MMSE Estimation for Poisson Noise Removal in Images

Stanislav Pyatykh* and Jürgen Hesser

Abstract—**Poisson noise suppression is an important prepro-cessing step in several applications, such as medical imaging, microscopy, and astronomical imaging. In this work, we propose a novel patch-wise Poisson noise removal strategy, in which the MMSE estimator is utilized in order to produce the denoising result for each image patch. Fast and accurate computation of the MMSE estimator is carried out using k-d tree search followed by search in the K-nearest neighbor graph. Our experiments show that the proposed method is the preferable choice for low signal-to-noise ratios.**

Index Terms—**Estimation, image processing, image denoising.**

I. INTRODUCTION

Images corrupted with Poisson noise appear in many ap-plications, such as medical imaging, fluorescence microscopy, and astronomical imaging. There, the noise removal quality plays an essential role for the further image processing steps. For Poisson noise, the noise variance equals the true image intensity, which means that noise is signal-dependent and the signal-to-noise ratio equals the square root of the image intensity. For this reason, the methods developed for additive white Gaussian noise are not applicable for Poisson noise directly.

Many of currently existing approaches reduce the Poisson noise removal to the Gaussian noise removal. This is an attractive strategy since the state of the art algorithms for Gaussian noise suppression are very efficient: it was shown in [1], [2] that they have almost achieved theoretical performance boundary. The reduction of the denoising problem for Poisson noise to that for Gaussian noise is based on the variance sta-bilization [3], [4], [5]. First, a pixel-wise variance-stabilizing transformation (such as the Anscombe root transformation [6]) is applied in order to transform the input image with Poisson noise into the image with signal-independent Gaussian noise. Second, a method for Gaussian noise removal is utilized (e.g. [7], [8], [9]). Finally, an inverse transformation is applied. In [4] and [10], it was pointed out that application of the algebraic inverse introduces a bias and the exact unbiased inverse was developed.

Nevertheless, the variance stabilization becomes inaccurate as the image intensity decreases, which limits the applica-bility of the above mentioned approaches. This fact led to development of denoising methods, which work with Poisson-distributed data directly. Among them, [11] and [12] are the

The authors are with the University Medical Center Mannheim, Heidelberg University, Theodor-Kutzer-Ufer 1-3, D-68167 Mannheim, Germany (e-mail: stanislav.pyatykh@medma.uni-heidelberg.de; juergen.hesser@medma.uni-heidelberg.de).

most efficient. First, these algorithms group similar image patches into clusters. Next, patches of each cluster are rep-resented using a small number of dictionary elements. Due to the projection onto the subspace defined by these dictionary elements, the noise components orthogonal to this subspace are canceled out. Therefore, the dimensionality of this subspace, i.e. the number of the dictionary elements used to represent each patch, should be made as small as possible. In [11], the dictionary elements are computed using Poisson principal component analysis of each patch cluster, which is based on the Bregman divergence for Poisson distribution. In [12], the dictionary elements are selected using a greedy algorithm, which searches for the maximum of the likelihood of the denoised patch given the input noisy patch.

In this paper, we propose a different technique for Poisson noise removal. Instead of projecting image patches onto a sub-space defined by a dictionary, the denoised patch is computed as the minimum mean square error (MMSE) estimator of the true patch given the noisy patch. If all possible noise-free image patches were available for the algorithm as prior infor-mation, such approach would provide the optimal estimate of the true patch. However, we show that it outperforms existing methods also with computationally tractable amount of prior information.

The MMSE estimation was considered in the image de-noising context in [1], [2], where it was used to compute the theoretical performance bounds of denoising algorithms. Since the direct computation of the MMSE estimator is very time-consuming, practical algorithms based on the MMSE estimation were not developed in these works. An optimized denoising algorithm based on the MMSE estimation was presented in [13] for the case of Gaussian noise, however, its speed drops as signal-to-noise ratio decreases. In this paper, we propose a fast and efficient method to compute the MMSE estimator for an image patch. This leads to a practical image denoising algorithm, which is able to process images in a reasonable time and outperforms existing approaches in terms of the output image quality.

