Feature-Preserving Noise Removal

Khalid Youssef, *Graduate Student Member, IEEE,* Nanette N. Jarenwattananon and Louis-S. Bouchard

***Abstract*—Conventional image restoration algorithms use transform-domain filters, which separate the noise from the sparse signal among the transform components or apply spatial smoothing filters in real space whose design relies on prior assumptions about the noise statistics. These filters also reduce the information content of the image by suppressing spatial frequencies or by recognizing only a limited set of shapes. Herein we demonstrate efficient denoising using a nonlinear filter that operates along patch neighborhoods and multiple copies of the original image. The use of patches enables the algorithm to account for spatial correlations in the random field whereas the multiple copies are used to recognize the noise statistics. The nonlinear filter, which is implemented by a hierarchical multi-stage system of multilayer perceptrons, outperforms state-of-the-art denoising algorithms such as those based on collaborative filtering and total variation. Compared to conventional denoising algorithms, our filter can restore images without blurring them, making it attractive for use in medical imaging where the preservation of anatomical details is critical.**

***Index Terms*—multilayer perceptrons, image restoration, image denoising, magnetic resonance imaging, multiple copies.**

I. Introduction

THE process of removing noise in an image requires *a priori* knowledge of the noise distribution. In Wiener filtering, for example, the optimal filter requires knowledge of the power spectra of signal and noise [1]. It is common practice to perform denoising in some transform domain where the signal is sparse and discard the transform coefficients that do not overlap with those containing the signal [2], [3], [4]. Multi-resolution techniques such as the wavelet transform can achieve better sparsity and separation of the noise depending on the type of basis element that are used and the over-completeness of the basis [5], [6]. A recent method, Block-matching and 3D filtering (BM3D), also termed "collaborative filtering" has shown state-of-the-art performance by grouping patches that look similar into 3D blocks and performing transform domain filtering of each 3D similarity block [7]. These algorithms yield outstanding performance as long as the images obey certain conditions consistent with algorithm assumptions, such as noise statistics or the type of patterns contained in the image. If the assumptions are not met, this can give rise to artifacts or losses in the image fine structure [8].

From an experimental standpoint, the simplest method to reduce noise is signal averaging. In the case of additive white Gaussian noise (AWGN) signal averaging can be performed until the mean value of the signal is sufficiently high relative to the noise. In magnetic resonance imaging (MRI), signal transients from each phase-encode step are acquired and averaged to produce a mean-value image with higher signal-to-noise ratio (SNR). In photography, the analogous process consists of increasing the exposure time. The "mean-value" image in the case of AWGN generally reveals better contrast-to-noise. In the general case, there are three potential problems with the signal-averaging approach. First, there is an implicit assumption about the noise statistics. The mean value is the best estimate of the signal only in special cases. For example, signal averaging fails to remove multiplicative noise. Second, averaging does not account for spatial correlations in the image. Third, in the process of averaging data from multiple transients into single values (the mean values), much information is discarded about the noise, whose true nature can only be revealed upon realizations of the random experiment.

In this paper, we propose to denoise each pixel of an image using a nonlinear filter that operates along a data block which consists of patch neighborhoods of a pixel and multiple copies of the same image. Thus, an image pixel is denoised using information from one block of size $D = r \times N_p$, where N_p is the number of pixels in a patch and r is the number of copies. In this study, we use rectangular patches of size $N_p = 17 \times 17$ and $r = 7$ copies of the image. The nonlinear filter is designed based on multilayer perceptrons (MLP), which have been shown to be universal function approximators [9]. The purpose of operating along patches is to account for possible spatial correlations in the random field of the image. The method makes no assumptions about the noise statistics. There is also no guesswork involved in determining suitable thresholds, parameters or dictionaries. Because every algorithm deserves an acronym, we propose to call it multiple copies-multilayer perceptrons (MC-MLP).

Our test case will be MRI images, due to the common practice in MRI to perform signal averaging as a way to improve SNR. Poor SNR in MRI may arise in low magnetic fields or during the detection of metabolites or insensitive nuclei such as ^{23}Na. Such an MRI experiment can be readily configured to save individual transients separately to provide multiple copies of the same image. Magnitude-mode MRI images typically feature Rician noise. We validate MC-MLP by comparing its performance to the total variation method (TV) [10], [11], and a Rician noise MRI implementation of the state-of-the-art denoising algorithm Block-matching and 4D filtering (BM4D) [12], which performs collaborative filtering. MC-MLP outperforms both methods in terms of conventional methods such as peak signal-to-noise ratio (PSNR), feature similarity (FSIM), and mean structural similarity (MSSIM), as well as subjective visual quality metrics [13], [14]. We also demonstrate excellent performance in the removal of multi-

K. Youssef and L.-S. Bouchard are with the Department of Bioengineering, University of California, Los Angeles, 410 Westwood Plaza, Los Angeles, CA 90095 USA

N.N. Jarenwattananon and L.-S. Bouchard are with the Department of Chemistry and Biochemistry, University of California, Los Angeles, 607 Charles Young Drive East, Los Angeles, CA, 90095 USA

plicative noise, illustrating the independence of the method to the type of noise.

II. Denoising by Multiple Copies

A. Nonlinear Filter Design With MLP

An image Y is a mapping

$$Y : \{1, \ldots, N_x\} \times \{1, \ldots, N_y\} \to \mathcal{S}, \tag{1}$$

where $\mathcal{S}$ is a set of allowed pixel values. The Cartesian product $\{1, \ldots, N_x\} \times \{1, \ldots, N_y\}$ will be indexed by $t = (i, j)$. An experimentally measured image contains noise and is therefore a random field X whose realization is denoted by $X(\omega)$. It is convenient to denote $X_t(\omega)$ the value of the t-th pixel in the image $X(\omega)$ (a matrix of size $N_x \times N_y$). The probability space is $(\Omega, \mathcal{F}, P)$, where $\omega \in \Omega$ and Ω is the set of all possible outcomes

$$\Omega = \{I_1, I_2, \ldots, I_{N_I}\}, \quad N_I = (\#\mathcal{S})^{N_x \times N_y} \tag{2}$$

where I_j ($j = 1, \ldots, N_I$) is a $N_x \times N_y$ matrix whose elements belong to the set $\mathcal{S}$ and $I_i \neq I_j$ for $i \neq j$. There is no fundamental restriction on $\mathcal{S}$. For example, if $\mathcal{S}$ is a discrete set such as $\mathcal{S} = \{0, 1, \ldots, 255\}$, then the σ-algebra on Ω can be taken to be the power set $\mathcal{F} = 2^{\Omega}$ whereas if $\mathcal{S}$ is a continuous interval such as $[0, 255]$ then Ω can be taken to be the Borel σ-algebra generated by the interval.

