ADDITIVE NOISE REMOVAL BY SPARSE RECONSTRUCTION ON IMAGE AFFINITY NETS

Rajagopalan Sundaresan[⋆]
University of Arizona

Fatih Porikli
Mitsubishi Electric Research Labs

ABSTRACT

This paper presents a new image denoising method based on sparse reconstruction by dictionary learning and collaborative filtering. First, we form an affinity net, in which a node represents an image patch, for the given image by clustering similar patches. For each cluster, we learn an undercomplete dictionary and represent clusters nodes by imposing sparsity inducing norm as a combination of few atoms. Depending on its affinity to other nodes, a single node could be present in multiple clusters making the clusters overlapping. This enables a single global estimation for each filtered pixel to be obtained by collaboratively aggregating its reconstructed patches in the corresponding clusters. Extensive experimental results demonstrate superior performance for additive noise removal without requiring the correct noise variance.

***Index Terms*—** Image denoising, sparse coding, dictionary learning, collaborative filtering

1. INTRODUCTION

Image denoising is considered a low level yet critical problem that has been tackled by a variety of techniques that rely on implicit and explicit modeling of the contaminated additive noise, which is often assumed to have a Gaussian distribution function. The additive image denoising is basically modeled as

$$y_k = x_k + z_k \tag{1}$$

where y_k, z_k and x_k correspond to the contaminated, original and noise intensities of the pixel k.

Several denoising approaches have been developed over the years. These can broadly be classified into local and nonlocal methods. While local methods have a limited estimation support for a consecutive weighted averaging, nonlocal methods utilize redundancy and work on larger support regions to provide better statistics. An extensive overview of the denoising methods can be found in [1], and we briefly discuss the most popular ones here to indicate the main differences and drawbacks.

The earliest local smoothing filters including Lee filter [2], Wiener filter, anisotropic bilateral filter [3] and their many variants compute a weighted average for each pixel based on the assumption that pixels inside the local window are sampled from the same distribution. To overcome their shortcomings particularly under high noise variance, sparsity inducing full-rank transformations such as DCT and wavelets are introduced. Among these techniques BLS-GSM [4] can be considered to one efficient denoising technique that preserves textures and edges.

Alternatively, nonlocal means based methods (e.g. [5]) obtain a single pixel estimate by computing a weighted average of all similar pixels in the image. These weights are assumed to be directly proportional to the similarity of the reference pixel (more specifically its surrounding patch) to the other pixels, either satisfy a predefined threshold or constitute a fixed size set. The basic insight of nonlocal methods is that patches in a natural image are redundant unlike random noise patterns.

Following on the nonlocal means idea, BM3D [6] exploits sparsity and nonlocal means based redundancy of patches to produce the state-of-the-art results. BM3D essentially imposes sparse representation on patch groups as hard thresholded scheme of DCT and wavelet decompositions to determine its Wiener filter responses it applies in its second stage. Inherently, it requires the noise variance to scale the Wiener filter coefficients. Recently, Aharon *et al* proposed the concept of KSVD [7], which constructs data driven dictionaries instead of the DCT and wavelet bases from image patches, to exploit the sparsity. In KSVD, an image patch is represented as a linear combination of a limited number of atoms in an overcomplete (4 or more times) dictionary and the final image is obtained as a weighted average of the reconstructed patches and the original noisy image assuming the noise variance is again known. Several other techniques, e.g. K-LLD (locally learned dictionaries) [8], LSSC (learned simultaneous sparse coding) [9] and CSR (clustering based sparse representation) [10], PPB (probabilistic patch based filter) [11], which poses denoising as a weighted maximum likelihood estimation, also share similar intuitions yet produce arguably incremental improvements in comparison to BM3D being dependent on the same constraints about the noise shape and variance.