II. MMSE ESTIMATION FOR IMAGE PATCHES

Consider an unknown noise-free image patch of size $\sqrt{d} \times \sqrt{d}$ rearranged into vector $\mathbf{x} \in \mathbb{R}^d$ and the corresponding patch of the input noisy image represented as vector $\mathbf{y} \in (\mathbb{N} \cup 0)^d$. We model patch $\mathbf{x}$ using random vector $\mathbf{X} \in \mathbb{R}^d$ with probability density function $f_{\mathbf{X}}(\cdot)$, whereas patch $\mathbf{y}$ is modeled by random vector $\mathbf{Y} \in (\mathbb{N} \cup 0)^d$ with probability mass function $f_{\mathbf{Y}}(\cdot)$. Assume that we know the likelihood

$P(\mathbf{Y}_i = y | \mathbf{X}_i = x)$, where $\mathbf{X}_i$ and $\mathbf{Y}_i$ are the i-th elements of $\mathbf{X}$ and $\mathbf{Y}$ respectively. In the case of Poisson noise,

$$P(\mathbf{Y}_i = y | \mathbf{X}_i = x) = \begin{cases} x^y e^{-x}/y! & x > 0 \\ 0 & x = 0, y > 0 \\ 1 & x = 0, y = 0 \end{cases} \quad (1)$$

Let us denote the peak intensity of the noise-free image by I_{max}. The problem to denoise patch $\mathbf{y}$ is typically formulated in terms of the peak signal-to-noise ratio (PSNR): we look for estimate $\hat{\mathbf{x}} \in \mathbb{R}^d$, which maximizes

$$\text{PSNR} = 10 \log_{10} \frac{I_{max}^2}{\text{MSE}} \quad (2)$$

or, equivalently, which minimizes the mean square error (MSE):

$$\begin{aligned} \text{MSE} &= E(\|\hat{\mathbf{x}} - \mathbf{X}\|^2 | \mathbf{Y} = \mathbf{y}) \\ &= \int \|\hat{\mathbf{x}} - \mathbf{u}\|^2 f_{\mathbf{X}}(\mathbf{u}|\mathbf{Y} = \mathbf{y}) d\mathbf{u}. \end{aligned} \quad (3)$$

Such $\hat{\mathbf{x}}$ is the MMSE estimator [1], which is computed as

$$\hat{\mathbf{x}} = E(\mathbf{X}|\mathbf{Y} = \mathbf{y}) = \int \mathbf{u} f_{\mathbf{X}}(\mathbf{u}|\mathbf{Y} = \mathbf{y}) d\mathbf{u}. \quad (4)$$

Using Bayes' theorem and the properties of marginal distributions, the latter expression can be rewritten as

$$\begin{aligned} \hat{\mathbf{x}} &= \frac{1}{f_{\mathbf{Y}}(\mathbf{y})} \int \mathbf{u} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}) f_{\mathbf{X}}(\mathbf{u}) d\mathbf{u} \\ &= \frac{\int \mathbf{u} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}) f_{\mathbf{X}}(\mathbf{u}) d\mathbf{u}}{\int f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}) f_{\mathbf{X}}(\mathbf{u}) d\mathbf{u}} \end{aligned} \quad (5)$$

where $f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u})$ is the probability of $\mathbf{y}$ given $\mathbf{u}$, which can be computed using (1):

$$f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}) = \prod_{i=1}^{d} P(\mathbf{Y}_i = \mathbf{y}_i | \mathbf{X}_i = \mathbf{u}_i). \quad (6)$$

Since the probability density function for image patches $f_{\mathbf{X}}(\cdot)$ cannot be accurately represented in some analytic form, direct computation of the integrals in (5) is not possible in practice. In order to approximate these integrals, one can use a large number of natural image patches $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_P)}$, which are treated as realizations of random vector $\mathbf{X}$ [1]:

$$\hat{\mathbf{x}}_{\text{app}} = \frac{\sum_{k=1}^{N_P} \mathbf{u}^{(k)} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}^{(k)})}{\sum_{k=1}^{N_P} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{u}^{(k)})}. \quad (7)$$

This expression gives a computationally feasible approximation for the MMMSE estimator, which can be utilized for denoising of patch $\mathbf{y}$.

