Name:- Saurabh Singhal
Roll No:- 171CO240

DIP Assignment

Problem:- Implement at least 3 different versions of NLM method. Compare the methods on simulated noisy images. Create noisy images by adding Gaussian noise of standard deviations 5, 10, 15, 20 and 25. Use PSNR as a similarity metric. Assignments should contain a brief description of each method, Experimental results and analysis and Conclusion.

Solution:

Denoising Image:

The goal of image denoising methods is to recover the original image from a noisy measurement,

$$v(i) = u(i) + n(i)$$

where v(i)is the observed value, u(i)is the "true" value and n(i) is the noise perturbation at a pixel i. The best simple way to model the effect of noise on a digital image is to add a gaussian white noise.

I will show the NLM implementation by 3 following ways:-

1. **Probabilistic Non-Local Means (PNLM)**
2. **Non-Local Means Filter(NLmeansfilter)**
3. **Fast Non-Local Mean Image Denoising Implementation**

Probabilistic Non-Local Means:

Calculation of NLM weights is the most computationally expensive part of the algorithm and is related to many parameter optimization schemes. It has long been noticed that the NLM weight function is somewhat inadequate because it tends to give non-zero weights to dissimilar patches . NLM weight function gives two equally probable D l,k very different weights and that it fails to give the largest weight to the most probable case. For D l,k close to its expected value, the weight errors are not too large because the corresponding region in the exponential curve is almost linear with a moderate slope. However, for D l,k far away from its expected value ,weight errors are very large because the NLM function tends to give nonzero weights to these highly improbable cases, so weight errors grow quickly.
Instead of including the exponential function in weighting pixels, they propose the following probabilistic weight

$$w_{l,k} = f_{l,k}(\widehat{D}_{l,k}/\rho^2)$$

where f l,k (·) is the theoretical probability density function (p.d.f.) of the r.v. D l,k , D l,k is the estimated distance with estimated variance σ 2 , and ρ is a tuning parameter. This weight function can be interpreted as the probability of seeing a noisy patch difference when two clean patches match perfectly. Since it is clear we shall give a smaller weight in the more typical case when this perfectly matching condition fails, then gives the largest similarity weight we shall consider.
The p.d.f. of D l,k is

$$f_{l,k}(D) = \chi^2_{\eta_{l,k}}(D/\gamma_{l,k}) = \frac{(D/\gamma_{l,k})^{\eta_{l,k}/2-1}\exp(-D/2\gamma_{l,k})}{2^{\eta_{l,k}/2}\Gamma(\eta_{l,k}/2)}$$

where parameters γ l,k and η l,k can be determined by the first two cumulants of D l,k as shown below.

$$\gamma_{l,k} = \operatorname{var}[D_{l,k}]/\,(2\mathrm{E}[D_{l,k}])$$
$$\eta_{l,k} = \mathrm{E}[D_{l,k}]/\gamma_{l,k}$$

The cumulant E[D l,k] is straightforward to find, and it is

$$\mathrm{E}[D_{l,k}] = \sum_{j\in\P} \mathrm{E}[d_{l+j,k+j}] = |\P|.$$

With regards to var[D l,k] , the following identity always holds

$$\operatorname{var}[D_{l,k}] = \sum_{i,j\in\P} \operatorname{cov}[d_{l+i,k+i}, d_{l+j,k+j}]$$

where the covariance can be written as follows.

$$\operatorname{cov}[d_{l+i,k+i}, d_{l+j,k+j}] = \mathrm{E}[d_{l+i,k+i}d_{l+j,k+j}] - \mu^2_{d_{l,k}}$$

Non-Local Means Filter:

Given a discrete noisy image v = {v(i) | i ∈ I}, the estimated value NL[v](i), for a pixel i, is computed as a weighted average of all the pixels in the image,

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

where the family of weights {w(i, j)}j depend on the similarity between the pixels i and j, and satisfy the usual conditions 0 ≤ w(i, j) ≤ 1 and ∑w(i, j)=1.
The similarity between two pixels i and j depends on the similarity of the intensity gray level vectors v(Ni) and v(Nj), where Nk denotes a square neighborhood of fixed size and centered at a pixel k. This similarity is measured as a decreasing function of the weighted Euclidean distance. The application of the Euclidean distance to the noisy neighborhoods raises the following equality

$$E||v(\mathcal{N}_i) - v(\mathcal{N}_j)||_{2,a}^2 = ||u(\mathcal{N}_i) - u(\mathcal{N}_j)||_{2,a}^2 + 2\sigma^2.$$

