CS7640: Advanced Image Processing **Danny Perry**

Sparse coding and non-local image denoising

1 Introduction

One of the classic problems in image analysis is image denoising. Some of the landmark approaches to image denoising include [8] and [9]. Each of these model the denoising problem using partial differential equations, as anisotropic diffusion in the first and total variation over the image in the second.

Another approach to image denoising is to "learn" what a certain patch distribution looks like, and then to take the expectation of that distribution. [6] introduce the algorithm NL-means that makes use of this kind of learning approach. It is defined by this formula:

$$NL[u](x) = \frac{1}{C(x)} \int_\Omega \exp((-G_a * |u(x+.) - u(y+.)|^2)(0)/h^2) u(y) dy \tag{1}$$

where $u(\cdot)$ is the noisy image, $x, y, z \in \Omega$, G_a is a Gaussian kernel and h is a filtering parameter, and

$$C(x) = \int_\Omega \exp((-G_a * |u(x+.) - u(z+.)|^2)(0)/h^2) dz,$$

For a discrete noisy image $v = v(i)|i \in I$,

$$NL[v](i) = \sum_{j \in I} w(i,j) v(j), \tag{2}$$

where

$$w(i,j) = \frac{1}{Z(i)} \exp(-||v(N_i) - v(N_j)||_2^2 / h^2),$$

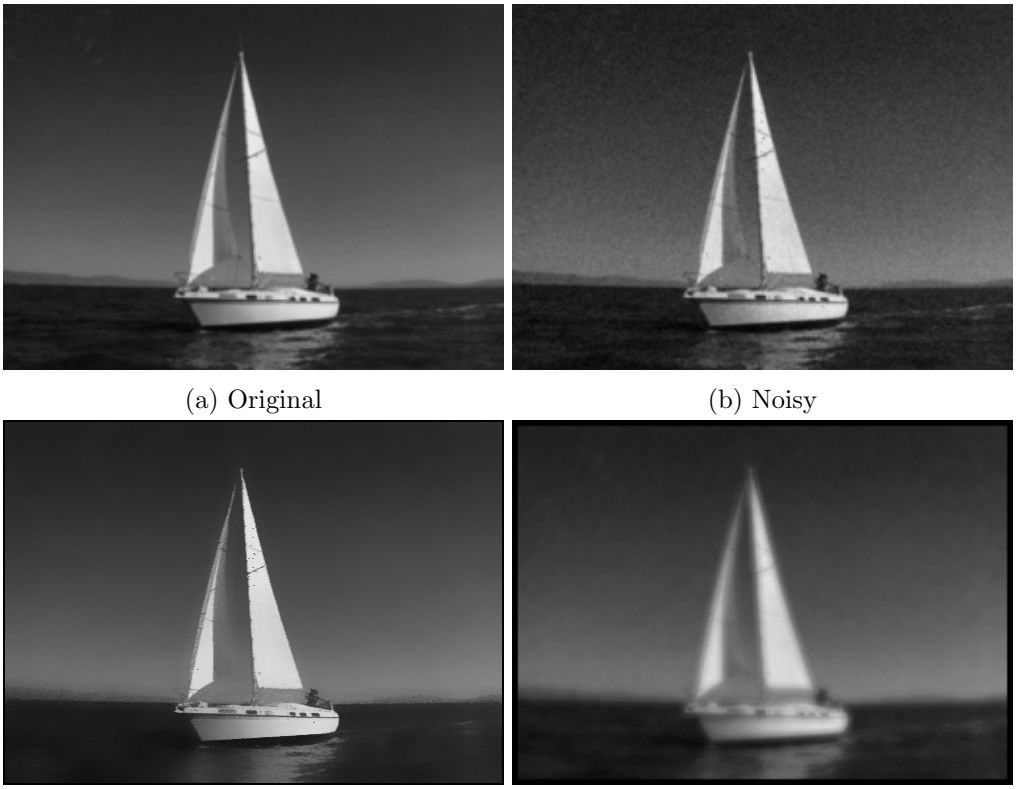
and

$$Z(i) = \sum_k \exp(-||v(N_i) - v(N_k)||_2^2 / h^2)$$

A closely related approach was presented in [4], where instead of finding the expectation of the patch distribution, the entropy of the patch was minimized using gradient descent.