P is a probability measure depending on the nature of the experiment and could be unknown or arbitrary. Its structure can be inferred from individual realizations $\omega_1, \omega_2, \ldots$. For fixed $\omega \in \Omega$, the mapping $X_t(\omega) \to \mathcal{S}$ as a function of t yields a realization (sample path) of the random field. The sample path is an image in the sense of the mapping (1).

Denoising by multiple copies is the task of finding an estimator $\tilde{Y}$ of the true image Y given the prior information from r realizations of the sample path $\{X(\omega_1), X(\omega_2), \ldots, X(\omega_r)\}$ where $X(\omega) = \eta(\omega, Y)$, such that $\tilde{Y} \approx Y$ according to a suitable distance metric. Here, $\eta(\omega, \cdot)$ stands for the noise function, which is determined by the probability measure, P. An example of $\eta(\omega, \cdot)$ is AWGN, which takes the form, $X(\omega) = Y + \Gamma(\omega)$, where $\Gamma(\omega)$ is a $N_x \times N_y$ matrix of random values that are Gaussian-distributed.

To obtain a good estimate $\tilde{Y} \approx Y$, we must reduce the uncertainty of the estimate. There are at least two ways to do this. The first is to look at more instances of X, for example, by having several copies of the same image, i.e. $\{X(\omega_1), X(\omega_2), \ldots, X(\omega_r)\}$ where r is sufficiently large. Since $X(\omega)$ varies with each instance ω according to the noise distribution whereas Y is independent of ω, the more instances ω we have, the more certain we can be about the value of our estimate. Although in practice, this increases computational cost, we found that $r = 7$ is a good compromise.

The second approach to obtain more information is to look at the neighborhood of the pixel $X_t(\omega)$. We will denote the coordinates of the neighborhood by the set of points in a square region U_t centered on $t = (i, j)$:

$$U_t = \{t' = (i', j') | i' \in [i - d, i + d], j' \in [j - d, j + d]\} \tag{3}$$

U_t is referred to as a patch in the text below, where d is the number of pixels included away from the pixel's coordinates $t = (i, j)$. This patch region U_t contains $N_p = (2d + 1)^2$ pixels. When denoising images it is important to account for spatial correlations in the random field $X(\omega)$ due to the shape of the deterministic function Y, or possible spatial correlations in the noise function (if any). Modern denoising algorithms operate on patches on the premise that images contain specific shapes and patterns such as curves, edges and plain surfaces. Denoising algorithms often take the mean of all available copies and recognize one or more type of low level patterns in patches such as edges. Yet a general searching algorithm can learn higher level patterns and can increase certainty by taking separate copies into consideration rather than just their mean.

The nonlinear filter design utilizes the information from r copies and a patch U_t centered on the pixel X_t with neighborhood distance d and finds a function

$$\exists f(X(\omega_1)|_{U_t}, X(\omega_2)|_{U_t}, \ldots, X(\omega_r)|_{U_t}) = \tilde{Y}_t^{\circ} \tag{4}$$

where $X(\omega_1)|_{U_t}$ denotes the restriction of the matrix $X(\omega_1)$ to the U_t neighborhood. It is a $(2d+1) \times (2d+1)$-dimensional matrix with entries taking values in $\mathcal{S}$. $\tilde{Y}^{\circ}$ is the best estimate of Y that can be obtained from the information provided by all r copies $X(\omega_1)|_{U_t}, X(\omega_2)|_{U_t}, \ldots, X(\omega_r)|_{U_t}$ of the U_t neighborhood, for all such neighborhoods ($\forall t$). The r two-dimensional matrices $X(\omega_1)|_{U_t}, X(\omega_2)|_{U_t}, \ldots, X(\omega_r)|_{U_t}$ are reshaped into 1D vectors of length N_p, then concatenated into a 1D vector of length $D = r \times N_p$ denoted by $\vec{x}_t$ and used inputs to a MLP whose transfer function is a hyperbolic tangent. Thus, a MLP with D inputs, K outputs, one hidden layer with M nodes yields a K-dimensional output vector $\vec{\tilde{y}}_t$ whose k-th component is given by the iterated hyperbolic tangents:

$$\tilde{y}_t(k) = \tanh\left(\sum_{l=0}^{M} \theta_{l,k}^{(2)} \tanh\left(\sum_{j=0}^{D} \theta_{j,l}^{(1)} x_t(j)\right)\right) \tag{5}$$

where $z(l) = \tanh\left(\sum_{j=0}^{D} \theta_{j,l}^{(1)} x_t(j)\right)$, for $l = 0, \ldots, M$ are the outputs of the hidden layer. We use the convention where $x_t(0) = 1$ and $z(0) = 1$, so that $\theta_{0,l}^{(1)}$ and $\theta_{0,k}^{(2)}$ represent biases to the transfer function. The generalization to arbitrary numbers of hidden layers is straightforward by nesting additional hyperbolic tangents. The calculation of the vector $\vec{\tilde{y}}$ is called feed forward propagation. In our implementation, each MLP has a single output corresponding to a single pixel in the image. Thus, $K = 1$ and we may drop the vector notation, writing $\tilde{y}_t$ instead of $\vec{\tilde{y}}_t$.

Let $\vec{\Theta} = \left[\theta_{j,l}^{(1)}|_{j=0..D}^{l=0..M}, \theta_{l,k}^{(2)}|_{l=0..M}^{k=0..K}\right]$ be a vector of length m containing weights and bias values for all the nodes. The MLP is trained to solve for f by searching for an optimal $\vec{\Theta}$ that minimizes the sum of square errors in (6)

$$E(\vec{\Theta}) = \frac{1}{2} \sum_{s=1}^{S} e_s(\vec{\Theta})^2, \tag{6}$$

where $e_s(\vec{\Theta}) = y_s - \tilde{y}_s$ is the MLP error corresponding to sample s for a given set of MLP parameters. Here, y_s is the desired target value from a low noise image for an input vector $\vec{x}_s$ from a noisy training sample, and $\tilde{y}_s$ is the MLP estimate of y_s. While the coordinate t is a suitable index in the feed forward phase where every pixel in the image is processed, we denote it by s in the training phase where an error is calculated. Training samples are input-output pairs $(\vec{x}_s, y_s)$ picked from training images in no specific order where the entire image or only parts of the image might be used for training. s corresponds to the sample number in the training dataset. S is the total number of training samples.