Here, we propose a different strategy for additive noise removal using dictionary learning and collaborative filtering. We start by building an affinity net for the noisy image. In this affinity net, an image patch corresponds to a node that is linked (clustered) together with other similar patches. After determining the clusters in the net, we form a matrix by the nodes in each cluster and learn a thin dictionary that faithfully captures the underlying variations in the matrix. Each cluster is represented and filtered by its unique dictionary. The size of each dictionary is sufficiently very small since the affinity imposes a cluster to be constituted by closely similar patches. Since a single node most likely to participate in different clusters, there are overlaps among the clusters and hence multiple estimates for a single node (thus for its underlying pixels). The denoised estimate is obtained by combining the estimates obtained for a single pixel.

Our main contribution is the concept of adapting dictionaries to nonlocal variations through clusters present in the given image without any assumption on the noise function shape and variance. We demonstrate experimentally that our technique performs as well as the-state-of-the-art, even if they are supported by the correct noise variance, while generating visually pleasing results without oversmoothening. For simplicity, we call this affinity nets based collaborative filtering approach as Sparse Reconstruction on Affinity Nets (SRAN).

[⋆]This work was performed at Mitsubishi Electric Research Labs.

2. SPARSE RECONSTRUCTION ON AFFINITY NETS

2.1. Affinity Net

The affinity net is an undirected graph based representation of all image patches where each node corresponds to an image patch and the weight of each vertex to the similarity of two patches it belongs to. In other words, the affinity net is a weighted graph with the weights corresponding to the distance between the patches. The affinity between two vector form image patches p_i and p_j is computed as

$$\omega(p_i, p_j) \; = < p_i, p_j > = \sum_{k=1}^{K} p_{i,k} p_{j,k} \tag{2}$$

for the K pixels contained in a patch assuming each patch has the same number of pixels. Although it is possible to use other metrics (e.g. ℓ_1 weights, frequency features, etc.) to measure the similarity between patches, we preferred the dot product to not bias for any particular intensity and pixel position. The corresponding graph containing the clustered nodes is called as affinity net because clustering depends on similarity rather than distance.

We use the affinity net to learn a locally adapting dictionary for each node. However, using the entire affinity net to learn that many dictionaries would be computationally prohibitive since the affinity net contains as many clusters as the number of image pixels and it is fully connected. In order to decrease the computational load drastically and to concentrate on more relevant samples, we cluster nodes by keeping a maximum of m vertices that have sufficiently high weights $w(p_i, p_j) > \tau$ within a search window of radius r. A cluster c_i is denoted as

$$c_i = \left\{ \; p_j : \omega(p_i, p_j) > \tau, i = 1, .., n \; \right\} \tag{3}$$

for n pixels in image Y. Clustering process puts the nodes having close coordinates together in the feature space. We empirically observed that the noise removal performance is not sensitive to the value of τ as long as most clusters contain 10 to 100 patches. If the affinity between two nodes is high, a node is attached to the cluster of the other node as illustrated in Fig. 1. Depending on the self-similarity of the image, clusters can be in different sizes. A reference node might not contain any similar nodes and hence the cluster size will be one. Also, a single node could be present in one or more clusters and participate in filtering of each cluster they are present. In this case, we will obtain multiple estimates of a node.

The corresponding graph containing the clustered nodes is called as affinity net because clustering depends on similarity rather than distance. If the affinity between two nodes is high, a node is attached to the cluster of the other node as illustrated in Fig. 1. As shown, depending on the self-similarity of the image, clusters of can be in different sizes. The dark circles represent the reference nodes and the red circles are in their corresponding clusters. A reference node might not contain any similar nodes and hence the cluster size will be one. Also, a single node could be present in one or more clusters and participate in collaborative filtering in each cluster they are present. In this case, we will obtain multiple estimates of a node.

