EECS 556 Project: Final Report
Investigation of the Non-local Means Algorithm

Brittan Farmer, Arun Sundar Govindarajan, and Raj Tejas Suryaprakash

1 Introduction

Image denoising has been a subject of interest in the field of image processing for many years. Noise is inherent during image acquisition. Reducing the amount of noise in an image makes the image more pleasing to the eye and it is also an important pre-processing step since it improves the performance of high level tasks such as edge detection and object tracking.

There are many different denoising algorithms, but most belong to one of the following three classes:

1. Filters that act on a local region within an image, like mean, median or Gaussian filters;
2. Filters that take the entire image into consideration, such as frequency domain filters which reduce noise in the Fourier or wavelet domain; and
3. Neighborhood filters, which act on pixels with similar gray level values.

In this report, we study the non-local means algorithm of Buades, Morel and Coll [1, 2], which is a variation of the third kind of algorithm, since the weighting is based on the similarity of the patches. In section 2, we introduce the algorithm, optimize its performance with respect to its parameters, and compare it with the NPLS algorithm. In section 3, we study a variation of non-local means aimed at reducing the computational complexity [3]. In section 4, we explore several variations of the non-local means algorithm [4, 5] which are intended to improve the quality of the denoised images.

2 The Non-local Means Algorithm

For a discrete 2-D image $v[n,m]$ of size $N \times M$, let $\mathcal{N}_{n,m}$ denote the square neighborhood of fixed size and centered at pixel $[n,m]$. We shall refer to such neighborhoods as "patches." Then, the non-local means algorithm is defined by the formula

$$NL[f][n,m] = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} w[n,m;k,l] f[k,l],$$

where the weights $\{w[n,m;k,l]\}$ are given by

$$w[n,m;k,l] = \frac{1}{Z[n,m]} e^{-\frac{\|f(\mathcal{N}_{n,m}) - f(\mathcal{N}_{k,l})\|_{2,a}^2}{h^2}}$$

and where the Euclidean norm is weighted by a Gaussian kernel matrix with standard deviation a and $Z[n,m]$ is the normalizing constant

$$Z[n,m] = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} e^{-\frac{\|f(\mathcal{N}_{n,m}) - f(\mathcal{N}_{k,l})\|_{2,a}^2}{h^2}}.$$

The more similar the patch centered at $[k, l]$ is to the patch centered at $[n, m]$, the greater the weight it has in the average used to estimate $f[n, m]$.

Under certain stationarity assumptions, as the size of the image grows, the non-local means algorithm converges to the conditional expectation of a pixel given the neighborhood around it. This conditional expectation is the function of the surrounding neighborhood which gives the optimal mean squared error. Thus, the non-local means can be viewed as asymptotically optimal.

There are several parameters present in the non-local means algorithm. One is the size of the patch. We will assume that square patches of size $P \times P$ are used. The width P of the patch is a parameter. We will often refer to the patch radius p which gives a $(2p + 1) \times (2p + 1)$ patch. Note that $O(P^2)$ operations are necessary to compute the L_2 norm of the difference of two patches. The L_2 norm can be weighted by a Gaussian kernel, which gives a greater significance to pixels in the center of the patch. None of the articles mentioned how a is chosen, so in all of our tests, we used $a = 0$, which simply has the effect of normalizing the L_2 norm by dividing by the area of the patch.

As described above, to calculate the NL-means image at a single pixel, we must compute the weights of all other pixels in the image. If the image is $N \times N$, then this requires $O(N^2P^2)$ operations for each pixel, giving a total of $O(N^4P^2)$ operations. For a large image, this is a huge number of computations. To reduce the number of operations, the authors of [1, 2] introduce an $S \times S$ search window around a pixel, outside of which all weights are assumed to be zero. Again we will often refer to the search window radius s which gives a $(2s + 1) \times (2s + 1)$ search window. This amounts to using weights

$$w[n, m; k, l] = \frac{1}{Z[n, m]} \operatorname{rect}_2\left(\frac{s}{2}(k - n), \frac{s}{2}(l - m)\right) e^{-\frac{\|f(\mathcal{N}_{n,m}) - f(\mathcal{N}_{k,l})\|_{2,a}^2}{h^2}}.$$

This is reasonable since we expect most of the similar patches to be spatially close to the given patch. Using the search window reduces the number of computations to $O(N^2S^2P^2)$.

The filtering parameter h allows one to adjust the width of the averaging kernel. Considered as a function of the L_2 norm of the difference of patches, the averaging kernel is $w(d) = \exp(-d^2/h^2)$, that is an unnormalized Gaussian with standard deviation h. If $h \approx 0$, then the kernel will be very narrow, and the weights will be 1 when the two patches are exactly the same and approximately 0 when the patches have any difference. If $h \approx \infty$, then the kernel will be very wide, and all the weights will be approximately 1, regardless of the similarity of the patches. We want to choose h large enough so that some averaging occurs and noise is removed. However, if h is too large, too much averaging occurs and details will be lost. So we must also choose h small enough that a pixel is only averaged with similar patches.