III. EFFICIENT PATCH DENOISING

In order to approximate $\hat{\mathbf{x}}$ by $\hat{\mathbf{x}}_{\text{app}}$ accurately enough, one has to use about $N_P = 10^8$ natural image patches. Therefore, direct evaluation of the sums in (7) is very computationally expensive and, therefore, inapplicable. In this section, we present a fast and practical method to compute $\hat{\mathbf{x}}_{\text{app}}$.

A. Offline step

Several data structures are constructed based on patches $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_P)}$ at the offline (pre-processing) step.

First, patches $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_P)}$ are normalized since the intensity range of images from which they are extracted does not correspond to the intensity range of the input image. Specifically, each patch $\mathbf{u}^{(k)}$ is divided by the average of all intensities occurring in $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_P)}$, resulting in normalized patch $\overline{\mathbf{u}}^{(k)}$:

$$\begin{aligned} \overline{I} &= \frac{1}{d N_P} \sum_{k=1}^{N_P} \sum_{i=1}^{d} \mathbf{u}_i^{(k)} \\ \overline{\mathbf{u}}^{(k)} &= \mathbf{u}^{(k)}/\overline{I} \quad k = 1, \ldots, N_P. \end{aligned} \quad (8)$$

Next, we apply the k-means clustering to patch set $\overline{\mathbf{u}}^{(1)}, \ldots, \overline{\mathbf{u}}^{(N_P)}$. For each cluster j, it provides the cluster centroid $\mathbf{c}^{(j)}$ and the number of patches assigned to this cluster n_j. Since each patch $\overline{\mathbf{u}}^{(k)}$ can be approximated by the centroid of the cluster it is assigned to, (7) can be replaced with

$$\hat{\mathbf{x}}_{\text{app}} = \frac{\sum_{j=1}^{N_C} \mathbf{c}^{(j)} n_j f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{c}^{(j)})}{\sum_{j=1}^{N_C} n_j f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{c}^{(j)})} \quad (9)$$

where N_C is the number of clusters.

Then, centroids $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(N_C)}$ are used to construct a set of N_T randomized k-d trees as described in [14]. For each node of a randomized k-d tree, the splitting dimension is chosen randomly from several dimensions with the largest variance, so that each k-d tree represents a different space partitioning.

Additionally, the K-nearest neighbor graph is constructed using centroids $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(N_C)}$ as nodes. For each centroid $\mathbf{c}^{(j)}$, we compute the neighbor list containing the indexes of K centroids from $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(N_C)}$, which are the nearest to $\mathbf{c}^{(j)}$ in the Euclidean metric (excluding $\mathbf{c}^{(j)}$ itself).

B. Online step

In order to compute estimator (9) efficiently, we utilize the fact that patches $\mathbf{c}^{(j)}$ similar to input patch $\mathbf{y}$ have the largest likelihood $f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{c}^{(j)})$ and, therefore, the largest contribution to the sums in (9). On the other hand, likelihood $f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mathbf{c}^{(j)})$ is almost zero for patches $\mathbf{c}^{(j)}$, which are significantly different from $\mathbf{y}$, so that these patches have almost no influence on the result. Therefore, we start the calculation of (9) from patches $\mathbf{c}^{(j)}$, which are the closest to $\mathbf{y}$, and continue by processing patches farther away from $\mathbf{y}$ until convergence of the result.