2.2. Dictionary Learning

For each of the affinity net cluster c_i, we form a cluster matrix $C_i^{M\times K}$ and decompose it into a linear function of a dictionary $D_i^{M\times d}$ and its sparse coefficients $X_i^{d\times K}$, where M is the number of nodes in the cluster, d is the number of atoms in the dictionary, and

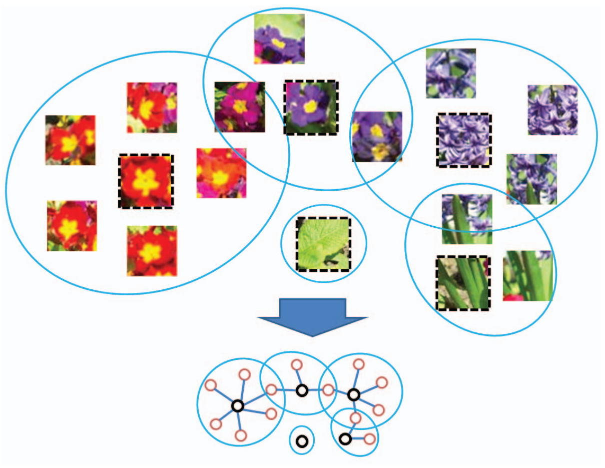


Fig. 1. Affinity net for a flower image. Note that some nodes may be singular with no vertex. Affinity net contains as many clusters as the number of nodes.

K is the number of pixels in the patch. We arrange the column vectors p_i of the cluster nodes into a cluster matrix as $C_i = [.., p_i, ..]^T$ as shown in Fig. 2. In C_i, each row corresponds to cluster node including the reference node. The number of columns in this matrix is the dimensionality (size) of the patch, while the number of rows is the number of patches in the cluster.

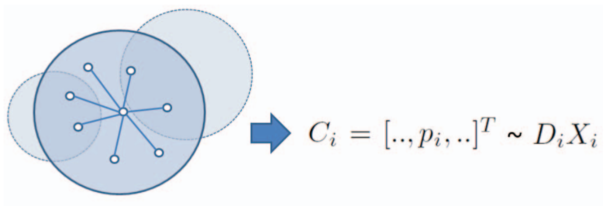


Fig. 2. Cluster matrix construction

As opposed to the traditional column-wise ordering, we use the row-wise ordering before learning the dictionary. Our intuition here is that the variation along the columns, that is, the intensity variation of the same pixel locations across the similar patches, should be very small since the patches are all similar. A pixel intensity that does not comply with its references in the other patches in the cluster should not be considered as a representative in the dictionary.

Let us consider a signal represented as $y \in \Re^M$. We consider that this signal is sparse in a dictionary $D \in \Re^{M\times d}$, thus the signal can be represented as a linear combination of a few (e.g. 1$\sim$3) atoms in the dictionary. In general, such dictionaries are overcomplete ($d >> M$), that is, the number of its columns (atoms) is greater than the number of its rows (data dimensionality). The sparsest representation X^* that is obtained with the dictionary D is given by the relaxed sparsity inducing norm

$$X^* = \arg\min_X ||| X ||_0 \qquad \text{subject to} || C - DX ||_2^2 \leq \epsilon \tag{4}$$

where the norm on the coefficients can be set to ℓ_1 to enable convex programming. For natural images, full-rank power compact dictionaries such as wavelet basis and DCT partially satisfy above objective when they are applied a hard shrinkage.

We jointly learn the dictionary and the sparse coefficients by alternating between a sparse coding stage and a dictionary update stage to solve the optimization problem

$$D_i^*, X_i^* = \arg\min_{D_i, X_i} ||x_j||_0 \,, \forall j \;\; ||C_i - D_i X_i||_2^2 \leq \epsilon \qquad (5)$$

where x_j is the coefficient column vector corresponding to pixel j in the patch grid and ϵ is the maximum allowed sparsity. We set the initial dictionary matrix with d random and ℓ_2 normalized columns of C_i. Alternatively, the k-means clustering (with d centers) can be applied to cluster nodes to obtain the initial dictionary.

