.34	20.50	21.74	21.53	18.35	18.48	20.46	19.75	20.67	20.66
σ	<i>Couple</i>						<i>Hill</i>					
20	24.54	25.62	28.20	27.25	28.42	28.60	25.51	26.38	28.68	27.98	28.82	28.97
40	21.67	22.10	24.64	23.95	25.00	25.11	22.58	23.11	25.55	24.79	25.70	25.85
60	20.35	20.60	23.07	22.32	23.36	23.37	21.33	21.69	23.95	23.26	24.21	24.16
80	19.64	19.81	22.02	21.40	22.32	22.30	20.68	20.93	22.90	22.42	23.19	23.12
100	19.24	19.35	21.23	19.80	21.56	21.52	20.29	20.49	22.07	21.85	22.37	22.39
σ	<i>House</i>						<i>Lena</i>					
20	28.20	30.02	32.92	31.82	32.73	32.47	26.90	28.03	29.83	29.19	29.93	30.06
40	23.26	24.27	28.31	27.31	28.91	28.77	22.40	23.11	26.40	25.96	26.23	26.70
60	21.40	21.79	26.05	24.72	26.68	26.58	20.22	20.60	24.49	23.51	24.49	24.80
80	20.52	20.70	24.46	22.96	25.20	25.04	19.09	19.32	23.10	22.00	23.22	23.45
100	20.04	20.13	23.21	20.80	23.96	23.83	18.47	18.62	22.03	20.98	22.25	22.41
σ	<i>Man</i>						<i>Pepper</i>					
20	25.14	26.09	28.12	27.37	28.13	28.43	26.17	27.89	29.58	28.86	29.61	29.76
40	21.93	22.26	24.78	24.29	24.91	25.19	21.19	22.23	25.43	24.61	25.44	26.02
60	20.26	20.45	23.12	22.24	23.26	23.48	19.05	19.61	23.28	22.24	23.35	23.77
80	19.33	19.46	22.01	20.72	22.26	22.27	17.92	18.22	21.71	20.58	21.93	22.16
100	18.78	18.87	21.07	20.42	21.48	21.38	17.27	17.45	20.51	19.54	20.86	20.93

with NLM and One-step Sinkhorn.

In Table V we show the average PSNR over the 10 testing images. In this table, we observe that on average One-step Sinkhorn 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 PoG-NLM with NLM and One-step Sinkhorn, 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.

Besides studying the trend of PSNR as a function of σ , it is also interesting to compare the PSNR when we increase the spatial parameter h_s . In Table VI, we show the PSNR improvement when we use different $h_s \in \{5, 10, 20, 50, 100\}$ for a 128×128 image. The results show that when h_s increases, the PSNR improvement also increases. One reason is that in (6), 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 Sinkhorn-Knopp depends on how large h_s is.

C. Comparison with GLIDE

GLIDE [9] is a recently proposed method that implements a full Sinkhorn-Knopp iteration. In addition to Sinkhorn-Knopp, GLIDE also 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 pilot estimate. The result in Table VII shows that in general PoG-NLM has at least 0.65dB improvement over GLIDE. This result is consistent with our

TABLE V: PSNR comparison with different noise level σ . Averaged over 10 testing images. $h_s = 10$.

	NLM	Sinkhorn	Ours	PSNR ₂	PSNR ₃
σ	(PSNR ₁)	(PSNR ₂)	(PSNR ₃)	-PSNR ₁	-PSNR ₁
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

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

	NLM	Sinkhorn	Ours	PSNR ₂	PSNR ₃
h_s	(PSNR ₁)	(PSNR ₂)	(PSNR ₃)	-PSNR ₁	-PSNR ₂
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

observation that full Sinkhorn-Knopp is an incomplete EM algorithm.

D. Comparison with EPLL

Our next experiment is to compare the Gaussian mixture model (GMM) of EPLL [22] and the proposed PoG prior. In this experiment, we feed the noisy patches to two EM algorithms to learn a GMM and a PoG prior. The patch size is fixed at 5×5 , 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 algorithms.

From Table VIII, we observe that EPLL with a noisy input performs poorly. The reason should be clear because the GMM learned from the noisy patches is *not* the prior distribution of the clean patches. In contrast, while the PoG learned in the proposed method is also not the prior of the clean patches, it actually leads to better results.

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 IV). This result further shows the effectiveness of the proposed PoG-NLM for single image denoising.

E. Complexity and Limitations

Finally, we discuss the complexity and limitations of the proposed PoG-NLM.

PoG-NLM is a one-step denoising algorithm provided the PoG is given. However, learning the PoG using the EM algorithm is time-consuming, and the complexity depends on the number of clusters k . In addition, since k needs to

TABLE VII: PSNR comparison between GLIDE and PoG-NLM. Average over 10 testing images. $h_s = 10$.

	GLIDE	Ours	
σ	(PSNR ₁)	(PSNR ₂)	PSNR ₂ - PSNR ₁
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

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
σ	(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

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 PoG-NLM is that for low-noise images the computing time could be very long. However, PoG-NLM 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 PoG-NLM 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 PoG-NLM 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 PoG-NLM for small images, but becomes better as image size increases.

At this point readers may perhaps ask whether it is possible to take out the mean of the patches when learning the PoG, as it could reduce the number of clusters. However, from our experience, we find that this approach actually degrades the denoising performance. Our observation is that if the PoG is learned from a collection of zero-mean patches, the denoising step in (27) 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 PoG has a zero-mean) because the means are iteratively updated. We followed the same approach to iteratively update

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

σ	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”.