Denoising using sparse representation of images can be described in a similar way. Instead of modeling the original distribution of the (potentially noisy) patches, sparse representation restricts the distribution of the patch to a subspace defined by an overcomplete dictionary. A smoothing over this restricted representation is then performed. K-SVD [3] is one popular approach to approximating the sparse representation, which has been shown to have a noise rejection capability [7]. Such a denoising approach is presented in [7].

I've implemented the NL-means and K-SVD-based denoising algorithms using the Julia language [5], a language similar in syntax to Matlab, but with some potential speed advantages. Here I evaluate both algorithms by denoising a variety of images.



(a) Original (b) Noisy

(c) NL-Means (d) K-SVD-based Denoising

Figure 1: Results of each algorithm on a noisy image of a sailboat.

2 Results

Each algorithm was run against several images. The results of three of those images are show below: the cameara man image (Figure 5), a sailboat image (Figure 1), and a slice of a seismic image (Figure 7).

For the cameraman and sailboat images Gaussian noise with $\sigma^2 = 100$ was added to the image. This was done to make the denoising effect more apparent.

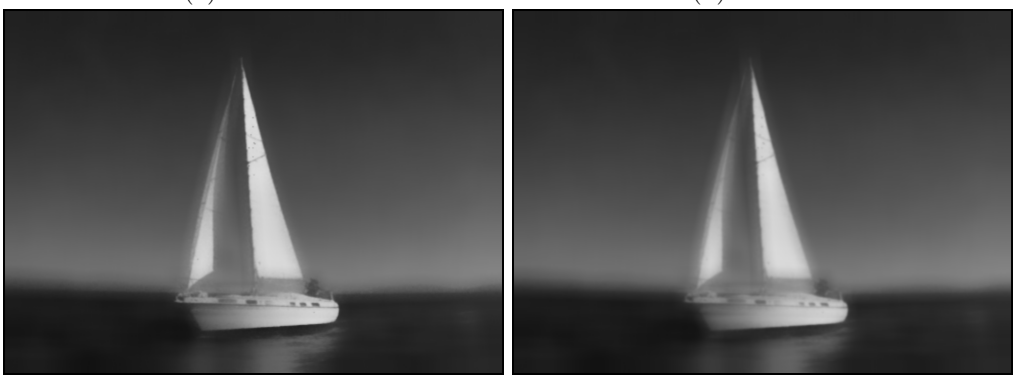
For the slice of seismic volume, the image was equalized to bring out the horizons. Denoising was applied with the goal of making the horizons appear more continuous. The seismic image is a slice taken from the Netherlands data set found on the OpendTect data repository [1].

2.1 NL-Means algorithm

The NL-means algorithm is very straightforward to implement. However, the algorithm is deceivingly costly. As summarized above, the naive implementation requires $O(n^4)$, with an image of size n by n.

Luckily, for a large class of images, the pixels most similar to a specific pixel neighborhood will be those in the immediate vicinity. This property is known as the "mixing property" in a markov random field [2].

(a) $\sigma^2 = 0.2$ (b) $\sigma^2 = 0.6$



(c) $\sigma^2 = 1.0$ (d) $\sigma^2 = 1.4$

Figure 2: Varying the σ^2 parameter for the NL-Means algorithm, with $h = 0.1$ constant.

For this reason, as noted in [6], when running the algorithm a search window is usually specified much smaller than the entire image. [6] specifically recommends a search window of size 21 by 21, and a patch size of 7 by 7. I did some experimentation but essentially use these or similar parameters.

Besides the patch size and search window size, the NL-means algorithm requires an additional two parameters: σ^2, the variance of the Gaussian kernel, and h the support of the similarity window. I experimented with various values for these parameters as well, and show some results.

Results on various images are shown, for example, the result of using NL-Means on the sailboat image is shown in Figure 1c. Results for the cameraman image is shown in Figure 5c, and for the slice of seismic imagery is shown in Figure 7c. Note how the algorithm is able to smooth while still retaining details in the image.