This equality shows the robustness of the algorithm since in expectation the Euclidean distance conserves the order of similarity between pixels.
The pixels with a similar grey level neighborhood to v(Ni) have larger weights in the average. These weights are defined as,

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||_{2,a}^2}{h^2}},$$

where Z(i) is the normalizing constant

$$Z(i) = \sum_j e^{-\frac{||v(\mathcal{N}_i) - v(\mathcal{N}_j)||_{2,a}^2}{h^2}}$$

and the parameter h acts as a degree of filtering. It controls the decay of the exponential function and therefore the decay of the weights as a function of the Euclidean distances.

Fast Non-Local Mean Image Denoising Implementation:
Here,they presented an efficient algorithm for nonlocal image filtering with applications in electron cryomicroscopy.It builds on the separable property of neighborhood filtering to offer a fast parallel and vectorized implementation in contemporary shared memory computer architectures while reducing the theoretical computational complexity of the original filter. In practice, this approach is much faster than a serial, non–vectorized implementation and it scales linearly with image size.

Normal NLM Approach(Not Fast):

$$u(s) = \frac{1}{Z(s)} \sum_{t \in \mathcal{N}(s)} w(s,t) v(t) \ , \qquad (1)$$

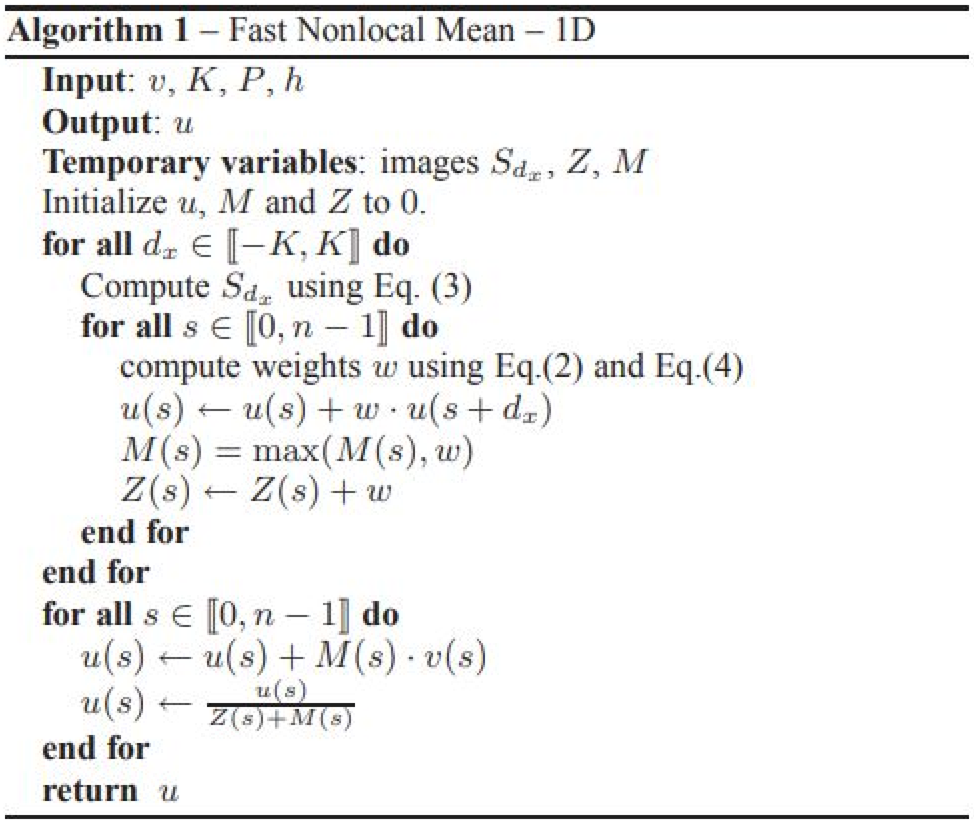
$$w(s,t) = g_h \left(\sum_{\delta \in \Delta} G_\sigma(\delta) \left(v(s+\delta) - v(t+\delta) \right)^2 \right) \ , \qquad (2)$$