We expect that h should depend on σ, the standard deviation of the additive noise. Suppose we have an image $f[n, m]$ and noise $v[n, m]$ which are jointly WSS and uncorrelated with known auto-correlation functions $R_f[n, m]$ and $R_v[n, m] = \sigma^2\delta_2[n, m]$. If we apply NL-means to the noisy image $g[n, m] = f[n, m] +$

$v[n,m]$, for the L_2 norm of the difference of the patches at $[n,m]$ and $[k,l]$ we have

$$
\begin{aligned}
&E\left[\|g(\mathcal{N}_{n,m}) - g(\mathcal{N}_{k,l})\|_{2,0}^2\right] \\
&\quad = E\left[\frac{1}{(2p+1)^2}\sum_{\alpha,\beta=-p}^{p}\left(g[n+\alpha,m+\beta]-g[k+\alpha,l+\beta]\right)^2\right] \\
&\quad = \frac{1}{(2p+1)^2}\sum_{\alpha,\beta=-p}^{p} E\left[\left((f[n+\alpha,m+\beta]+v[n+\alpha,m+\beta]) - (f[k+\alpha,l+\beta]+v[k+s,l+t])\right)^2\right] \\
&\quad = \frac{1}{(2p+1)^2}\sum_{\alpha,\beta=-p}^{p} E\left[f[n+\alpha,m+\beta]^2 + v[n+\alpha,m+\beta]^2 + f[k+\alpha,l+\beta]^2 + v[k+s,l+t]^2\right. \\
&\qquad\qquad \left. - 2(f[n+s,m+t]f[k+\alpha,l+\beta] + v[n+\alpha,m+\beta]v[k+\alpha,l+\beta])\right] \\
&\quad = \begin{cases} 0, & n=k \text{ and } m=l \\ 2\left(\sigma^2 + R_f[0,0] - R_f[n-k,m-l]\right), & \text{otherwise.} \end{cases}
\end{aligned}
$$

Suppose the image were $f[n,m]$ is a constant so that $R_f[n,m]$ is also constant. Then, the expected L_2 norm of the difference is simply $2\sigma^2$. Since the patches of the original image were exactly the same, we would want to average such pixels in order to reduce the noise. If the image has a more complicated autocorrelation function, or if the image is not WSS, then the expected value of the L_2 norm is more difficult to analyze. However, it is still sensible to assume that we should average together patches with L_2 norms on the order of $2\sigma^2$ or less.

In [1, 2], the authors use the gaussian function for their averaging kernel. This is probably optimal in the case of a constant image with gaussian noise, but it is not obvious that this would apply in more general situations. One has the freedom to choose other decreasing functions. For instance, when we began our project, the authors of [1, 2] had published on their IPOL website (`http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/`) the following weights:

$$
w(d) = \begin{cases} 1, & d < 2\sigma^2 \\ 1 - \frac{d^2-2\sigma^2}{2\gamma}, & 2\sigma^2 \le d^2 \le 2\sigma^2 + 2\gamma \\ 0, & \text{otherwise,} \end{cases}
$$

where γ is the expected value of the L_2 norm of two patches. It would be difficult to estimate γ from a noisy image, so it would be good to avoid estimating it. We tried using the function $w(d) = e^{-d^4/h^4}$, which is flatter than the gaussian for small d but has sharper decay for large d (see figure 1). We found this to work quite well. (While writing the report, we checked the IPOL website again and noticed that they had changed the weight to $w(d) = \exp(\max(d^2 - 2\sigma^2, 0)/h^2)$, which is flat near $d = 0$ and drops off exponentially for $d^2 > 2\sigma^2$.)

In [1, 2], the authors use a patch size of 7×7 pixels, a search window size of 21×21 pixels, and a filtering parameter of $h = 10 * \sigma$. We wanted to test whether these parameters were optimal. It is difficult to quantify when a denoised image is "optimal" in terms of visual quality, so we use the mean squared error (MSE) as an objective measure of the quality of the denoised image. We use the 512×512 "peppers" test image and add white Gaussian noise with standard deviation 30. Note that in our experiments we assume σ is known, but in practice, it would need to be estimated.

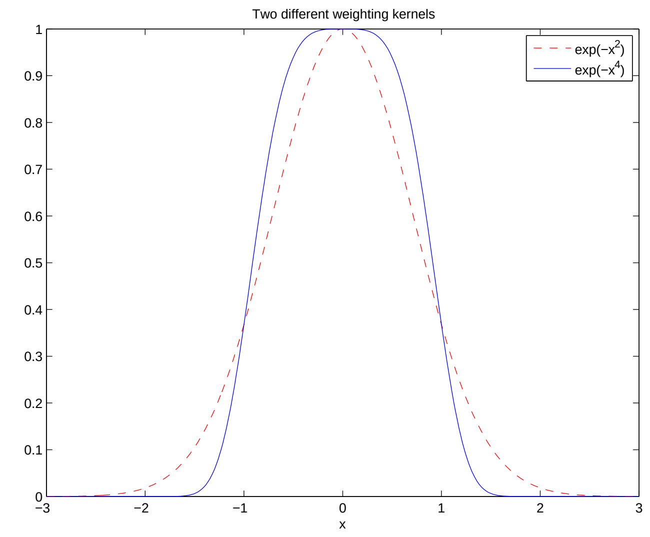

Figure 1: The $\exp(-x^4)$ is flatter than the $\exp(-x^2)$ near $x = 0$ and has sharper decay.

First, we consider varying the patch radius. We vary the patch radii between $p = 1$ and $p = 5$, which leads to patches of odd sizes from 3×3 to 11×11. The other parameters were held fixed at $h = 2\sigma$ and $s = 10$, i.e. we used a 21×21 search window. We find that the optimal MSE is achieved when the patch radius is 1, that is, for 3×3 patches (see figure 2). This does not necessarily give the most visually appealing results, as can be seen in figure 3.