In the sparse coding stage, we use the orthogonal matching pursuit to compute the representation vectors x_j. We update the dictionary one column at a time by first defining the group of rows (pixels) that use this atom then computing the overall representation error matrix for this group without using the current atom, and then applying singular value decomposition to assign the first column of the left decomposition matrix as the new updated atom. For the clusters that the number of nodes is very small (e.g. $M < 4$), we apply the Wiener in the second time application of our algorithm. We estimate the noise variance from the difference between the first time application result and the original noisy image.

2.3. Sparse Reconstruction

We are not aiming for a perfect reconstruction but rather elimination of the inconsistent pixel intensities by a linear combination of a few atoms in the dictionary. Because of this special row-wise structure of the cluster matrix, a small dictionary can successfully capture the coherent pixel intensity changes. Our cluster dictionary is therefore significantly under-complete; the number of columns d (and the maximum rank of the dictionary) is less than the patch size $d << K$.

After the dictionary learning and sparse coefficient computation, we have the new cluster matrix C_i^*. The rows of C_i^* are the reconstructed patches. We assign these reconstructed patches to the respective locations from where they are extracted. Because patches are overlapping and a patch may belong to multiple clusters, there are multiple estimates for a given pixel.

The final estimate at a pixel is obtained by computing the average of all computed estimates. Aggregation can also be performed by considering the affinity net edge weights at a pixel and then normalizing the corresponding estimates. This collaborative aggregation enables preserving commonly shared patterns and rejecting speckle noise at the same time. In other words, for more than one estimate of a pixel, we simply average all the obtained estimates.

Note that, none of the above formulations use the noise variance, or assume the noise distribution shape is Gaussian. The SRAN algorithm is summarized here:

```
Input: noisy image $Y$, initial estimate $Y_0 = Y$
Output: denoised image $I$
cluster patches $p_i \in Y_0 \rightarrow c_i$
for each $c_i$ do
    arrange $p_i \in Y$ (not $Y_0$) into cluster matrix $C_i$ using $\omega(p_i, p_j)$
    from $Y_0$
    if $M < 4$ and first run then
        $C_i^* \leftarrow$ Wiener($C_i$)
    else
        $C_i^* = D_i^*, X_i^* \leftarrow$ Eq. 5
    end if
end for
$I \leftarrow$ back project and aggregate $C_i^*$.
```

	Barbara	Camera.	House	Lena
Noisy	16.47	16.63	16.31	16.34
KSVD	26.35	24.84	29.41	28.61
BM3D	27.99	25.46	30.74	29.92
SRAN Filter	27.48	25.48	30.52	29.44

Table 1. PSNR comparison between various filters

3. EXPERIMENTS

We tested SRAN filter with images corrupted by simulated additive noise. In order to make a fair comparison with the existing methods (BM3D and KSVD based denoising) the noise is assumed to be Gaussian with zero mean. We set the same noise variance in their formulations when we generated the best results of BM3D and KSVD. Benchmark images are assessed quantitatively and perceptually.

The first step of our algorithm involves block matching. Three parameters of this step may influence the performance of our filter. These are the patch size, search range, similarity threshold. These parameters have to be optimized with a trade-off between the execution time and quality of the final results. Since we performed all our experiments with fixed parameters of the patch size $K = 64 = 8 \times 8$ and the search range $r = 39$. We limited the minimum similarity between nodes to $\tau = 0.002$ and the maximum number of similar nodes in a cluster to $m = 400$. Our experiments show that performance does not require manual tuning for each image. We used a simple hard thresholding scheme suggested in [6] to obtain the initial estimate.

Figure 3 show the denoised images. The noise variance in all these images are set to $\sigma = 40$ (for max intensity 255) because we wanted to test the performance of these filters under heavy noise conditions. From the results, it is evident that certain artifacts produced by the other algorithms are not present in our results. Though our filter produces smooth results wherever the noise is dominant, the finer details of the original image are preserved.