The weight computation is by far the most time consuming part when generating the restored image u.Updated equations so that weights can be calculated in constant time are as follows:

$$S_{d_x}(p) = \sum_{k=0}^{p} \left(v(k) - v(k+d_x) \right)^2 , \ \ p \in \Omega. \qquad (3)$$

$$w(s,t) = g_h \left(S_{d_x}(s+P) - S_{d_x}(s-P) \right) \qquad (4)$$

Fast Algorithm for Denoising Image:



```
Algorithm 1 – Fast Nonlocal Mean – 1D
  Input: $v, K, P, h$
  Output: $u$
  Temporary variables: images $S_{d_x}$, $Z$, $M$
  Initialize $u$, $M$ and $Z$ to 0.
  for all $d_x \in [\![-K, K]\!]$ do
    Compute $S_{d_x}$ using Eq. (3)
    for all $s \in [\![0, n-1]\!]$ do
      compute weights $w$ using Eq.(2) and Eq.(4)
      $u(s) \leftarrow u(s) + w \cdot u(s + d_x)$
      $M(s) = \max(M(s), w)$
      $Z(s) \leftarrow Z(s) + w$
    end for
  end for
  for all $s \in [\![0, n-1]\!]$ do
    $u(s) \leftarrow u(s) + M(s) \cdot v(s)$
    $u(s) \leftarrow \frac{u(s)}{Z(s)+M(s)}$
  end for
  return $u$
```

Results Comparison

I created noisy images by adding Gaussian Noise of standard deviations 5,10,15,20,25.

PSNR values of noisy image with different standard deviations:

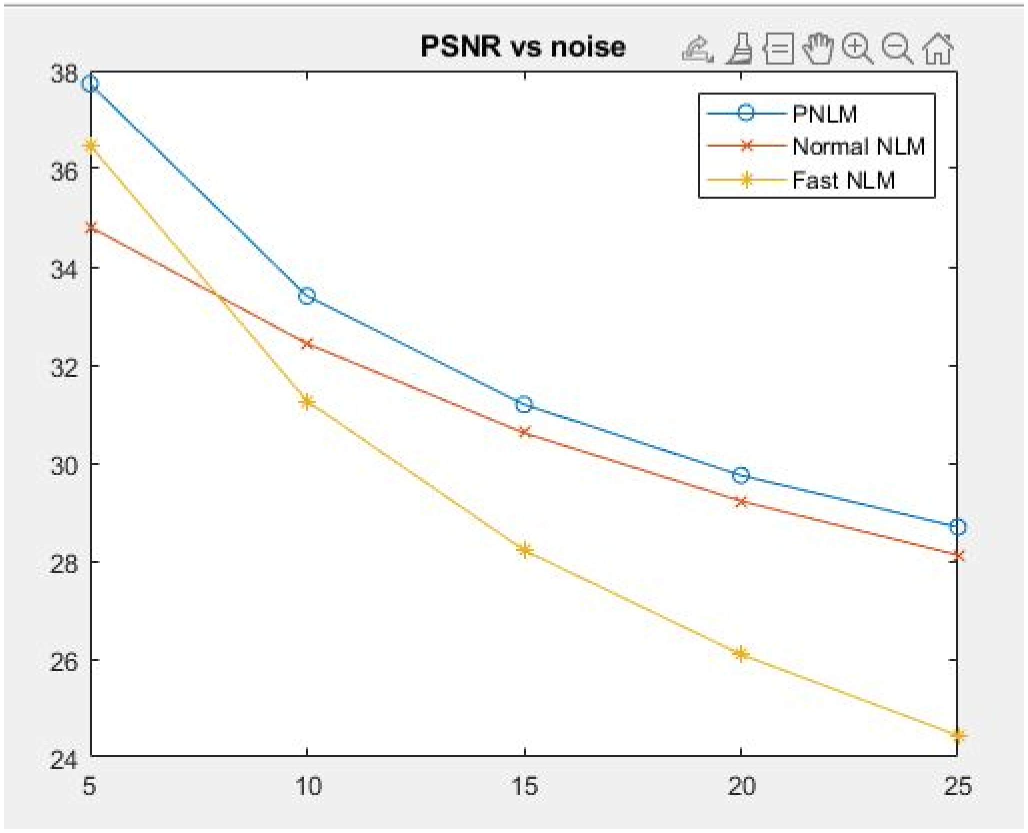
Standard Deviation	PSNR
5	34.1829
10	28.0990
15	24.6320
20	22.1374
25	20.1188