Next, we vary the search window radius s. We consider radii between 1 and 10, which give search windows with odd sizes from 3×3 to 21×21. The other parameters were held fixed at $h = 2\sigma$ and $p = 1$, i.e. we used 3×3 patches. We find that the optimal MSE is achieved when the search window radius is $s = 4$, that is, for 9×9 windows (see figure 4). Our reasoning for this result is that if the search window is too small, not enough pixels are averaged and noise is not removed. If the search window is too large, the algorithm assigns weights to patches far away in the image which may only be similar because of the additive noise. As a result too many pixels are averaged, and the image becomes blurred. This is evident in the representative images for $s = 1, 4,$ and 10 (see figure 5).

Finally, we consider the filtering parameter h. Since we expect the optimal h to increase as σ increases, we normalize h by dividing by σ. We vary h/σ between 1 and 3, while fixing $p = 1$ and $s = 4$. We find that the optimal parameter is $h/\sigma = 1.8$ (see figure 6). This allows for a kernel that is wide enough that we expect similar original patches to be averaged, even though they are made less similar because of noise. It is also narrow enough to prevent dissimilar patches from having too large a weight. In figure 7, we see this balance between under- and over-smoothing.

Although the graphs pictured are for the peppers image, we performed the same analysis for the bricks image and found nearly identical optimal parameters.

Now that we have found parameters that allow for the non-local means algorithm to produce images with an optimal MSE, we are ready to compare it with another denoising algorithm. We will not attempt to make an exhaustive survey of denoising methods, since such a survey is already present in [2]. Instead, we compare to the non-quadratically penalized least squares (NPLS) algorithm described in class, which is similar to the

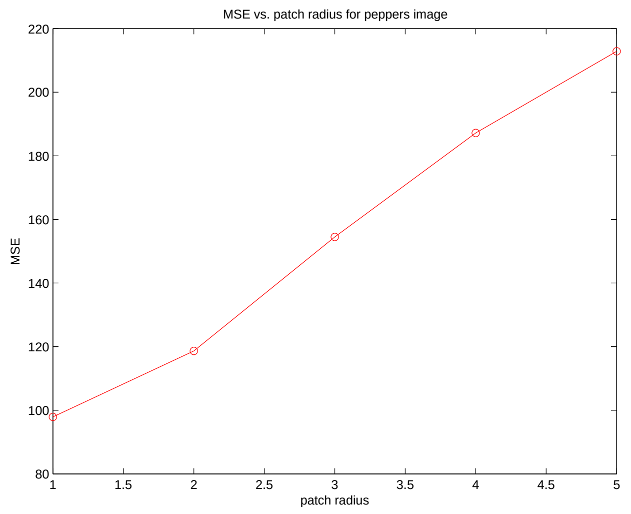

Figure 2: The optimal MSE occurs when the patch radius is 1.

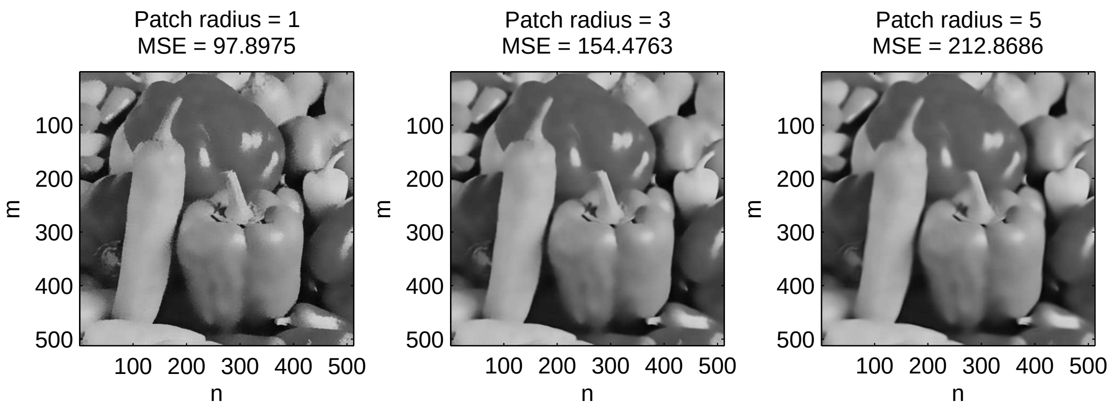

Figure 3: Although the optimal MSE occurs when the patch radius is 1, it does not necessarily produce the most visually appealing result.

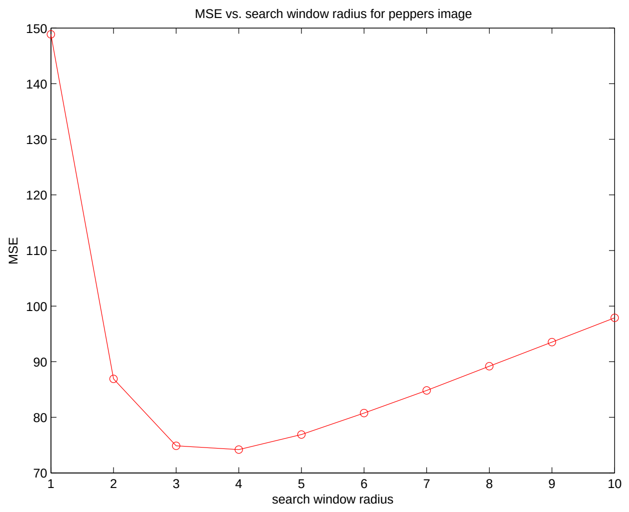

Figure 4: The optimal MSE occurs when the search radius is 4.