This procedure is presented in Algorithm 1, which takes noisy patch $\mathbf{y}$ as the input and computes denoised patch $\hat{\mathbf{x}}_{\text{app}}$. Vector $\mathbf{s} \in \mathbb{R}^d$ represents the numerator of (9), whereas $w \in \mathbb{R}$ stores the denominator. First, centroids in the neighborhood of $\mathbf{y}$ are found by traversing each k-d tree (lines 6–9). The traversal starts from the root and goes to either the left or the right child depending on the location of $\mathbf{y}$ relative to the splitting plane until a leaf node is reached. Centroids $\{\mathbf{c}^{(j_m)}\}_{m=1}^{L}$ assigned to this node are then used to update $\mathbf{s}$ and w and added to the priority queue. Next, centroids farther away from $\mathbf{y}$ are found using the K-nearest neighbor graph (lines 10–14). At each iteration of the loop, the centroid with

Algorithm 1

1: $\mathbf{s} \leftarrow d$-dimensional zero vector
2: $w \leftarrow 0$
3: $\mu_{\mathbf{y}} \leftarrow \frac{1}{d} \sum_{i=1}^{d} \mathbf{y}_i$
4: $Q \leftarrow$ empty priority queue
5: Mark all clusters as unprocessed
6: **for each** k-d tree T **do**
7: $\quad \{j_m\}_{m=1}^{L} \leftarrow$ leaf of T containing $\mathbf{y}/\mu_{\mathbf{y}}$
8: $\quad$ ProcessClusters$\big(\{j_m\}_{m=1}^{L}\big)$
9: **end for**
10: **while** Q is not empty **and** w has not converged **do**
11: $\quad$ Pop element j^* with the highest priority from Q
12: $\quad \{j_m\}_{m=1}^{K} \leftarrow$ neighbors of $\mathbf{c}^{(j^*)}$
13: $\quad$ ProcessClusters$\big(\{j_m\}_{m=1}^{K}\big)$
14: **end while**
15: $\hat{\mathbf{x}}_{\mathrm{app}} \leftarrow \mathbf{s}/w$
16: **return** $\hat{\mathbf{x}}_{\mathrm{app}}$

Sub-Algorithm ProcessClusters$\big(\{j_m\}\big)$:

1: **for each** unprocessed cluster j_m **do**
2: $\quad \mathbf{s} \leftarrow \mathbf{s} + n_{j_m} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mu_{\mathbf{y}}\mathbf{c}^{(j_m)}) \cdot \mu_{\mathbf{y}}\mathbf{c}^{(j_m)}$
3: $\quad w \leftarrow w + n_{j_m} f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mu_{\mathbf{y}}\mathbf{c}^{(j_m)})$
4: $\quad$ Push j_m into Q with priority $f_{\mathbf{Y}}(\mathbf{y}|\mathbf{X} = \mu_{\mathbf{y}}\mathbf{c}^{(j_m)})$
5: $\quad$ Mark cluster j_m as processed
6: **end for**

the largest likelihood is removed from the priority queue and its previously unprocessed neighbors are examined. The loop repeats until the convergence criterion is met (line 10), namely, until the relative change of w during the last M iterations is less than 10^{-12}.

Note that centroids $\mathbf{c}^{(j)}$ were constructed using normalized patches $\overline{\mathbf{u}}^{(k)}$. Therefore, noisy patch $\mathbf{y}$ should be normalized in order to be utilized for the k-d tree search (line 7) and, vise versa, centroids $\mathbf{c}^{(j)}$ should be multiplied by the mean of $\mathbf{y}$ to compute $\mathbf{s}$, w, and the likelihood (see the sub-algorithm ProcessClusters).

In order to denoise the input image, we extract its overlapping patches in a sliding window manner. Next, these patches are denoised independently from each other using Algorithm 1 and returned to their original positions. Since they are overlapped, there are several estimates for each pixel, which are then averaged to produce the final denoised image.