The algorithm minimizes the errors at all nodes. Back-propagation uses the output error in (6) to determine errors of individual nodes in the remaining layers [15]. From $n(k) = \sum_{l=0}^{M} \theta_{l,k}^{(2)} \tanh\left(\sum_{j=0}^{D} \theta_{j,l}^{(1)} x(j)\right)$, the partial error for a weight $\theta_{l,k}^{(2)}$ is obtained by

$$\frac{\partial E(\vec{\Theta})}{\partial \theta_{l,k}^{(2)}} = \frac{\partial E(\vec{\Theta})}{\partial n(k)} z(l). \tag{7}$$

The error at a node is determined by taking into account the sum of partial errors of weights for all connections emanating from it. $\vec{\Theta}$ is iteratively updated using Levenberg-Marquardt search [16]. $\Delta\vec{\Theta}$ is calculated at each iteration using the update rule,

$$\Delta\vec{\Theta} = -[J^T J + \mu \mathbf{1}]^{-1} J^T \vec{e}, \tag{8}$$

and is added to $\vec{\Theta}$. $\vec{e} = (e_1, e_2, \ldots, e_S)$ is a vector of MLP errors for all samples.

Here, $\mathbf{1}$ is an identity matrix and

$$J = \begin{bmatrix} \frac{\partial e_1(\vec{\Theta})}{\partial \Theta_1} & \frac{\partial e_1(\vec{\Theta})}{\partial \Theta_2} & \cdots & \frac{\partial e_1(\vec{\Theta})}{\partial \Theta_m} \\ \frac{\partial e_2(\vec{\Theta})}{\partial \Theta_1} & \frac{\partial e_2(\vec{\Theta})}{\partial \Theta_2} & \cdots & \frac{\partial e_2(\vec{\Theta})}{\partial \Theta_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_S(\vec{\Theta})}{\partial \Theta_1} & \frac{\partial e_S(\vec{\Theta})}{\partial \Theta_2} & \cdots & \frac{\partial e_S(\vec{\Theta})}{\partial \Theta_m} \end{bmatrix} \tag{9}$$

is the Jacobian matrix containing first derivatives of MLP errors with respect to the $\vec{\Theta}$ parameters. When μ is large the method behaves like a steepest descent method. When μ is small the method is equivalent to a Gauss-Newton method. μ is updated at each iteration depending on how E changes.

While denoising by MLP has been shown to be possible given enough layers, nodes and training samples, several challenges exist in practice to make it computationally feasible. Burger and co-workers [17] studied the use of MLPs as a pure learning denoising approach by training a relatively large MLP to denoise images corrupted by AWGN. Their MLP did not use multiple copies. It was trained to denoise patches of size 17×17 stitched together after denoising to reconstruct the final image. Their MLP contained four hidden layers with 2,047 nodes each, and was trained with 362 million samples derived from a database of 1,500 images and requires approximately one month of training time on a GPU for a specific noise level. While their MLP could compete with BM3D, it remains impractical in real applications due to its exceedingly high computational cost and lengthy execution times.

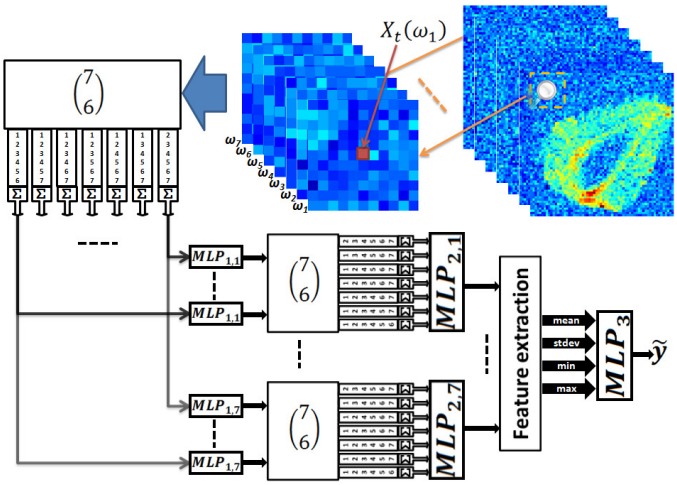

Fig. 1. System design for the MC-MLP denoising algorithm as implemented by multiple stages.

Here we use an entirely different approach where the system's architecture (Fig. 1) uses multiple stages of MLP. We divide our system into two phases, a training phase and a feed forward phase. The training phase is where the MLPs learn to build an optimized model for the application at hand. This is where the nonlinear filter is designed. This phase can take anywhere from 15 minutes to several hours on a modern laptop computer, depending on the noise level. Once the training phase is complete, the MLPs operate in feed forward mode. This is the phase where the nonlinear filter is applied to the image data. The feed forward phase is much faster than the training phase. The time required to denoise an image is on the order of several seconds to a few minutes, depending on the size of the image.

B. Training Phase

1) Multiple Stages: The first step to reducing computational cost is using several small MLPs trained in multiple stages instead of one large MLP. While the MLP architecture can be optimized to further enhance performance, architecture optimization is a topic on its own and will not be considered here. The MLPs used in our tests feature 6 hidden layers each with 10 nodes per layer. Performance is better with more training samples, but larger datasets require more nodes, increasing MLP size and computation time. The lower the noise level, the less training is required. Thus, we do training in a first stage of small MLPs with a relatively small dataset to minimize noise to a high degree, albeit not state-of-the-art level. When training is done, first stage MLPs operate in feed forward mode and are used to denoise original training images. The end result is a set of estimates with arbitrary residual errors, yielding arbitrary noise distributions with much smaller standard deviation. Seven first stage MLPs are shown in the diagram and yield seven estimates for each training image with much lower noise than original copies. Estimates are used to generate a new dataset for training the second denoising stage

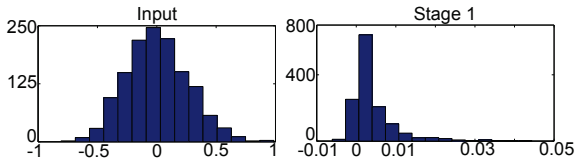

Fig. 2. Example of the evolution of the noise distribution after a denoising stage. Here (left) the Gaussian input noise distribution is very different from the (right) noise distribution after the first denoising stage.