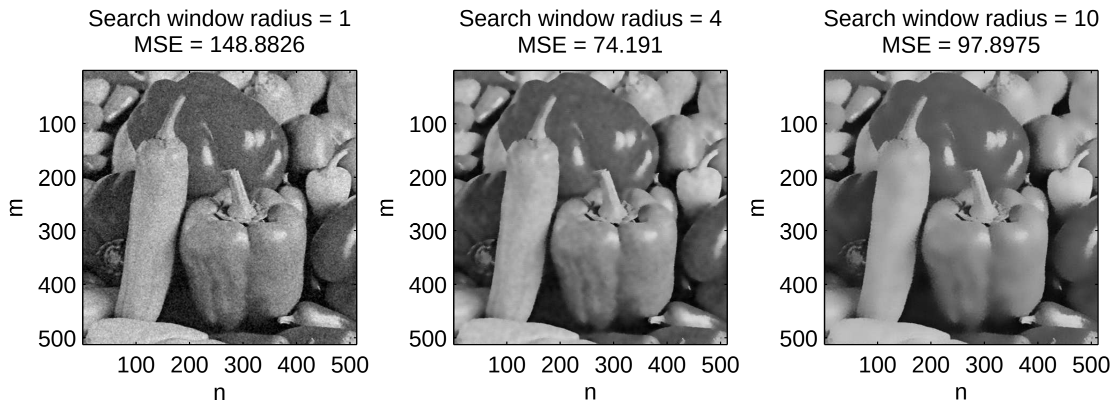

Figure 5: When the search radius is 4, the algorithm is able to remove noise without too much smoothing.

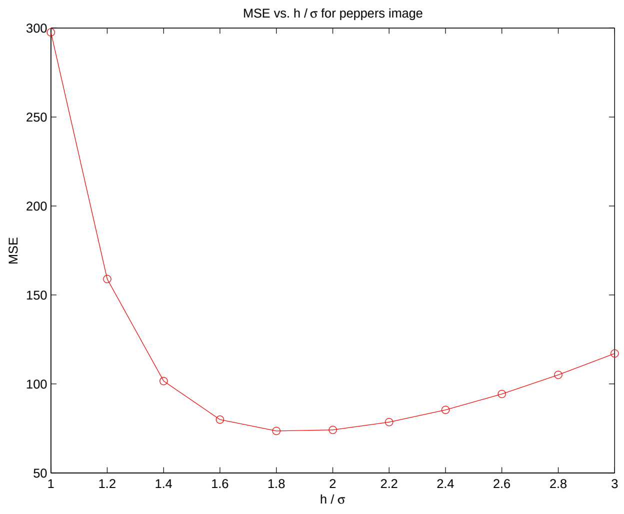

Figure 6: The optimal MSE occurs when the search radius is 4.

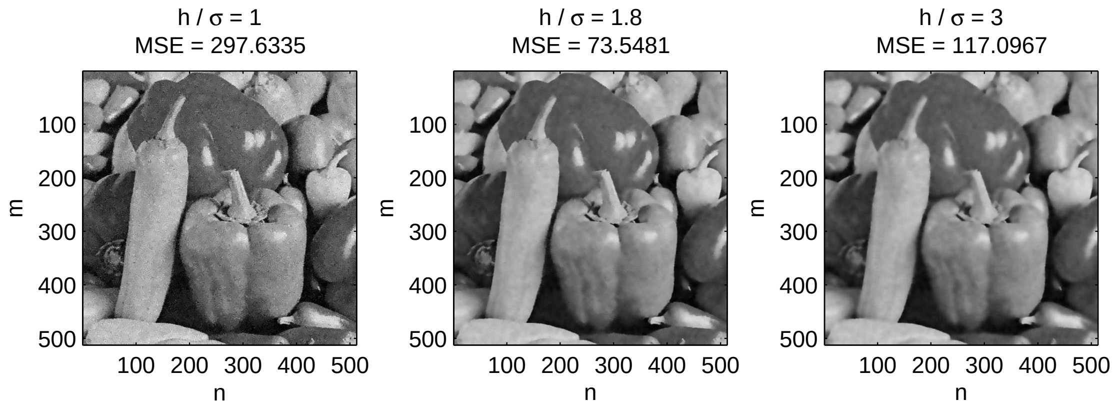

Figure 7: When the search radius is 4, the algorithm is able to remove noise without too much smoothing.

Image number	NL-means MSE	NPLS MSE
1	111.42	288.39
2	44.07	64.21
3	187.32	224.59
4	87.59	81.25
5	131.89	156.76
6	111.01	133.70

Table 1: Comparison of NL-means and NPLS for patterned images.

total variation (TV) denoising method used for comparison in [2]. This algorithm seeks to minimize a cost function with a maximum likelihood term as well as a non-quadratic penalty on the difference between the intensities of neighboring pixels. Since only the differences between neighbors are penalized, this can be viewed as a local filter. We can describe the algorithm mathematically as

$$\hat{f}[n,m] = \arg\min_{f[n,m]} \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \frac{1}{2}|f[n,m]-g[n,m]|^2 + \sum_{n=1}^{N-1}\sum_{m=0}^{M-1} \frac{\beta}{2}\psi(f[n,m]-f[n-1,m]) + \sum_{n=0}^{N-1}\sum_{m=1}^{M-1} \frac{\beta}{2}\psi(f[n,m]-f[n,m-1])$$

There are many choices of the potential function. $\psi(t) = |t|$ corresponds to a TV penalty. The potential function we use is the Lange3 potential $\psi(t) = \delta^2\left(\left|\frac{t}{\delta}\right| - \log\left(1 + \left|\frac{t}{\delta}\right|\right)\right)$. This cost function is minimized using a separable paraboloidal surrogates algorithm. There are several parameters to be chosen for this algorithm. We adjust the parameters following the same procedure as was used for non-local means. We find that the MSE does not appreciably improve after 300 iterations. We found the MSE to be lowest for $\beta \approx 16$ and $\delta \approx 1$. We use these parameters to compare the non-local means algorithm and the NPLS algorithm.

