SSIM Based Image Denoising

Group no 5

Members:

Ankit Rai
181EE109

Bhuvan Balaji J
181EE215

Sachin Pandey
181EE237

Saurabh Madhan
181EE142

Vishesh Kodwani
181EE155

Abstract of the work:

Image denoising is a critical part of image processing. We do the process to estimate the original image required by trying to suppress or remove noise which leads to deteriorated visual quality from a noise-contaminated version of the picture. There are several methods/algorithms to reduce noise from a signal (an image in this case). In this case, we try to incorporate the **Structural Similarity Index Measure SSIM** [1] into the framework of **Non-Local Means denoising** [2]**.** SSIM is used for measuring the similarity between two images. In our approach, we use SSIM to determine the similarity between the patches and their weights to denoise specific patches of the image by weighted averaging of neighbouring patches and finally use NLM algorithm [3], which takes a mean of all the pixels present in the given image and weigh it by how similar it is to the target patch/pixel to denoise the given image. We chose this method as it is better than the Local means algorithm as it comparatively has better clarity after denoising. Reduction of loss of detail also happens after denoising the image.

Base Paper: Rehman, Abdul & Wang, Zhou. (2011). **SSIM-BASED NON-LOCAL MEANS IMAGE DENOISING**. IEEE International Conference on Image Processing (ICIP). 217-220. 10.1109/ICIP.2011.6116065.

Explanation Video

---

1. Introduction:

Non-Local Means Denoising is one of the most widely used and successful methods used for image denoising. Using this technique, we assign weights depending upon the similarity of the patches in the image, we are currently trying to denoise and the remaining patches of the image within a boundary or a neighbourhood [4]. The conventional method uses MSE to calculate the weights, leading to a reduced perceptual quality of the denoised image, reducing the denoising method's effectiveness. Therefore, we prefer using SSIM in the NLM framework as it has the upper hand compared to MSE in measuring perceptual image quality.
Using SSIM comes with challenges; SSIM is used for measuring similarity between images that are already denoised. However, here we are comparing noisy patches of the image to assign weights. Hence using SSIM in the presence of noise will not yield good results while defining weights. The reason is, SSIM usually tones the structures of the two patches being compared. However, when the noise level is more significant than the actual signal, it distorts the image's original structure. Thus, the SSIM would match the noise pattern rather than the fundamental structure of the image patch.
Another obstacle is, SSIM will pick the patches with similar structures. However, the patches can have different contrast and mean values. When we perform weighted averages while running NLM, directly averaging these patches will result in further distortion, spoiling the whole process. To rectify it, once we finish assigning the weights, we need to modify the contrast and mean values of the image patches before performing the average of weights for the NLM process.

2. Theory:

2.1 NLM Algorithm:

2.1.1 Principle of NLM (Non-local Means) algorithm

In a noisy image, patches often appear repetitively. In the below picture, we can see three patches. Namely P, Y1, and Y2.

Here we can observe that the image patches P and Y1 have a similar kind of neighbourhood structure. Whereas the image patches P and Y2 are almost altogether the same. Therefore, while assigning the similarity weights, the weight $\boldsymbol{\omega}$ (P, Y1) will be greater than $\boldsymbol{\omega}$ (P, Y2) while filtering the image patch P. By performing this operation with all the image patches, it obtains the similarity weight of the image patch P.

2.1.2 Formula

The estimated value at any pixel is obtained by calculating the weighted average of all the pixels in the image. Still, the family of weights depends on the similarity between the pixels i and j, so similar pixels in the neighbourhood give larger weights.

Consider a pixel i, which has a grey value denoted by $v(i)$ and a filtered grey estimate represented by $NL(v)(i)$ which is calculated using the below formula, by obtaining a weighted average of the all the pixels having similar neighbourhoods present in the given noisy image

$$NL(v)(i) = \sum_{j \in I} \omega(i,j) v(j)$$

Here I denotes the whole image space for summation, $\omega(i,j)$ denotes the weight, i.e., the measure of pixel j 's influence on pixel i. We calculate it using the below mentioned formula.

$$\omega(i,j) = \frac{1}{C(i)} e^{\frac{\|v(N_i) - v(N_j)\|^2_{2,\alpha}}{h^2}}$$

Where N_i is the image block of the pixel to be restored (reference block).