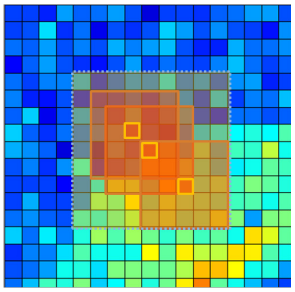

Fig. 3. Multiple stage training allows for using smaller patches at each stage. Here, a 17×17 patch is subdivided into smaller patches which are sent to the multiple stages of denoising for processing.

MLPs. Multiple MLP stages can be added in a similar manner and trained hierarchically. A third denoising stage uses dataset generated from estimates of the second denoising stage MLPs and so on. In our tests, four MLP stages were used. For clarity, only two MLP denoising stages are shown in Fig. 1. The last stage in the diagram is a little different and will be discussed separately.

This multistage denoising approach is extremely powerful and is a major driving force behind the performance of our algorithm. Multistage denoising is, unfortunately, not possible with other denoising methods. This is because other methods make assumptions about the noise statistics whereas the MLP approach is noise independent. For example, an algorithm that is designed for use with Rician noise cannot be used for multistage denoising because its output does not necessarily have a Rician noise distribution.

2) Multiple Copies: Before patches from r noisy copies are introduced to the input layers of the first denoising stage MLPs, they are first grouped into r combinations of $(r-1)$ copies. This produces r distinct realizations of an image random field, but with reduced noise levels. This is denoted by the $\binom{7}{6}$ block in the diagram, where 7 combinations of 6 copies are grouped and added together. This reduces noise levels at the input to enable shorter training times. This operation is made possible because the MLP denoising process is noise independent. This technique is also not applicable in general to other methods where assumptions are made about noise distribution. For example, adding images with Rician noise produces an image with non-Rician noise, violating the basic assumption of the algorithm.

3) Patch Size: Using $d = 8$ for (3) produces patches of size 17×17. When 7 copies are used this yields an input vector of length 2023. Multistage training allows using smaller patches per stage while still allowing the system to use information from a large patch size. This concept is illustrated in Fig. 3. For $d = 2$ for first stage MLPs, each output represents a center pixel from patches of size 5×5 from original noisy copies. Using $d = 2$ for second stage, each output represents a center pixel from the 5×5 patches from the first denoising stage MLPs estimates. This collectively gives an effective patch size of 9×9 from original noisy copies to be used as inputs for the second denoising stage MLPs. In general, the effective d value for a stage is the sum of individual d values from previous stages. This technique reduces processing and memory requirements, making our method applicable to devices with low computational resources. Using a smaller d value in a consecutive stage yields a reduction in dimensionality, giving the option of optimizing for speed or memory. Larger d values are typically assigned for the first training stage where smaller data sets can be used. Decreasing d in subsequent stages allows for larger data sets.

4) Feature Extraction: In our discussion so far, each stage includes multiple MLPs. Ultimately, we need one final value for each pixel. One way to do this is by averaging values of all estimates from the final stage MLPs. Alternatively, an additional stage can be added with one MLP and $d = 0$ to get a final value. However, instead of using raw MLP outputs to train this additional (now final) stage, we use feature extraction to enhance generalizing ability. This is indicated by the feature extraction block where the mean, standard deviation, minimum and maximum of outputs from stage 2 are used to train the MLPs in stage 3 to get a final result.

C. Feed Forward Phase

After completing MLP training for all stages, the system is used in feed forward mode where image denoising is performed. In our case, we require 7 noisy copies of an image to produce a clean estimate. Patches are extracted for each pixel from its surrounding neighbors for all 7 copies, producing 7 patches of size $N_p = (2d+1)^2$. Denoising is performed hierarchically. While MLPs in each stage are independent and can be processed in parallel, the performance at each stage depends on results from preceding stage. Pixels estimates from the first stage are reorganized into their corresponding positions in the image. The same data acquisition process is performed on the first stage image estimates using d values of the second stage. The final stage produces one final estimate for each pixel. The estimates are regrouped to produce a final estimate of the denoised image. The total time for denoising an image depends on its size. The computer used in our work is a laptop equipped with a 4-core Intel® Core™ i7-3610QM CPU @ 2.30 GHz per core. The average time required for denoising of an 128×128 image was approximately 15 s. Time grows linearly with the number of pixels, i.e. a 256×256 image requires approximately 15 s $\times$ 4 = 60 s.

III. Results

Results obtained for various noise levels and distributions indicate that our method can outperform the current state-of-the-art denoising methods provided the algorithm is given enough training time and samples, under the condition of

reasonable training time. The longest training time we encountered was less than 10 hours. Because our method allows optimizing performance for specific applications while maintaining good generalizing ability, datasets need only be on the order of hundreds of thousands training samples. We first perform a time comparison to find out how much training time it takes our system to achieve comparable results with the benchmark MLP from [17] and BM3D. We then apply our method to denoising MRI images with Rician noise and compare our results under different noise levels with BM4D and TV. We also apply our method to denoising images under arbitrary noise including multiplicative and additive components for different noise levels. Finally, we apply our method to real MRI data and compare our results to an array of MRI denoising algorithms. In order to keep comparisons consistent, we use implementation demos provided by each group for their algorithm on their website. Links for each method can be found in the reference section. Note that as our approache is learning-based, noise estimation is not explicitly required by our method. In practical applications, learning-based denoising approaches implicitly learn the noise distribution for the specific application from training data and do not require a separate noise estimation step. When a learning-based approach is used for general denoising, a database for different noise levels is created and noise estimation methods are considered at that point, however this is outside the scope of this paper.

A. Time Comparison With Benchmark MLP and BM3D

A demo for the BM3D method is provided by [7] which can be downloaded at the URL [18]. The demo provided by [17] contains weights for an MLP that was trained for 1 month on a GPU system to denoise images with pixel values in the range $[0, 255]$ contaminated by AWGN of standard deviation 25 (in units of the pixel range). The demo for the MLP method can be downloaded at the URL [19]. We applied our MC-MLP method to denoise an MRI image after adding AWGN with equivalent standard deviation to the one provided by the MLP method of [17] and measured the time it takes to train our system to achieve comparable results. Given that our method uses 7 copies, we use a standard deviation of $25\sqrt{7}$ for the AWGN introduced to each copy as shown in (10),

$$X_t(\omega) = Y_t + 25\sqrt{7}\Gamma_t(\omega), \tag{10}$$

where the probability measure P is defined as follows: Γ_t is a zero-mean Gaussian random field which is spatially uncorrelated in the sense that Γ_t is independent of Γ_u whenever $t \neq u$. Thus, the spatial correlations in X_t, as seen by the denoising algorithm, are due to the signal Y_t only.