We compared the performance of the NL-means algorithm with that of the NPLS algorithm for a test set of 20 images containing both natural images as well as images with repeating patterns. Tables 1 and 2 given below give the MSEs of both the algorithms. We can see that in 5 out of 6 pattern images, the NL-means algorithm's performance is better than that of the NPLS. And for the 14 natural images, NL-means has lower MSEs in all cases. Thus, based on the difference between the MSEs in Tables 1 and 2, we can say that in most cases the NL-means algorithm performs slightly better than the NPLS algorithm for natural images, whereas it outperforms the NPLS for most images with repeating patterns.

3 Efficient Variations of the NL-means Algorithm

Many extensions to the non-local means algorithm have been introduced in the literature. Some of these focus on improving the computational efficiency of the algorithm. Others focus on producing higher quality images than the standard non-local means algorithm.

The computational complexity of the basic non-local means algorithm increases as we process images of larger sizes or if we increase the size of the search window within the image. This is because we must compute the L_2 norms for patches centered at each pixel against patches centered at all other pixels in the image. For faster computation, we need to reduce the computation of L_2 norms, or in other words, we need

Image number	NL-means MSE	NPLS MSE
1	47.76	53.86
2	46.18	51.75
3	86.61	97.87
4	121.02	124.84
5	66.14	1.94
6	101.70	120.98
7	89.53	121.16
8	49.36	52.77
9	175.83	190.90
10	96.28	111.11
11	138.15	178.37
12	82.40	95.38
13	160.20	226.09
14	83.25	101.42

Table 2: Comparison of NL-means and NPLS for natural images.

to consider the patches which have the lowest L_2 norms (and consequently higher weights). To eliminate the subset of patches with the highest L_2 norms, in [3], Orchard et. al. propose an SVD-based technique. The reason for using SVD is that it compresses the energy into the first few coordinates and represents the subspaces of a matrix in decreasing order of importance. Figure 8 shows the singular vectors for the bricks image (shown in figure 13), for a patch size of 5×5.

The trick is to approximate the L_2 norm of the patches by using the coordinates with respect to the basis of singular vectors. We consider every pixel $[i, j]$ of an image of size $N \times N$ and the patch centered at that pixel $[i, j]$. The patch is vectorized and stacked as a row of a matrix M. If the patch is of size $P \times P$, then the vector is of size $P^2 \times 1$ and the matrix M is of size $N^2 \times P^2$. We compute the SVD of the patch. The collection of the right singular vectors V is a suitable basis for the comparison of the coordinates. Since $M = USV^T$, multiplying M by V gives the coordinates. We proceed by computing the difference between the first coordinate which corresponds to the pixel $[i, j]$ and the first coordinates of all the other pixels. We eliminate the subset of the patches with the highest approximate L_2 norms. We then add the contribution from the second coordinate and select the patches with the lowest approximate L_2 norms. In our project, we used the contribution from the first three coordinates and we found the L_2 norms computed using the first three coordinates are a good approximation of the actual L_2 norms between patches (see figure 9). Though the MSE is slightly higher compared to the actual NL-means algorithm, the computation time reduces to a great extent (see figures 10 and 11). As an example, the computation time for the SVD based NL-means technique for the true image in figure 10 is around 3 seconds, whereas the computation time for the actual NL-means algorithm is around 15 seconds.

4 Adaptive Variations of the NL-means Algorithm

While the SVD variations on the NL-means algorithms concentrated on improving the computational speed of the NL-means algorithm, the adaptive algorithms focus on obtaining a better denoising performance from the images. We report our study of two such adaptive variations on the NLM algorithm.

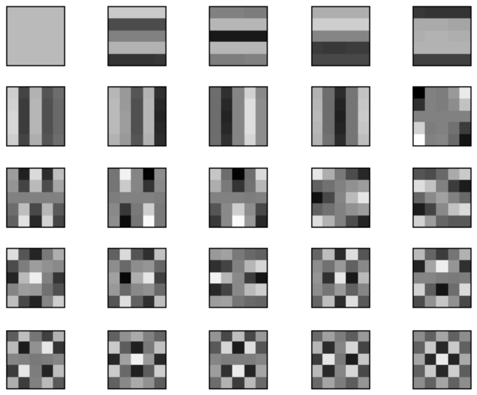

Figure 8: The singular values obtained by taking the SVD of all patches of the "bricks" image.