N_j is a similar image block whose size is same as that of N_i.
$v(N_i)$ is the grey value of each pixel in reference block.
$v(N_j)$ is the pixel of each block in the similar block grey value.
$\|v(N_i) - v(N_j)\|^2_{2,\alpha}$ is the Gaussian weighted Euclidean distance.
α is the standard deviation of the Gaussian kernel.
h is the attenuation parameter that controls the degree of filtering.
The normalized parameter $C(i)$ is given by,

$$C(i) = \sum_{j \in I} e^{\frac{\|r(N_i) - v(N_j)\|^2_{2,\alpha}}{h^2}}$$

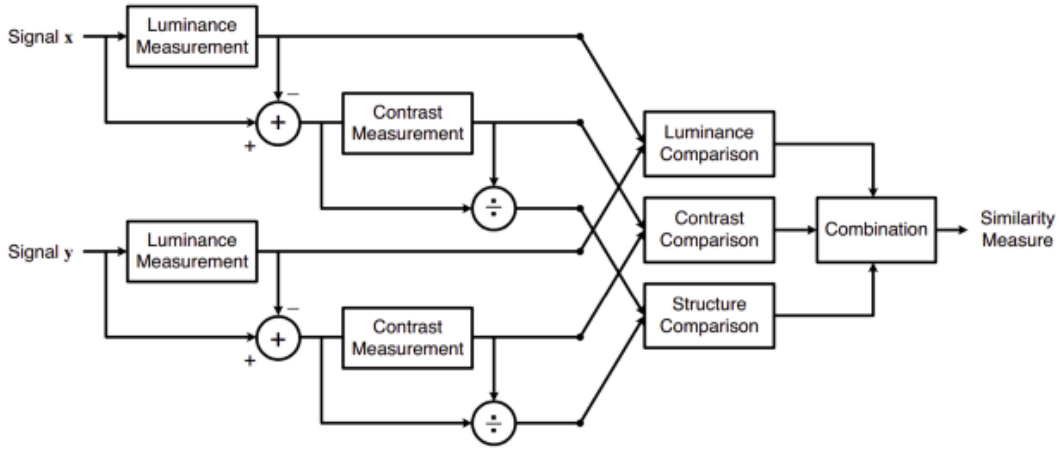
and $\omega(i,j)$ meets $\omega(i,j) \in [0,1]$ and $\sum_{j \in I} \omega(i,j) = 1$.

2.1.3 Shortcomings of NLM

Even though NLM is regarded to be one of the most efficient denoising techniques, it has a few shortcomings. NLM has difficulty in precisely establishing the weight of similarity between two image patches [5]. This is one of the reasons why we opted to perform denoising using SSIM in the framework of NLM.

Since the image contains many pixels, there are similarities between the image patches, and sometimes they tend to shift the values in a certain direction. To avoid this, we use steps to translate between the different patches in the image instead of using local means. The level of contrast between the image patches is not satisfactory enough. Since the image patch adopts a panning operation, the image's self-similarity cannot be completely utilized. Sometimes pixel patches with visible similarity are assigned lower weights compared to other patch combinations and hence it leads to a decrease in the overall image quality.

2.2 SSIM:



The above picture depicts the arrangement and flow of SSIM where Signal X refers to the Reference image, and Signal Y refers to the Sample image.

The above system tells us about the Structural similarity index of two images which lies between the range of -1 and 1, where one indicates that the two images are very similar to each other while -1 indicates that the two images highly differ.

In order to calculate SSIM, we need three main quantities. They are,

- **Luminance**
  Luminance is the value we get by averaging the pixel values of all the pixels present in the image chosen. It is denoted by μ (Mu) and the formula for calculating it is given below.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

  The luminance comparison function $l(\mathbf{x}, \mathbf{y})$ which is seen later, is a function of μ_x and μ_y.

- **Contrast**
  Contrast is obtained by calculating the standard deviation of the pixel values present in the image. The symbol designated to it is σ (sigma) and it can be found out by applying the formula below.