IV. EXPERIMENTS

A. Setup

The standard test images of size 256×256 shown in Fig. 1 were utilized in the experiments. Since the variance equals the expected value for Poisson noise, the peak intensity of the original image defines the noise level in the noisy image. On one hand, it was shown in [12] that the strategy referred to as binning is preferred when the peak intensity is less than 1. According to this approach, the input image is first downsampled by aggregating neighboring pixels, which results in a smaller image with a higher peak intensity. This image is then processed by any Poisson denoising algorithm and upscaled to the original size using interpolation. On the other hand, when
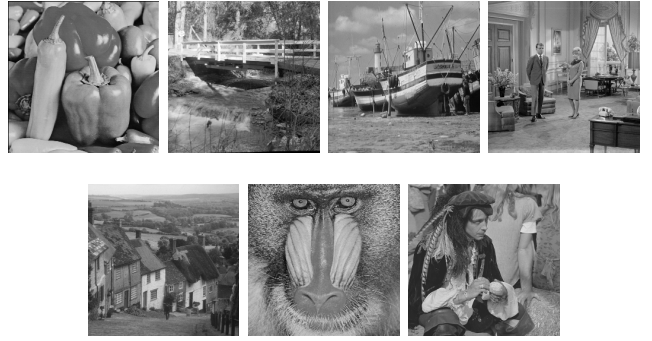

Fig. 1. Test images. From left to right: Peppers, Bridge, Boat, Couple, Hill, Mandril, Pirate.

the peak intensity is higher than 5, the methods based on the variance stabilization are preferable. Therefore, we varied the peak intensity between 1 and 5 in our experiments.

We compared our approach with the SPDA [12] and the BM3D with the exact unbiased inverse Anscombe transformation [4], which are currently the best existing techniques. Their code is available online and was taken with the default parameter settings. The C++ implementation of our algorithm, which was utilized in the experiments, is available at http://physics.medma.uni-heidelberg.de/cms/content/poisson-denoising. All overlapping patches from the BSD500 image set [15] were taken as $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_P)}$, which resulted in a set of $N_P \approx 7 \cdot 10^7$ patches.

B. Choice of Parameters

Increase of the patch size $\sqrt{d} \times \sqrt{d}$ makes it possible to handle more complex image structures but requires a larger number of patches N_P to represent the natural image patch distribution. In our experiments, which were conducted with $N_P \approx 7 \cdot 10^7$, we used patches of size 14×14.

Regarding the number of clusters N_C, the larger it is, the more accurately estimator (9) approximates (7). Nevertheless, N_C cannot be arbitrary large since it should be computationally feasible to construct the k-d trees and the K-nearest neighbor graph. Due to this reason, we set N_C to 10^6.

Increase of the neighbor list size K results in a larger number of clusters, which should be processed at each iteration of the loop in lines 10–14 of Algorithm 1. On the other hand, it reduces the number of loop iterations, which are required to achieve sufficient denoising accuracy, and makes it possible to set smaller M in the convergence criterion. We utilized $K = 2d$ and $M = 10$ in our experiments.

The number of k-d trees N_T and the k-d tree leaf size L do not significantly affect the results and were set to $N_T = 64$ and $L = 32$.

C. Results

The results are presented in Table I. For small peak intensities, the presented method outperforms the existing techniques by about 0.3 dB. As the peak intensity increases, the results of our algorithm become comparable to those of [4].

TABLE I

THE DENOISING RESULTS. EACH VALUE IS THE AVERAGE OF FIVE NOISE REALIZATIONS. THE BEST RESULT IS SELECTED WITH THE BOLD FONT. THE PSNR IS GIVEN IN DECIBELS.

Method	Peak	Image							Average
		Peppers	Bridge	Boat	Couple	Hill	Mandril	Pirate	
SPDA [12]		19.96	19.21	20.07	20.05	20.61	18.40	20.06	19.77
BM3D [4]	1	19.86	19.22	20.01	19.93	20.69	18.14	20.12	19.71
proposed		**20.38**	**19.55**	**20.24**	**20.26**	**20.98**	**18.43**	**20.49**	**20.05**
SPDA [12]		21.20	20.14	21.01	20.97	21.81	18.92	21.22	20.75
BM3D [4]	2	21.94	20.31	21.07	21.00	21.63	18.79	21.28	20.86
proposed		**22.26**	**20.65**	**21.28**	**21.22**	**22.05**	**18.98**	**21.60**	**21.15**
SPDA [12]		21.91	20.43	21.34	21.22	22.13	19.05	21.60	21.10
BM3D [4]	3	23.35	21.00	21.86	21.65	22.48	19.28	22.04	21.67
proposed		**23.37**	**21.21**	**21.93**	**21.81**	**22.78**	**19.35**	**22.29**	**21.82**
SPDA [12]		22.06	20.55	21.51	21.34	22.16	19.11	21.86	21.23
BM3D [4]	4	**24.03**	21.50	**22.36**	**22.26**	23.06	**19.58**	22.61	**22.20**
proposed		23.92	**21.60**	22.32	**22.26**	**23.23**	19.56	**22.79**	22.24
SPDA [12]		22.31	20.60	21.60	21.45	22.22	19.16	22.04	21.34
BM3D [4]	5	**24.67**	21.87	**22.84**	**22.65**	23.51	**19.82**	23.03	**22.63**
proposed		24.40	**21.90**	22.68	22.55	**23.61**	19.72	**23.09**	22.56