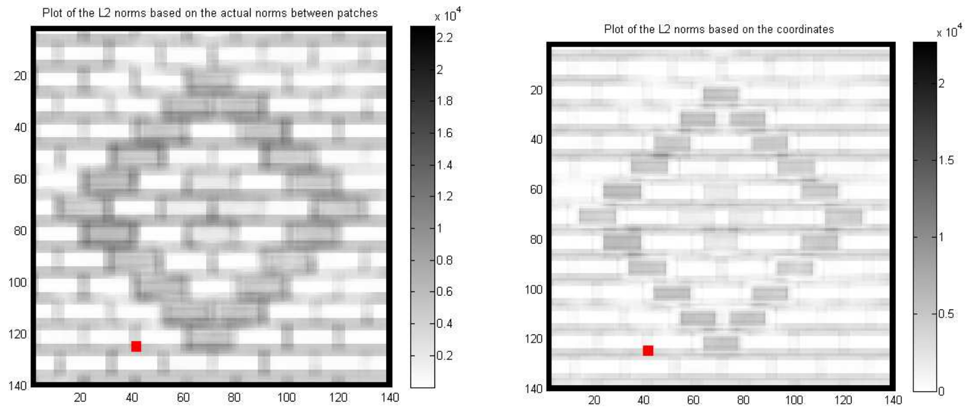

Figure 9: Comparison of the actual and approximated L_2 norms for the "bricks" image.

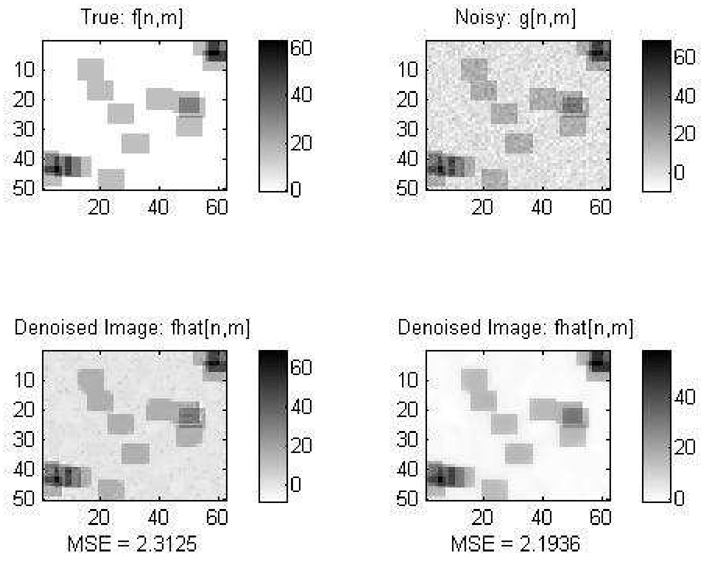

Figure 10: *Top row:* Original and noisy images. *Bottom row:* Outputs of the SVD-based non-local means and the NPLS algorithms.

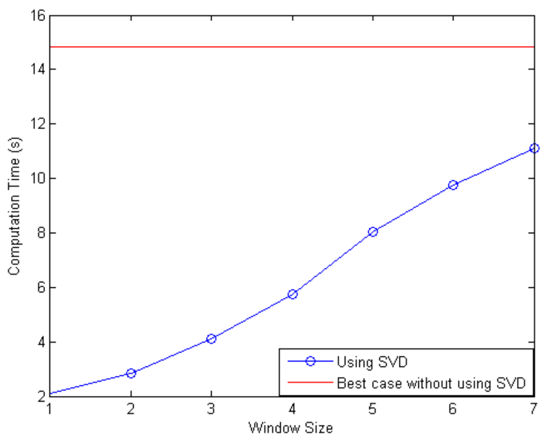

Figure 11: Comparison of the computation times for the SVD-based and best-case basic non-local means algorithm.

4.1 Motivation for adaptive algorithms

The NL-means algorithm exploits the structure inherent in the image by using the similarity in patches around pixels. In the basic version of the algorithm, the search window size, the patch size and the filtering parameter h are kept constant. This will lead to the same measure of averaging irrespective of whether the pixel is in a smooth region of the image or an edge pixel. Intuitively, we can guess that the performance of NL-means can be improved if smooth regions of the image are averaged more, and the edge regions less. This can be achieved by changing the parameters adaptively, according to the location of the pixel in the image.

4.2 Adaptive NL-means Algorithm

This version of the adaptive NL-means algorithm was proposed by [4]. In this method, a patch may belong to one of several different classes. Each class has associated with it a window size and a patch size, depending on whether the class corresponds to an edge pixel or a smooth pixel. Given a pixel, the SVD of the patch around the pixel is computed. The principal component of the SVD indicates the directionality of the gradient of the patch, and the corresponding singular value gives the degree of directionality (see figure 12). For a smooth image, there is no dominant direction and all the computed singular values are small. For an edge, there will be a dominant singular value. Given a spatial patch of size $N = n \times n$, we can group the gradient values into an $N \times 2$ matrix G, one column each for the horizontal and vertical gradients, so that

$$G = [\nabla f(1)^T \nabla f(2)^T \dots \nabla f(N)^T]^T$$

where

$$\nabla f(i) = \begin{bmatrix} \frac{\partial f(i)}{\partial x} & \frac{\partial f(i)}{\partial y} \end{bmatrix}$$

is the gradient of the image f at point i and G has the SVD $G = USV^T$, . Here V is a 2×2 matrix that gives the direction of the dominant field. Once the SVD has been computed, the dominant singular value has to be classified into one of K classes (our implementation used three classes - one for smooth, edge and one in between). This is done by using K-means clustering. The K-means algorithm classifies into one of K classes $C = [c_1, c_2 \dots c_k]$ by minimizing the sum of squares within cluster as

$$\underset{c}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{s(i) \epsilon c_k} |s(i) - \mu_k|^2.$$

where μ_k is the mean of c_k (see figure 13). According to the clustering, we used a patch size of 5×5 for smooth regions, and of 3×3 for the medium and edge patches. The reference [4] also recommends using rotated patches. The traditional NL-means can detect similar patches only if they have the same orientation. Therefore, a distant edge with a similar patch but different orientation cannot be detected. In order to obtain rotated matching between the patch, we consider the four dominant angles that the patch maybe oriented at. If $v = \begin{bmatrix} \nu_1 & \nu_2 \end{bmatrix}^T$ is the first column of the right singular matrix V, then the orientation of the gradient field can be calculated as

$$\theta = \arctan\left(\frac{\nu_1}{\nu_2}\right)$$

Since this does not have information about orientation, we try to match the patch with the four possible orientations at θ, $\theta+\pi$, $\theta-\pi$, and $-\theta$. Bicubic interpolation was used after block rotation. Upon implementing