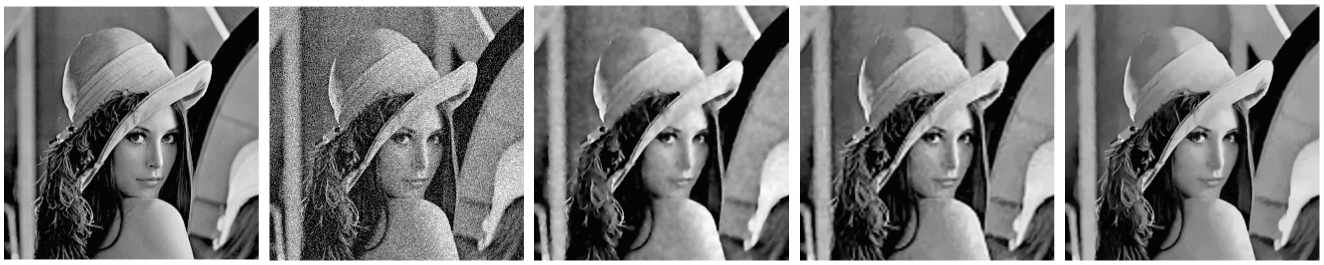
Sample quantitative evaluation scores of our method is presented in Table 1 for $\sigma = 40$ for Peak Signal to noise ratio (PSNR). As shown our method produces results that are quantitatively comparable (and sometimes better) to the state-of-the-art even though no assumption on the noise shape and variance is made.

4. CONCLUSION

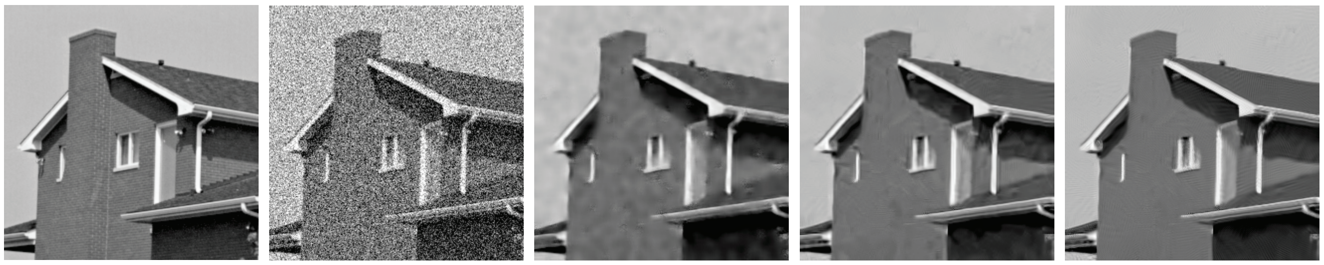
We proposed an image filter based on dictionary learning on affinity nets. Our filter takes advantage of the sparse representations in each cluster and removes additive noise by sparse approximations. As opposed to [7] that uses a single dictionary to represent all the image patches, our method has a very small undercomplete dictionary for each cluster. This provides a significant performance improvement as suggested by the experimental results. More importantly, as opposed to BM3D and KSVD, our method does not need to know the noise variance.

5. REFERENCES

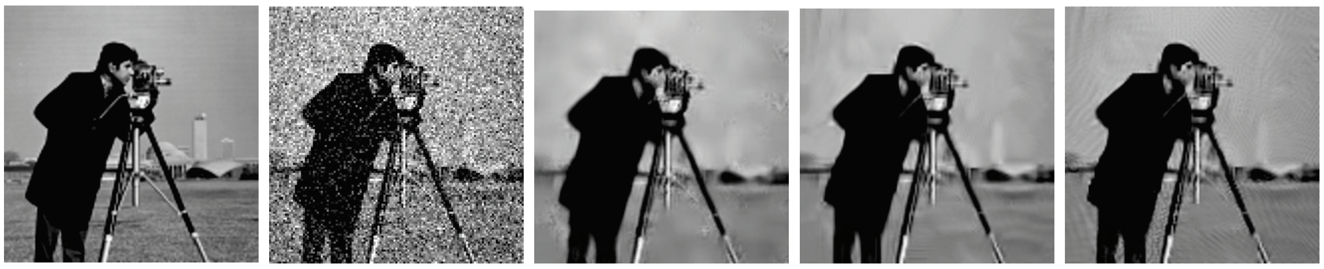
[1] V. Katvonik, A. Foi, K. Egiazarian and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *International Journal of Computer Vision*, vol 86 ,no . 1, pp 1-32, January 2010.