We generate 7 copies $X_t(\omega_1)$, $X_t(\omega_2)$, ..., $X_t(\omega_7)$ with a standard deviation of $25\sqrt{7}$ and use them as inputs to the MC-MLP algorithm. The individual copies are averaged to generate one copy with a standard deviation of 25 to be denoised by the other algorithms. Our method outperformed the MLP from [17] after only 1 hour of training on a standard laptop computer, as compared to 1 month of training in their case.

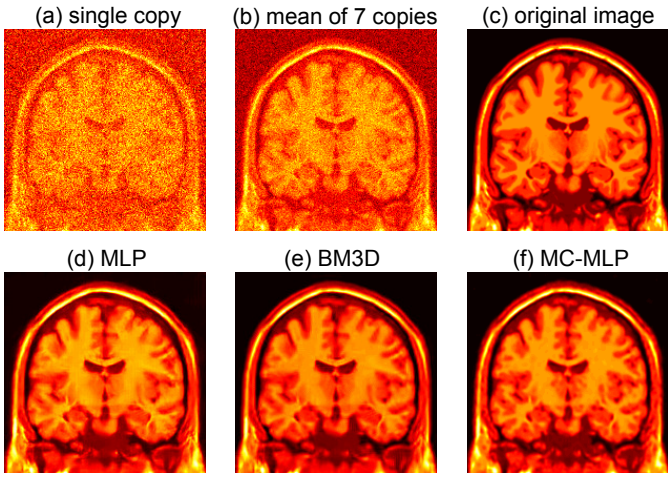

Fig. 4. Comparison of MC-MLP with MLP and BM3D after 10 hours of training at noise level $\sigma = 25$. The goal here is to demonstrate computational efficiency of MC-MLP as compared to MLP, by measuring the training time required to obtain similar performance as BM3D. (a) Noisy image (1 copy). PSNR: 11.723 dB. FSIM: 0.450. MSSIM 0.135. (b) The average of 7 copies. PSNR 20.242 dB. FSIM: 0.687. MSSIM 0.425. (c) Original (ideal) image with high SNR used as the gold standard for comparison. (d) MLP method applied to average of the 7 copies. PSNR 30.615 dB. FSIM: 0.922. MSSIM 0.869. (e) BM3D method applied to average of the 7 copies. PSNR 31.228 dB. FSIM:0.933. MSSIM 0.908. (f) Denoising using the MC-MLP method with 7 copies. PSNR 31.242 dB. FSIM:0.936. MSSIM 0.919.

Our method also outperformed BM3D after only 10 hours of training. The results are shown in Fig. 4.

B. Rician Denoising Comparison With BM4D and TV

BM3D is adapted to Rician noise by using a transform to map it to AWGN and then perform AWGN denoising. The Rician noise implementation was applied to volumetric MRI data. This extension to volumetric data is called BM4D. Instead of grouping similar 2D patches together in 3D blocks, BM4D groups similar 3D patches in 4D blocks. A demo with a dataset of volumetric MRI brain image is provided at the URL [20]. A minimum of 9 slices is required for the demo to work. We apply our method to the same problem and compare its performance on the same dataset used by the BM4D demo for different noise levels by changing the value of σ in the Rician noise distribution added to images. Images with Rician noise are generally defined as shown in (11),

$$X_t(\omega) = \sqrt{(Y_{tR} + \sigma\Gamma_t^{(1)}(\omega))^2 + (Y_{tI} + \sigma\Gamma_t^{(2)}(\omega))^2}. \tag{11}$$

where Y_{tR} and Y_{tI} are the real and imaginary signal components respectively. P is defined as follows: $\Gamma_t^{(1)}$ and $\Gamma_t^{(2)}$ are zero-mean Gaussian random fields which are statistically independent of each other and spatially uncorrelated in the sense that $\Gamma_t^{(i)}$, $i = 1, 2$ is independent of $\Gamma_u^{(i)}$ whenever $t \neq u$. Thus, the spatial correlations in X_t are due to the signal Y_t only.

We generated 7 copies for each noise level to be used by our method. Since BM4D requires 9 slices, we generate 7 noisy copies for 9 adjacent slices to be denoised by BM4D. Only the last slice is denoised by our method and compared

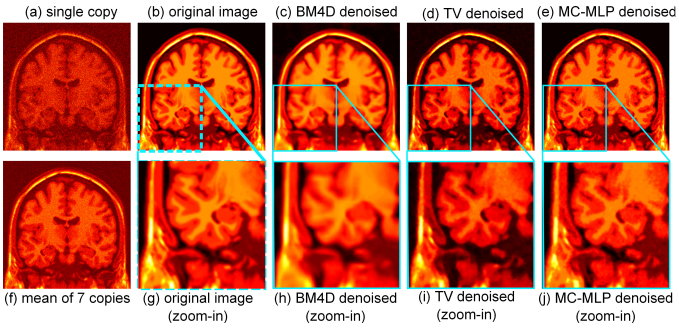

Fig. 5. Comparison of MC-MLP with BM4D and TV methods ($\sigma = 20$). (a) Noisy image (1 copy). PSNR 13.378 dB. FSIM 0.658. MSSIM 0.073. (b) Original (ideal) image with high SNR used as the gold standard for comparison. (c) BM4D method applied to each of the 7 copies. The average of 7 denoised copies is shown. PSNR 28.789 dB. FSIM 0.894. MSSIM 0.804. (d) TV method applied to each of the 7 copies. The average of 7 denoised copies is shown. PSNR 21.793 dB. FSIM 0.896. MSSIM 0.703. (e) Denoising using the MC-MLP method with 7 copies. PSNR 33.314 dB. FSIM 0.951. MSSIM 0.907. (f) Signal-averaged image corresponding to the mean of 7 copies. PSNR 16.718 dB. FSIM 0.832. MSSIM 0.273. (g) Close-up on the lower left quadrant of (b) (h) Close-up on the lower left quadrant of (c). (i) Close-up on the lower left quadrant of (d). (j) Close-up on the lower left quadrant of (e).