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

the means. However, we find that in general the denoising performance is still worse than the original PoG 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 Product of Gaussians (PoG) model. This observation allows us to generalize various symmetric smoothing filters including the standard 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 PoG non-local means (PoG-NLM). PoG-NLM is a simple modification of the NLM optimization framework by using the PoG prior for maximum-a-posteriori estimation. Equipped with a cross-validation scheme which can automatically determine the number of clusters, PoG-NLM shows consistently better denoising results than NLM, One-step Sinkhorn-Knopp and full Sinkhorn-Knopp. While PoG-NLM 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.

APPENDIX

A. Proof of Proposition 1

In order to derive the EM algorithm, we first need to specify the missing data. Referring to (9), we let Z_{ij} (with a realization z_{ij}) be the hidden random variable denoting the weight of each

Gaussian component. The distribution of Z_{ij} given $(\mathbf{p}_j, \boldsymbol{\mu}_i^{(t)})$ is a delta function:

$$\mathbb{P}(Z_{ij} = z_{ij} | \mathbf{p}_j, \boldsymbol{\mu}_i^{(t)}) = \begin{cases} 1, & \text{if } z_{ij} = \pi_{ij}^{(t)}, \\ 0, & \text{if } z_{ij} \neq \pi_{ij}^{(t)}, \end{cases}$$

where

$$\pi_{ij}^{(t)} \stackrel{\text{def}}{=} \frac{\mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)}{\sum_{i=1}^k \mathcal{N}(\mathbf{p}_j | \boldsymbol{\mu}_i^{(t)}, \boldsymbol{\Sigma}_i)}.$$

Denoting $\Theta = \{\boldsymbol{\mu}_i\}$, the complete data has a distribution

$$\log f(z_{ij}, \mathbf{p}_j | \Theta^{(t)}) = - \sum_{i=1}^k z_{ij} \|\mathbf{p}_j - \boldsymbol{\mu}_i^{(t)}\|^2.$$

At the t -th EM iteration, the j th Q -function is

$$\begin{aligned} Q_j(\Theta | \Theta^{(t)}) &\stackrel{\text{def}}{=} \mathbb{E}_{Z_{ij} | \mathbf{p}_j, \Theta^{(t)}} \left[\log f(z_{ij}, \mathbf{p}_j | \Theta^{(t)}) \right] \\ &= \mathbb{E}_{Z_{ij} | \mathbf{p}_j, \Theta^{(t)}} \left[- \sum_{i=1}^k z_{ij} \|\mathbf{p}_j - \boldsymbol{\mu}_i^{(t)}\|^2 \right] \\ &= - \sum_{i=1}^k \pi_{ij}^{(t)} \|\mathbf{p}_j - \boldsymbol{\mu}_i^{(t)}\|^2. \end{aligned}$$

Therefore, the overall Q -function is

$$\begin{aligned} Q(\Theta | \Theta^{(t)}) &= \sum_{j=1}^n Q_j(\Theta | \Theta^{(t)}) \\ &= - \sum_{j=1}^n \sum_{i=1}^k \pi_{ij}^{(t)} \|\mathbf{p}_j - \boldsymbol{\mu}_i^{(t)}\|^2, \end{aligned} \quad (40)$$

and hence the $(t+1)$ -th update of Θ is

$$\Theta^{(t+1)} = \underset{\Theta}{\text{argmax}} Q(\Theta | \Theta^{(t)}). \quad (41)$$

Since (40) is a sum of quadratic functions, each $\boldsymbol{\mu}_i$ in (41) can be solved individually as

$$\boldsymbol{\mu}_i^{(t+1)} = \underset{\boldsymbol{\mu}_i}{\text{argmin}} \sum_{j=1}^n \pi_{ij}^{(t)} \|\mathbf{p}_j - \boldsymbol{\mu}_i\|^2 = \frac{\sum_{j=1}^n \pi_{ij}^{(t)} \mathbf{p}_j}{\sum_{j=1}^n \pi_{ij}^{(t)}}.$$

B. Proof of Proposition 2

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 (31) 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 (31), 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 (32), (29) and (11) 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] 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.
- [9] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] P. Milanfar, "Symmetrizing smoothing filters," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, 2013.
- [15] 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.
- [16] 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.
- [17] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, pp. 876–879, 1964.
- [18] R. Sinkhorn and P. Knopp, "Concerning non-negative matrices and doubly-stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, pp. 343–348, 1967.
- [19] A. Cohen, "All admissible linear estimates of the mean vector," *Annals of Mathematical Statistics*, vol. 37, no. 2, pp. 458–463, Apr. 1966.
- [20] A. Buja, T. Hastie, and R. Tibshirani, "Linear smoothers and additive models," *Annals of Statistics*, vol. 17, pp. 453–510, 1989.
- [21] 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.
- [22] 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.
- [23] K. M. Taylor and F. G. Meyer, "A random walk on image patches," *SIAM Journal on Imaging Sciences*, vol. 5, pp. 688–725, 2012.
- [24] 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.
- [25] 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.
- [26] C. Stein, "Estimation of the mean of a multivariate normal distribution," *Annals of Statistics*, vol. 9, pp. 1135–1151, 1981.
- [27] 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.
- [28] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *European Conference on Computer Vision*, 2014, pp. 783–798.
- [29] L. Wasserman, *All of Nonparametric Statistics*, Springer, 2005.