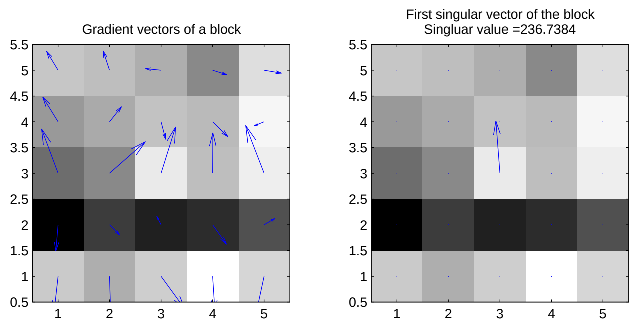

Figure 12: A block of the noisy "bricks" image and its gradient vectors (*left*) and the singular vector of the gradient block (*right*).

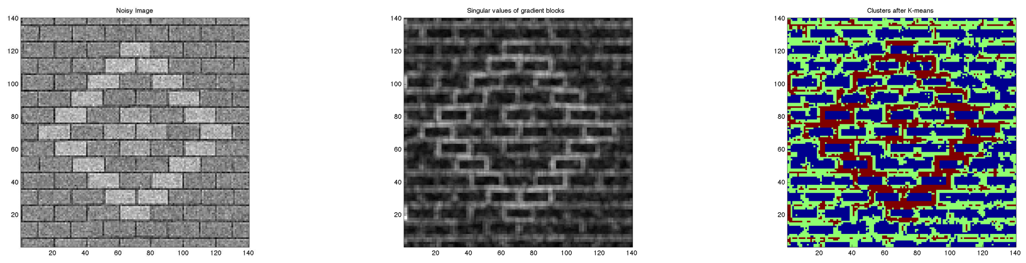

Figure 13: *Left to right:* The noisy "bricks" image, the singular values of the gradient vectors, and the clustered output of K-means.

the rotation in MATLAB, we found that the computation time was too high and no change in the MSE was observed from the adaptive NL-means we implemented without rotation. We believe this could be improved upon tweaking the parameters appropriately, but unfortunately, we faced a paucity of time to investigate this further. We present the results for the adaptive NL-means (implemented with SVD and K-means clustering, but without patch rotation) and the original image. We observe that the adaptive NL-means has a slightly higher MSE while visually indistinguishable from the traditional NL-means output. We believe this could have been improved by appropriately tweaking the parameters of the window size and patch size corresponding to the classes.

4.3 Anisotropic NL-means Algorithm

We also implemented another version of an adaptive NL-means algorithm proposed by [5]. So far, in the adaptive techniques considered, the weighting function has been isotropic. The central idea proposed in [5] is to adapt the weighting function in an anisotropic manner, based on characteristics of the underlying image, to improve the perceptual output. Each point $\underline{x} = (x, y)$ in the image is associated with an objective function that calculates its perceptual significance. This is defined based on the differential of the Gaussian

(also referred to as the Mexican Hat wavelet function)

$$\kappa(\underline{x}) = \frac{\sqrt{(\varphi_x(\underline{x}) * I_n(\underline{x}))^2 + (\varphi_y(\underline{x}) * I_n(\underline{x}))^2}}{\kappa_{max}}$$

where $I_n(\underline{x})$ is the noisy image, κ_{max} is the maximum value of κ for the image, and

$$\varphi_x(x) = \frac{1}{\sqrt{2\pi\sigma_s^3}} x e^{\frac{-x^2}{2\sigma_s^2}}$$

and $\varphi_y = \varphi_x^T$ and σ is the spatial spread of the function. Based on this value, we adapt the weighting parameters to achieve better perceptual quality. In smooth regions of the image, σ must be high enough to obtain high averaging and a smooth image, whereas in regions with high perceptual significance, σ must be low to preserve the details in the image. We can also determine the orientation of the corresponding image feature as

$$\theta(\underline{x}) = \arctan\frac{(\varphi_y(\underline{x}) * I_n(\underline{x}))}{(\varphi_x(\underline{x}) * I_n(\underline{x}))}$$

Using the values of $\kappa(\underline{x})$ and $\theta(\underline{x})$ obtained above, we can change the shape and orientation of the weighting function to better fit the image characteristics. In [5], the authors propose the following weighting function

$$w(\underline{l}, \underline{m}) = \frac{1}{Z(\underline{l})} e^{-\frac{(\psi(\kappa,\theta)||I_n(\varsigma_l) - I_n(\varsigma_m)||_2^2)}{h^2}} \tag{1}$$