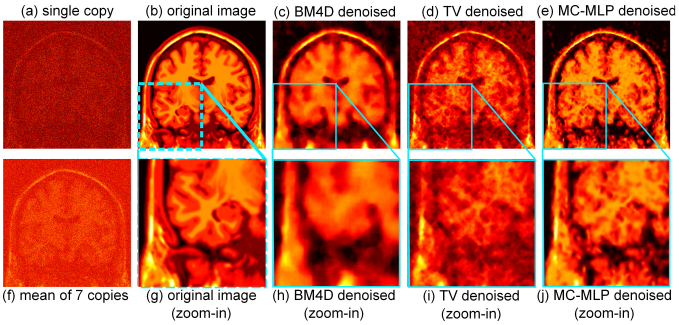

Fig. 6. Comparison of MC-MLP with BM4D and TV methods ($\sigma = 70$). (a) Noisy image (1 copy). PSNR 1.863 dB. FSIM 0.373. MSSIM 0.005. (b) Original (ideal) image with high SNR used as the gold standard for comparison. (c) BM4D method applied to each of the 7 copies. The average of 7 denoised copies is shown. PSNR 20.658 dB. FSIM 0.743. MSSIM 0.594. (d) TV method applied to each of the 7 copies. The average of 7 denoised copies is shown. PSNR 18.647 dB. FSIM 0.708. MSSIM 0.486. (e) Denoising using the MC-MLP method with 7 copies. PSNR 22.678 dB. FSIM 0.806. MSSIM 0.652. (f) Signal-averaged image corresponding to the mean of 7 copies. PSNR 3.523 dB. FSIM 0.560. MSSIM 0.030. (g) Close-up on the lower left quadrant of (b) (h) Close-up on the lower left quadrant of (c). (i) Close-up on the lower left quadrant of (d). (j) Close-up on the lower left quadrant of (e).

to BM4D. This provides the BM4D implementation 9 times more data than our method. To keep the comparison as fair as possible, we resized slices used for BM4D by a factor of 3×3, yielding the same amount of data for each method. Unlike the case with AWGN, taking the mean of copies with Rician noise distribution does not produce a Rician noise distribution. Instead, we performed BM4D separately on each copy and took the average of all the outcomes.

For further validation, we also compared our method to a TV implementation for Rician noise provided by Center of Domain Specific Computing (CSDC) at UCLA [21]. Figures 5 and 6 show results for low and extreme noise levels, respectively. Results from all noise levels are summarized

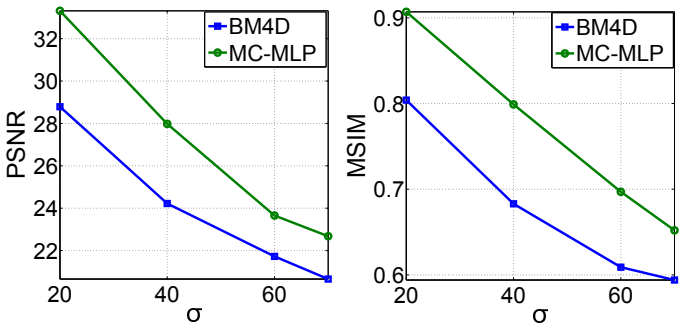

Fig. 7. PSNR and MSIM comparison of MC-MLP with BM4D method for different noise levels ($\sigma = 20, 40, 60$ and 70).

in Fig. 7. According to all three metrics used here, peak signal to noise ratio (PSNR), feature similarity (FSIM) and mean structural similarity (MSSIM), our method displayed superior performance across all noise levels. But aside from these metrics, the best assessment is the ability to identify specific anatomical features. It is clear by comparing (say) Fig. 6(j) to Fig. 6(h) or Fig. 6(i) that our method preserves the anatomical features whereas competing algorithms merely smooth and blur out important details. In particular, the gray to white matter contrast remains clearly defined with our denoising method. From a clinical point of view, this property is most important. Our method is able to depict much of the layering cortex whereas information is lost in the images from competing algorithms [Fig. 6(h,i)] compared to the original image [Fig. 6(g)], for example, in the dark signal regions in the inner surface of the sulcus. These results demonstrate the ability of our method to capture finer features than conventional denoising algorithms are able to.

C. Noise With a Multiplicative Component

In this test, we demonstrate the ability of our method to perform in the presence of multiplicative noise. We applied our method to denoising MRI images of a cherry tomato contaminated by the noise distribution of (12),

$$X_t(\omega) = \sigma_1 \Gamma_t^{(1)}(\omega)(Y_t + \sigma_2 \Gamma_t^{(2)}(\omega)), \qquad (12)$$

where P is defined as follows: $\Gamma_t^{(1)}$ and $\Gamma_t^{(2)}$ are zero-mean Gaussian random fields which are statistically independent of each other and spatially uncorrelated in the sense that $\Gamma_t^{(i)}$, $i = 1, 2$ is independent of $\Gamma_u^{(i)}$ whenever $t \neq u$. Thus, the spatial correlations in X_t are due to the signal Y_t only.

We tested our method with different noise levels by varying the value of σ_2. Since we are not aware of other methods designed for this type of noise, we compared the results of our system to the mean value of the noisy copies. The latter approach is the most commonly used method for reducing noise in experiments. The results are shown in Fig. 8. By comparing Fig. 8(c) to Fig. 8(b) [relative to the reference image, Fig. 8(g)], it is clear that the method does a very good job at removing the noise even under conditions of extreme noise.

TABLE I
PSNR, FSIM, AND MSSIM VALUES CORRESPONDING TO EACH IMAGE IN FIG. 9.

	1 Copy	Mean	MC-MLP	BM4D	TV	ORNLM	AONLM	ONLM	ODCT	PRINLM
Noise Level 1 - PSNR	17.95,	20.61,	26.13,	22.56,	25.82,	25.78,	23.07,	25.71,	23.13,	24.01,
FSIM	0.72,	0.87,	0.93,	0.92,	0.92,	0.90,	0.90,	0.90,	0.89,	0.92,
MSSIM	0.09	0.33	0.81	0.79	0.78	0.70	0.73	0.71	0.71	0.78
Noise Level 2 - PSNR	14.02,	16.46,	21.71,	19.50,	19.48,	20.77,	19.25,	21.15,	19.17,	20.30,
FSIM	0.57,	0.72,	0.85,	0.85,	0.79,	0.76,	0.76,	0.76,	0.78,	0.85,
MSSIM	0.04	0.09	0.70	0.58	0.42	0.41	0.37	0.44	0.47	0.52
Noise Level 3 - PSNR	12.95,	13.85,	19.39,	17.81,	16.73,	18.33,	16.54,	18.28,	16.54,	17.06,
FSIM	0.53,	0.58,	0.78,	0.76,	0.60,	0.76,	0.54,	0.54,	0.63,	0.78,
MSSIM	0.03	0.04	0.60	0.33	0.12	0.31	0.11	0.18	0.26	0.33