The least sensitive parameter is probably the Gaussian kernel variance, σ^2. Adjusting that parameter by small amounts seemed to influence the outcome in smaller steps than h. For example, Figure 2 shows some results of varying the kernel variance. What is essentially happening here is that the similarity between two patches is being slightly modified with the different kernels. With a Gaussian kernel with low variance, differences will be emphasized, providing less opportunity for blurring. With a Guassian kernel with high variance, the differences will be less strong, providing more opportunity for blurring.

The filter support parameter h is much more sensitive to a given value. Smaller changes in h evoke large changes in the sharpness/bluriness of the image. For example 3 shows some results with

(a) $h = 0.2$ (b) $h = 0.6$

(c) $h = 1.0$ (d) $h = 1.4$

Figure 3: Varying the h parameter for the NL-Means algorithm, with $\sigma^2 = 0.2$ constant.

varying support sizes. Similarly to σ^2, as the value increases there is more smoothing going on in the images, but the effect appears to be much more sensitive to the changes in input.

2.2 K-SVD Based Denoising

The K-SVD denoising algorithm is a little more complex than the NL-Means. While the algorithm coding isn't horrible, the interaction of the various parts make it a little more sensitive to various parameters, etc.

Also, similar to NL-Means, theoretically you can get some pretty bad upper bounds on runtime and storage requirements. A naive impelementation that does not make use of a sparse matrix data structure will quickly use a surprising amount of memory, if a large dictionary and a large number of image patches are used (requiring the sparse representation vector for all patches to be a large matrix in both dimensions).

However, [7] recommends some specific parameters that help to minimize those costs. For example, they use smaller 256 by 256 size images to limit the number of patches to encode. For larger 512 by 512 images they recommend skipping every other line, again to keep that dimension to a reasonable figure. Other recommended parameters include using an overcomplete dictionary with 256 atoms. Again, this size keeps the space and runtime requirements reasonable. Other parameters that need tuning include the sparsity limit (they recommend using under 6), the limit placed on the residual after each matching pursuit and K-SVD iteration (recommended to be adjusting according

(a) Denoising usine DCT-based dictionary. (b) Noisy-image based dictionary (K-SVD).

Figure 4: Results of using the sparse dictionary based denoising algorithm on the cameraman image. One used a DCT-based dictionary while the other used a dictionary trained using K-SVD on the noisy image. The differences are subtle, and difficult to convey visually.

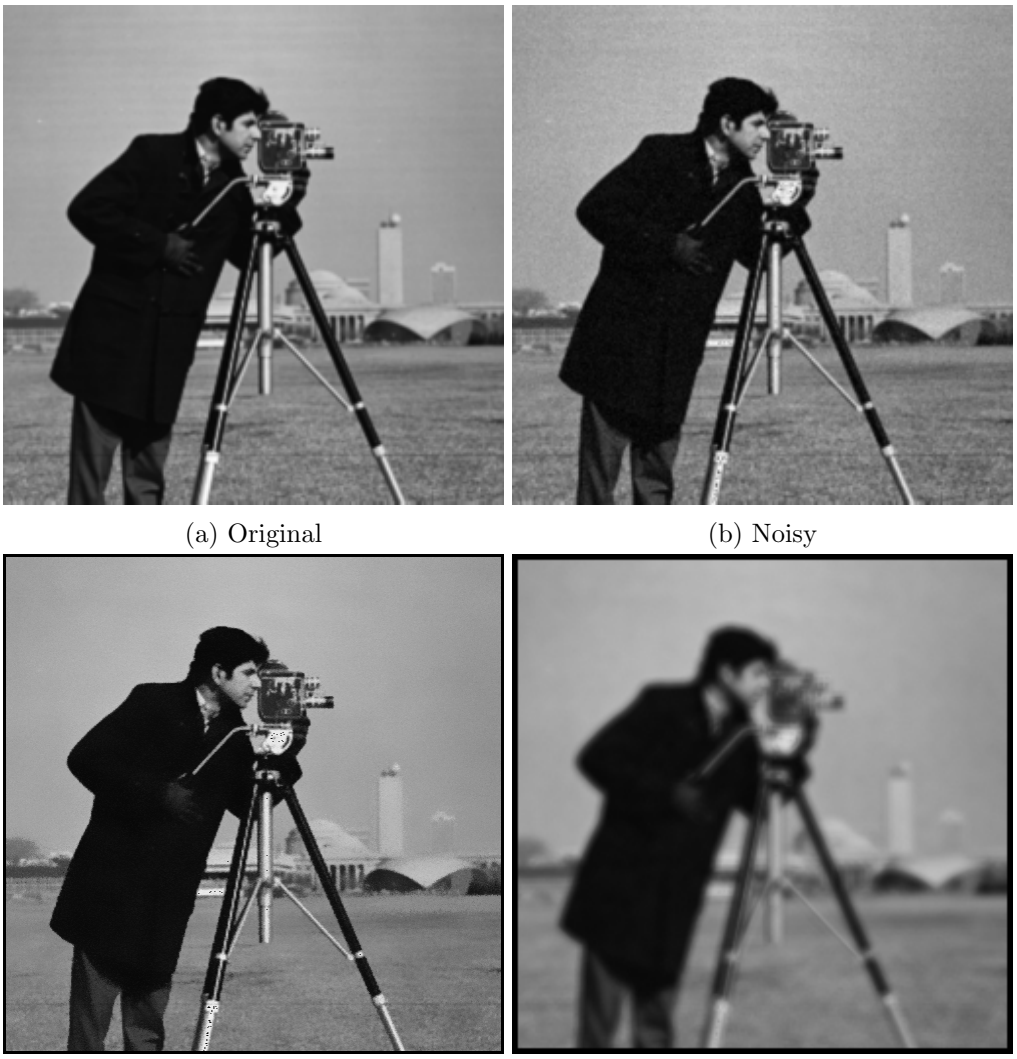
to the variance of the noise model σ^2, for example $(1.15 * \sigma)^2$), as well as the relaxation coefficient (they recommend this being a ratio of the noise model - for example $30/\sigma$). I use these or similar parameters for the results shown here.