It took, on average, 14 minutes for our method implemented in C++ to process one 256×256 image on a PC with CPU Intel i7-5820K 3.3 GHz.

V. DISCUSSION AND CONCLUSION

In this work, we present a new noise suppression method and demonstrate its applicability to Poisson noise removal. Compared with the approaches [12], [11], [4], our algorithm has a different assumption about the original image. Instead of utilizing image self-similarity or sparse patch representation, patches of the original image are assumed to be sampled from the distribution representing natural image patches. As a result, denoised patches are computed using the MMSE estimator, i.e. as the conditional expectation of the original patch given the noisy patch (4). As pointed out in [2], such approach would provide optimal denoising results if the number of patches N_P tended to infinity and the patch size d was increased to the image size. Nevertheless, it is useful also with fixed N_P and d. As the experiments showed, it is the preferable method for peak intensities below 4, for which the variance stabilization is not efficient enough.

Since a large number of patches is necessary to represent the image patch distribution, the choice of suitable data structures is of primary importance in order to calculate the MMSE estimator with a reasonable computational effort and make it practically applicable. In this work, it is demonstrated that the k-d tree search followed by the search in the K-nearest neighbor graph allows fast and accurate calculation of the MMSE estimator.

REFERENCES

[1] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2833–2840.

[2] A. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 73–86.

[3] J. Boulanger, C. Kervrann, P. Bouthemy, P. Elbau, J.-B. Sibarita, and J. Salamero, "Patch-based nonlocal functional for denoising fluorescence microscopy image sequences," *Medical Imaging, IEEE Transactions on*, vol. 29, no. 2, pp. 442–454, 2010.

[4] M. Mäkitalo and A. Foi, "Optimal inversion of the anscombe transformation in low-count poisson image denoising," *Image Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 99–109, 2011.

[5] B. Zhang, J. M. Fadili, and J.-L. Starck, "Wavelets, ridgelets, and curvelets for poisson noise removal," *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1093–1108, 2008.

[6] F. J. Anscombe, "The transformation of poisson, binomial and negative-binomial data," *Biometrika*, pp. 246–254, 1948.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

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

[9] H. C. Burger, C. Schuler, and S. Harmeling, "Learning how to combine internal and external denoising methods," in *Pattern Recognition*. Springer, 2013, pp. 121–130.

[10] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Processing*, vol. 89, no. 12, pp. 2609–2629, 2009.

[11] J. Salmon, Z. Harmany, C.-A. Deledalle, and R. Willett, "Poisson noise reduction with non-local PCA," *Journal of mathematical imaging and vision*, vol. 48, no. 2, pp. 279–294, 2014.

[12] R. Giryes and M. Elad, "Sparsity-based poisson denoising with dictionary learning," *Image Processing, IEEE Transactions on*, vol. 23, no. 12, pp. 5057–5069, 2014.

[13] C. Lee, C. Lee, and C.-S. Kim, "An mmse approach to nonlocal image denoising: Theory and practical implementation," *Journal of Visual Communication and Image Representation*, vol. 23, no. 3, pp. 476–490, 2012.

[14] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 11, pp. 2227–2240, 2014.

[15] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.