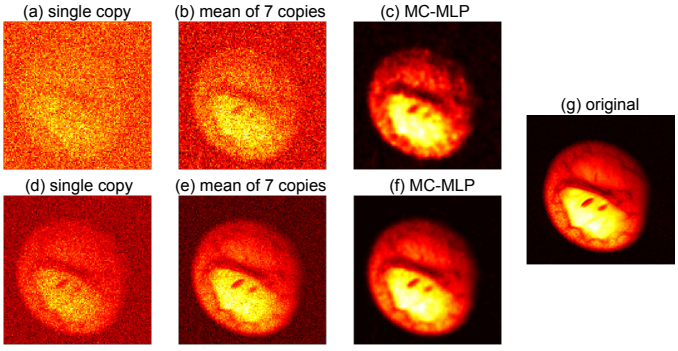

Fig. 8. Comparison of MC-MLP with mean for arbitrary noise including multiplicative component. (a) Noisy image (1 copy) $\sigma_2 = 50$. (b) The average of 7 copies ($\sigma_2 = 50$). PSNR 13.82 dB. FSIM 0.37. (c) Denoising using the MC-MLP method with 7 copies $\sigma_2 = 50$. PSNR 26.88 dB. FSIM 0.87. (d) Noisy image (1 copy) $\sigma_2 = 10$. (e) The average of 7 copies ($\sigma_2 = 10$). PSNR 26.52 dB. FSIM 0.79. (f) Denoising using the MC-MLP method with 7 copies $\sigma_2 = 10$. PSNR 33.77 dB. FSIM 0.95. (g) Original (ideal) image with high SNR used as the gold standard for comparison.

D. Application to MRI Images With Real Noise

So far we have evaluated the performance of the algorithm using MRI images that were contaminated by noise, as is common practice when evaluating novel denoising algorithms. This enabled us to select the noise distribution and type. In this section, we collected noisy MRI data sets and evaluated the denoising performance of our algorithm against a wide variety of high-performance denoising algorithms. The nature of the noise was not known *a priori*. Noisy MRI data was obtained using a spin-echo imaging pulse sequence with different repetition times (TR) to alter the SNR. All measurements were performed on a 9.4 T vertical bore Varian VNMRS micro-imaging system, using a 40 mm-i.d. imaging probe. The imaging parameters were: TR = 100 ms for noise level 1, 50 ms for noise level 2, TR = 30 ms for noise level 3 (Fig. 9), and 2000 ms for the high SNR original image (Fig. 10). TE = 19 ms for all noise levels (Fig. 9), and TE = 18 ms for original image (Fig. 10). Matrix size = 512 $\times$ 512, and field of view (FOV) = 45 mm $\times$ 30 mm. We compared our method to BM4D, TV, adaptive non-local means filter [22] (AONLM), adaptive multiresolution non-local means filter [23] (ONLM), optimized blockwise Rician non local means filter [24], [25], [26] (ORNLM), oracle-based 3D discrete cosine transform filter [27] (ODCT), and prefiltered rotationally invariant nonlocal means filter [27] (PRINLM). Our MC-MLP algorithm outperformed all other methods in terms of all the metrics used (PSNR, FSIM, and MSSIM) for quantification. The results are shown in Fig. 9 and in Table I.

IV. CONCLUSION

We presented a feature-preserving image denoising algorithm in which a nonlinear filter is designed using a hierarchical multistage system of MLPs. From the point of view of conventional metrics (PSNR, FSIM, MSIM), the algorithm outperforms state-of-the art methods from low to high noise levels, and can handle both additive and muliplicative noises, including Gaussian and signal-dependent Rician nosies. For moderate to low noise levels, competing algorithms are limited to special cases where the known noise distribution meets narrow criteria. Our approach is general and is applicable to situations with arbitrary noise distributions and can even operate under extreme noise levels. It can also be used in situations where the noise distribution is not known, where it can still learn to model it from experimental data.

The filtering is computationally efficient and shows that multiple copies of the same image enable more effective noise removal with better preservation of anatomical features. Competing denoising algorithms tend to smooth images to the point where important anatomical details are lost. There are several possible scenarios in which our method could be applied. One such application is MRI, where low SNR or low contrast-to-noise ratio situations frequently arise. Namely, with low-field MRI, MRI of lower sensitivity nuclei (such as ^{23}Na or ^{31}P), diffusion tensor imaging in the presence of strong diffusion gradients, MR spectroscopy of metabolites at low concentrations or functional MRI. Other scenarios could include photography under poor lighting conditions or low exposure times, electron microscopy, x-ray, position emission tomography and ultrasound imaging. Our method could also be applied to video data using neighboring frames provided that the motion is not too large or that motion tracking is used. We have shown that as little as 7 copies are required for good performance, making the method practical in terms of data acquisition times, as low-SNR situations generally require far more than 7 signal averages.

ACKNOWLEDGMENT

The authors would like to thank Stanley Osher for useful discussions. This work was partially funded by AFOSR grant no. FA9550-11-1-0270, DARPA QuASAR and NIH grant no. 5-R01-HL114086-03.