Where ς_l and ς_m represent the local neighborhoods around $\underline{l}$ and $\underline{m}$ respectively, and ς represents a neighborhood of the same size as ς_l and ς_m, and Z($\underline{\text{l}}$) is a normalizing constant. $\psi(\kappa, \theta)$ is the factor that changes the weighting function anisotropically, and is given by

$$\psi(\kappa, \theta) = \frac{1}{2\sigma_1(\kappa)\sigma_2(\kappa)} e^{-\beta}, \tag{2}$$

where

$$\beta = \frac{x_1}{2\sigma_1(\kappa)^2} + \frac{y_1}{2\sigma_2(\kappa)^2}. \tag{3}$$

Here x_1 and y_1 are the x and y coordinates rotated counterclockwise by an angle $\theta + \pi$. This aligns the weighting function along the edge in the image. Now, we can spread the function along the edge, and decrease the spread of the function across the edge, so as to preserve edge details. Since κ gives us an idea of the magnitude of the perceptual significance, we would like to vary the width of the weighting function as a function of κ. This is done by making $\sigma_1(\kappa)$ and $\sigma_2(\kappa)$ functions of κ. They can be adjusted between a minimum and maximum limit as follows

$$\sigma_1(\kappa) = \sigma_{1,min} + \alpha\kappa(\sigma_{1,max} - \sigma_{1,min})$$

and

$$\sigma_2(\kappa) = \sigma_{2,min} + \kappa(\sigma_{2,max} - \sigma_{2,min})$$

Here $\sigma_{n,max}$ and $\sigma_{n,min}$ are the maximum and minimum values of σ_n respectively and $\alpha \geq 1$ is a shrinkage factor. The motivation for the expressions given for β in equation (3) was unclear, since it was

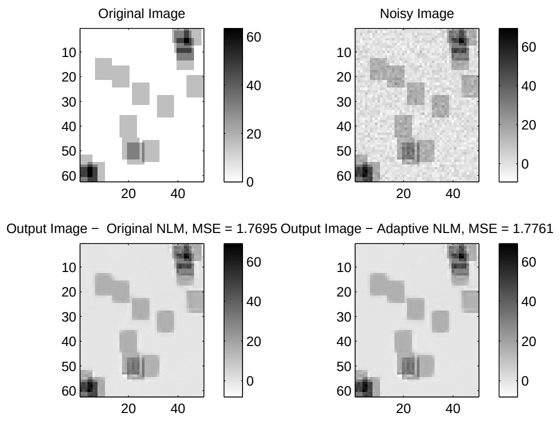

Figure 14: The output of the anisotropic NL-means.

our understanding that the weighting kernel must decay as it goes away from the pixel under consideration, whereas the equation given would decay as distance increases in one direction away from the edge and grow exponentially in the other. Further, in order to guarantee decay, β must increase as one moves away from the edge in both directions. Therefore, we modified the expressions given in the paper for implementation purposes, while trying to maintain the spirit of the idea presented in the paper. Accordingly, the expressions we used were

$$\beta = \frac{x_1^2}{2\sigma_1(\kappa)^2} + \frac{y_1^2}{2\sigma_2(\kappa)^2}$$

and

$$w(\underline{l}, \underline{m}) = \frac{1}{Z(\underline{l})} e^{\psi(\kappa,\theta)} e^{-\frac{(||I_n(\varsigma_l) - I_n(\varsigma_m)||_2^2)}{h^2}}$$

instead of equations (3) and (1) respectively. We implemented the algorithm in Matlab, and the results are shown in figure 14. We notice almost equivalent performance after implementing our version of the adaptive algorithm and a slight increase in the MSE compared to the traditional NL-means algorithm. We do not have a satisfactory explanation for this behavior, since it is possible that our understanding of the paper was not complete, especially since the paper reported an improvement in performance.

5 Conclusion

We investigated the NL-means algorithm as proposed in [1, 2]. We implemented the basic algorithm in Matlab and tweaked the patch size, search window size, and the filtering parameter h to optimize performance in terms of the MSE. We then compared this performance to a optimized version of NPLS and found that the NL-means algorithm performs slightly better than the NPLS algorithm for a testbed of 20 images. The improvement was more pronounced in patterned images.

Additionally, we implemented the SVD-based NL-means algorithm in order to reduce the computation time. We were able to reduce the computation time compared with the basic NL-means algorithm, but this compromised the image quality. We also implemented two adaptive variations of the non-local means algorithm. The first used the SVD of the gradient blocks to classify the image based on roughness. This information was used to adjust the patch radius and search window radius. The second used a filtered gradient to measure the strengths and orientation of edges. The weighting function is rotated and shaped to align with edges. We found that our implementations of the adaptive algorithms performed slightly worse than the basic non-local means algorithm. Both adaptive algorithms involve more parameters than the basic non-local means algorithm, so the poor performance may be due to non-optimal parameters.

Acknowledgements. We would like to thank Prof. Fessler for his guidance throughout the duration of the project.

References

[1] A. Buades, B. 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 vol. 2, June 2005.

[2] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[3] J. Orchard, M. Ebrahimi, and A. Wong. Efficient nonlocal-means denoising using the svd. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1732 –1735, Oct. 2008.

[4] T. Thaipanich, Byung Tae Oh, Ping-Hao Wu, Daru Xu, and C.-C.J. Kuo. Improved image denoising with adaptive nonlocal means (anl-means) algorithm. *Consumer Electronics, IEEE Transactions on*, 56(4):2623 –2630, Nov. 2010.

[5] A. Wong, P. Fieguth, and D. Clausi. A perceptually adaptive approach to image denoising using anisotropic non-local means. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 537 –540, Oct. 2008.