Results are shown for the K-SVD-based denoising in Figures 1d,5d, and 7d. You will note that the K-SVD denoised images are very blurry. While I was able to successfully decompose and reconstruct an image using the a dictionary and the K-SVD algorithm, I had some trouble tuning the parameters for denoising the images, and because the assignment is due, I unfortunately present these results as is.

I was able to implement the denoising using a random dictionary, a DCT based dictionary, and a dictionary constructed from the noisy images itself. Here I show some results comparing using the DCT and noisy-image based dictionaries. Figure 4 compares the denoising results using the different dictionaries on the noisy cameraman image. Likewise, Figure 6 compares some results using the different dictionaries on the noisy sailboat image.

3 Conclusion

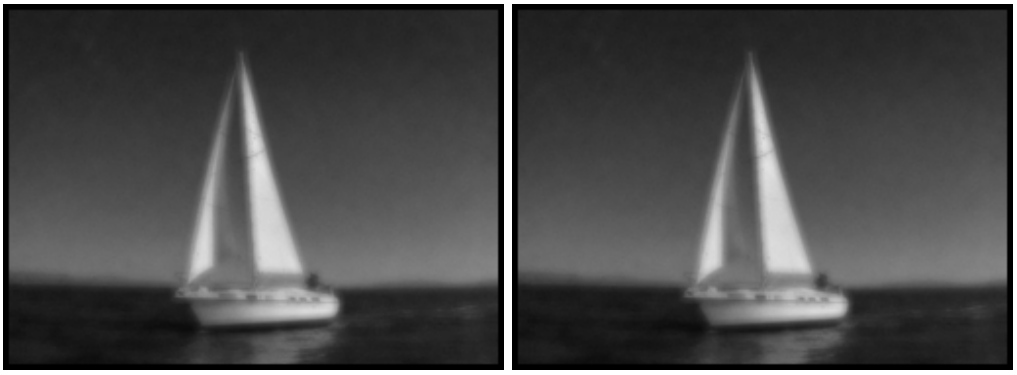
The classic problem of denoising has been attacked from a variety of approaches. Here we've presented two additional methods of denoising - NL-Means based on the expectation of the distribution of a patch in the image, and Sparse dictionary based denoising - using the decomposition of the image using an overcomplete dictionary to denoise. While the results here for the overcomplete dictionary aren't overwhelmining satsifying, the exercise did convey the flavor of the approach. And by considering the original results, and the results presented here for NL-Means, one can conclude that both techniques provide for interesting approaches to a classic problem.