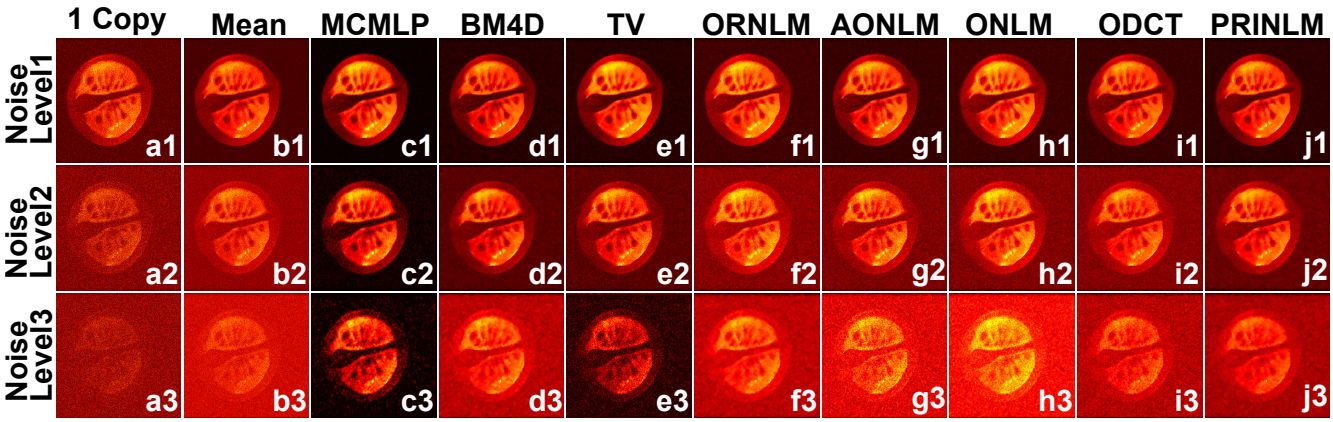

Fig. 9. Comparison of MC-MLP algorithm with several MRI denoising methods (BM4D, TV, ORNLM, AONLM, ONLM, ODCT, PRINLM) applied to a T_1-weighted image of a cherry tomato acquired on a Varian 9.4 T microimaging system using a spin-echo imaging sequence. Different noise levels (noise level 1, 2 and 3) were created by adjusting the TR (repetition time) value in the pulse sequence. The performance of each denoising algorithm is evaluated using the performance metrics of PSNR, FSIM, and MSSIM and the values are given in Table I. The MC-MLP algorithm outperformed other methods for all noise levels, according to all performance metrics. A high SNR image of the cherry tomato is shown in Fig. 10 for comparison. The salient feature of our algorithm is that not only the SNR of the denoised image is higher, even under conditions of extreme noise levels, but the features of the image are preserved as opposed to blurred out, as is the case for conventional algorithms.

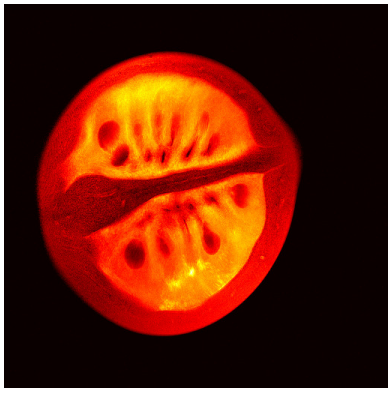

Fig. 10. High resolution MRI image of the cherry tomato used for evaluating denoising performance of the results of Fig. 9.

REFERENCES

[1] J. W. Tukey, "The extrapolation, interpolation and smoothing of stationary time series with engineering applications." *J. Am. Statist. Assoc.*, vol. 47, pp. 319–321, 1952.

[2] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *Proc. Int. Conf. Mach. Learn.*, vol. 8, pp. 689–696, 2009.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, pp. 3736–3745, 2006.

[4] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. Image Process.*, vol. 18, pp. 842–861, 2010.

[5] C. S. Anand and J. S. Sahambi, "Wavelet domain non-linear filtering for MRI denoising," *Magn. Reson. Imaging*, vol. 28, pp. 175–191, 1961.

[6] R. Yan, L. Shao, and Y. Liu, "Nonlocal hierarchical dictionary learning using wavelets for image denoising," *IEEE Trans. Image Process.*, vol. 22, pp. 4689–4698, 2013.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, pp. 1–16, 2007.

[8] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, pp. 490–530, 2005.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[10] R. W. Liu, L. Shi, W. Huang, J. Xu, S. C. H. Yu, and D. Wang, "Generalized total variation-based mri rician denoising model with spatially adaptive regularization parameters," *Magn. Reson. Imaging*, vol. 32, pp. 702–720, 2014.

[11] P. Getreuer, "Rudin-Osher-Fatemi total variation denoising using split bregman," *Image Processing On Line*, vol. 4.2, pp. 74–95, 2012.

[12] M. Maggioni and A. Foi, "Nonlocal transform-domain denoising of volumetric data with groupwise adaptive variance estimation," *Proceedings SPIE 8296, Computational Imaging*, 2012.

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, pp. 1–14, 2004.

[14] L. Zhanga, L. Zhanga, X. Moub, and D. Zhang, "FSIM: a feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, pp. 2378–2386, 2011.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[16] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networ.*, vol. 5, pp. 989–993, 1994.

[17] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" *Proc. CVPR IEEE*, pp. 2392–2399, 2012.

[18] K. Dabov, A. Danieyan, and A. Foi. (2014) BM3D demo software for image/video restoration and enhancement public release v2.00. http://www.cs.tut.fi/∼foi/GCF-BM3D/.

[19] H. Burger. (2012) Image denoising with multi-layer perceptrons. http://people.tuebingen.mpg.de/burger/neural_denoising/cvpr2012.html.

[20] M. Maggioni and A. Foi. (2013) BM4D software for volumetric data denoising and reconstruction public release ver. 2.3. http://www.cs.tut.fi/∼foi/GCF-BM3D/.

[21] P. Getreuer. (2009) reciandenoise: 2D and 3D total variation based Rician denoising. https://code.google.com/p/cdsc-image-processing-pipeline/downloads/list.

[22] J. Manjon, P. Coupe, L. Marti-Bonmati, D. Collins, and M. Robles, "Adaptive non-local means denoising of MR images with spatially varying noise levels," *J. Magn. Reson. Imaging*, pp. 192–203, 2010.

[23] P. Coupe, J. Manjon, M. Robles, and D. Collins, "Adaptive multiresolution non-local means filter for three-dimensional magnetic resonance image denoising," *IET Image Process.*, pp. 558–568, 2012.

[24] P. Coupe, J. Manjon, E. Gedamu, D. Arnold, M. Robles, and D. Collins, "Robust Rician noise estimation for mr images," *Med. Image Anal.*, pp. 483–493, 2010.

[25] P. Coupe, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot, "An optimized blockwise non local means denoising filter for 3D magnetic resonance images," *IEEE Trans. Med. Imaging*, pp. 425–441, 2008.

[26] N. Wiest-Daessle, S. Prima, P. Coupe, S. Morrissey, and C. Barillot, "Rician noise removal by non-local means filtering for low signal-to-noise ratio MRI: Applications to DT-MRI," 2008, pp. 171–179.
[27] J. Manjon, P. Coupe, A. Buades, D. Collins, and M. Robles, "New methods for MRI denoising based on sparseness and self-similarity," *Med. Image Anal.*, pp. 18–27, 2012.