PSNR values of a denoised image after applying different versions of NLM:

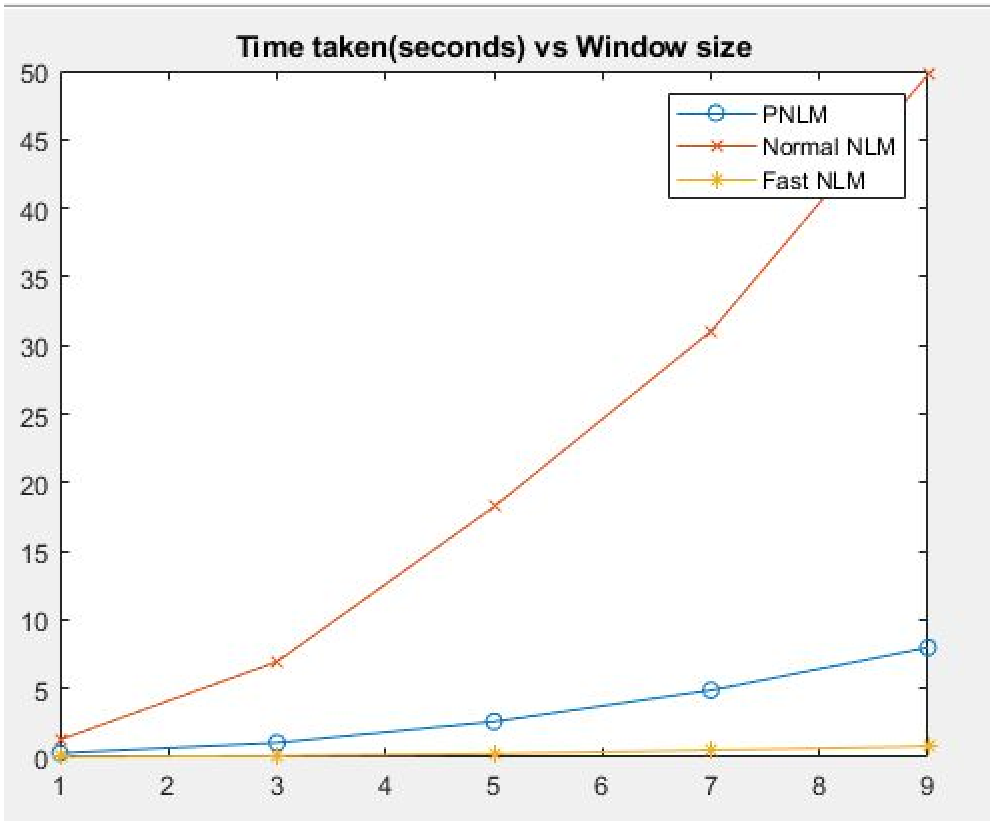
Standard Deviation	**PNLM**	**NLmeansfilter**	**FAST_NLM_II**
5	37.7303	36.4727	34.8132
10	32.4323	33.4022	31.2649
15	28.2196	30.6126	31.1950
20	29.7477	26.0917	29.2235
25	28.1199	28.6933	24.4382

Graph comparison:

1) Standard Deviation on X-axis vs PSNR on Y-Axis



2) Window Size(X-axis) vs Time Taken(Y-axis)



Analysis:

1) Execution Time

   Fast NLM<PNLM<Normal NLM

2) Similarity between Denoised image and Ground Truth(PSNR)

   Fast NLM<Normal NLM<PNLM

Conclusions:

1) Therefore, we can say that PNLM is best among the three versions because it gives the highest PSNR values as well as low execution time.
2) With increasing value of Window Size, execution time of NLmeansfilter increases very rapidly as compared to others.
3) Fast NLM is fastest among all the three but performs even worse than NLmeansfilter in case of highly noisy images.

References:
https://in.mathworks.com/matlabcentral/fileexchange/13176-non-local-means-filter
https://in.mathworks.com/matlabcentral/fileexchange/41390-probabilistic-non-local-means

https://in.mathworks.com/matlabcentral/fileexchange/38200-fast-non-local-mean-image-denoisin
g-implementation