(a) Original b) PSNR = 16.34 dB (c) PSNR = 28.61 dB (d) PSNR = 29.92 dB (e) PSNR = 29.44 dB



(a) Original b) PSNR = 16.31 dB (c) PSNR = 29.41 dB (d) PSNR = 30.74 dB (e) PSNR = 30.52 dB



(a) Original b) PSNR = 16.63 dB (c) PSNR = 24.84 dB (d) PSNR = 25.46 dB (e) PSNR = 25.48 dB

Fig. 3. Comparison of denoising results on images corrupted by additive white Gaussian noise with standard deviation 40. a) Original images, b) noisy imaged, c) KSVD denoising results with given correct noise variance, d) BM3D filtering results with given correct noise variance, e) Proposed SRAN filter. PSNR-wise SRAN clearly outperforms (+1.5dB) the KSVD denoising method. Note that unlike BM3D and KSVD, our method does not require the noise variance to be known. Also, image edges are better preserved (e.g. *House*'s chimney and *Lena*'s hair) than BM3D and KSVD.

[2] J.S Lee, ”Digital Image Enhancement and noise filtering by use of local statistics,” *IEEE Pat. Anal. Mach. Intell*, vol. PAMI-2, pp. 165-168, March 1980

[3] C. Tomasi and R. Manduchi, ”Bilateral filtering for gray and color images,” in *Proc.* 6^{th} *Int. Conf. Computer Vision*, Washington, DC, Jan 1998, pp. 839-846.

[4] J. Portilla, V. Strela, M. J. Wainwright and E.P. Simoncelli, ”Image denoising using a scale mixture of Gaussians in the wavelet domain,” in *IEEE Trans. Image Process,*, vo. 12, no. 11, pp. 1338-1351, Nov. 2003

[5] A. Buades, B. coll, and J-M. Morel, ”A non local algorithm for image denoising,” in *proc. IEEE Conf. Computer vision and Pattern Recognition*, Oct 2005, vol. 2, pp. 60-65.

[6] K. Dabov, A. Foi, V. Katvonik, and K. O. Egiazarian, ”Image denoising by sparse 3D domain collaborative filtering,” *IEEE Trans. Image Process.*, vol.16, no.8, pp. 2080-2095, Aug.2007.

[7] M.Elad and M.Aharon, ”Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Trans. Image Process.*, vol.15, no.12, pp. 3376-3745, Dec.2006.

[8] P. Chatterjee and P. Milanfar, ”Clustering based denoising with locally learned dictionaries,” *IEEE Trans. Image Process.*, vol.18, no.7, pp. 1438-1451, July 2009.

[9] J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman, ”Non-local sparse models for image restoration ,” in *Proceedings of the* 26^{th} *Annual International Conference on Machine Learning*, 2009, pp. 2272-2279.

[10] W. Dong, X. Li, L. Zhang and G. Shi, ”Sparsity-based Image Denoising via Dictionary learning and Structural Clustering,” *proc. IEEE Conf. Computer vision and Pattern Recognition*, 2011.

[11] C. Delledale, L. Denis and F. Tupin, ”Iterative Weighted Maximum Likelihood Denoising with Probabilistic Patch-Based Weights,” *IEEE Trans. Image Process.*, vol.18, no.12, pp. 2661-2672, Dec 2009.