(a) Original (b) Noisy

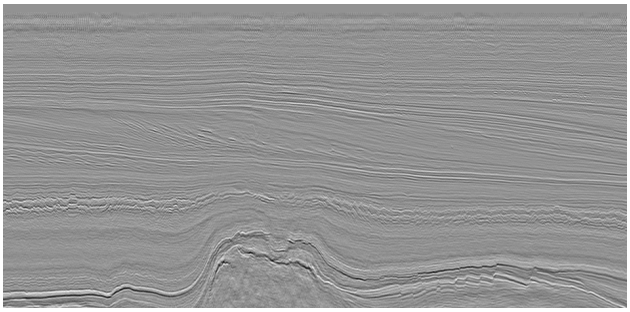
(c) NL-Means (d) KSVD-based denoising

Figure 5: Results of each algorithm on a noisy image of a cameraman.

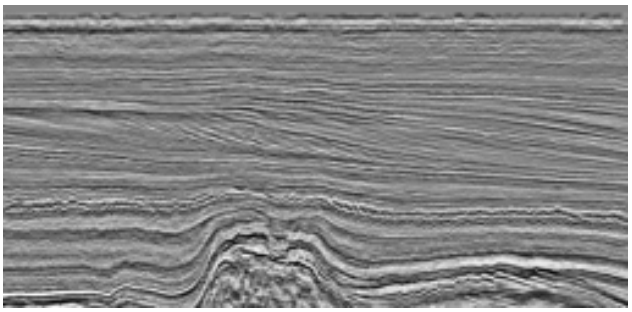


(a) Denoising usine DCT-based dictionary. (b) Noisy-image based dictionary (K-SVD).

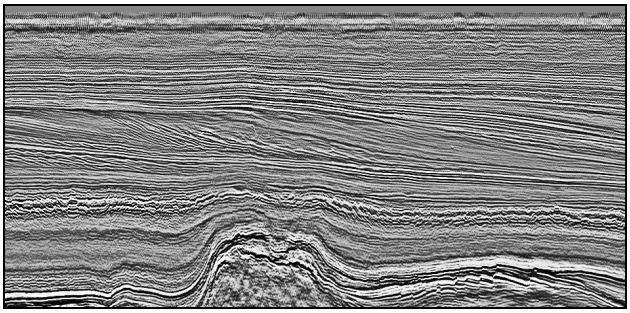
Figure 6: Results of using the sparse dictionary based denoising algorithm on the sailboat image. As in Figure 4, the differences are subtle, and difficult to see visually.



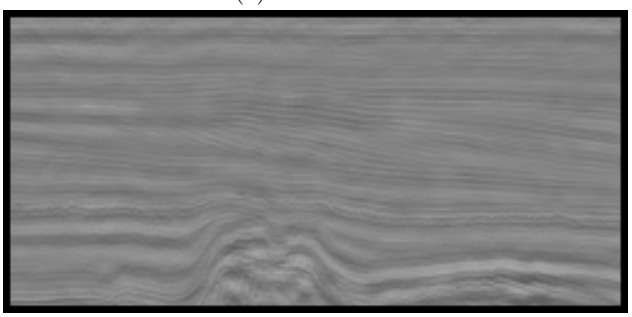
(a) Original



(b) Equalized no denoising



(c) NL-Means



(d) K-SVD-based denoising

Figure 7: Results of each algorithm on a slice of a seismic volume from the OpendTect project [1]. The slice has been equalized prior to smoothing to enhance appearance of horizons.

References

[1] Opendtect seismic repository. `http://www.opendtect.org/index.php/share-seismic-data/osr.html`. Accessed: 02/25/2013.

[2] Wikipedia: Markov random field. `http://en.wikipedia.org/wiki/Markov_random_field`. Accessed: 04/05/2013.

[3] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: Design of dictionaries for sparse representation. *Proceedings of SPARS*, 5:9–12, 2005.

[4] Suyash P Awate and Ross T Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):364–376, 2006.

[5] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[6] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[7] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006.

[8] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[9] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.