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N} (x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

  The contrast comparison function c(**x**, **y**) which is seen later, is the comparison of σ_x and σ_y

- **Structure**
  The structural comparison is obtained by dividing the pixel value with it's standard deviation, as shown in the below formula, where x is the input image.

$$(\mathbf{x} - \mu_x)/\sigma_x$$

Using these above three parameters, we now apply them into comparison functions that compares the input images based on the above-mentioned parameters. The formulae for these functions are listed below.

- **Luminance Comparison function:**

The inputs for the function are x and y, two images.

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

Here μ represents the image's mean. In order to get finite values, we add a constant to the denominator C_1, which is obtained by $C_1 = (K_1 L)^2$ where L represents the dynamic range of pixels and K_1 is just another constant.

- **Contrast comparison function:**
  Similarly, x and y are the input images

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Here σ represents the standard deviation of an image. In order to get finite values, we add a constant to the denominator C_2, which is obtained by $C_2 = (K_2 L)^2$ where L represents the dynamic range of pixels and K_2 is just another constant.

- **Structure comparison function**
  The formula for the function whose inputs are x and y is

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

In order to get finite values, we add a constant to the denominator C_3. Here σ represents the standard deviation of an image and the value of σ_{xy} is obtained by

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

Now we have all three comparison functions. Using these, we now calculate the SSIM index. It is obtained by the below formula

$$SSIM(\boldsymbol{x},\boldsymbol{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha}.\ [c(\mathbf{x},\mathbf{y})]^{\beta}.\ [s(\mathbf{x},\mathbf{y})]^{\gamma}$$

Here α, β, γ > 0 are present for us to tweak the index depending upon which parameter we want the index to be influenced by. On assuming α=β=γ=1, and $C_3 = C_2/2$, we finally obtain the equation for SSIM.

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

3. Implementation Methods:

To implement the above idea, we will use a two-stage process. We need these two stages because if we perform SSIM directly on the noisy image, the result will be a weight that will result in another noisy image when NLM is applied using code inspired from [6].

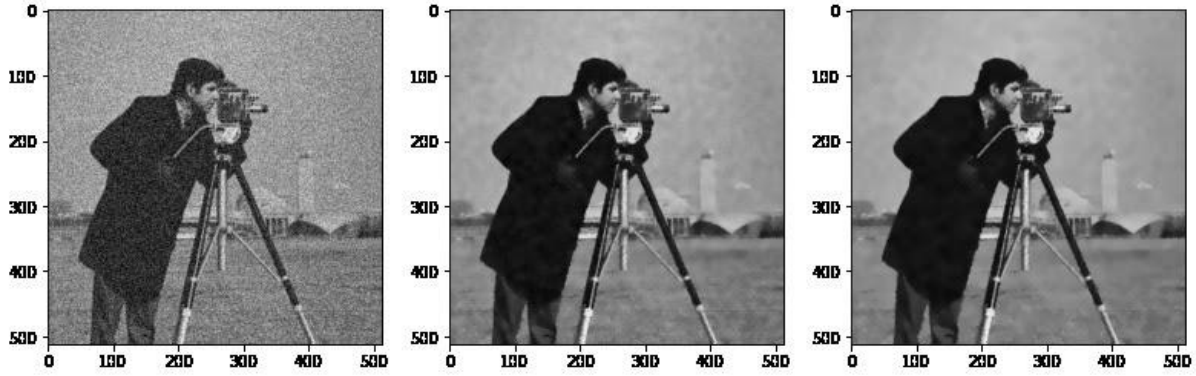
In the first stage, we apply NLM denoising on the noisy image to get a "Denoised Image." We will call it "denoised_img" for further use. This denoised_img will have its weights calculated using MSE, and we'll pass it to the second layer.

In the second layer, we will pre-process the denoised_img by adjusting its contrast and luminance as suggested in [7] before calculating the SSIM values. After pre-processing, we calculate the SSIM values and then again apply NLM Denoising on the processed denoised_img. This gives us our final image output, which is an improved result on the denoised_img, which was obtained using just NLM.

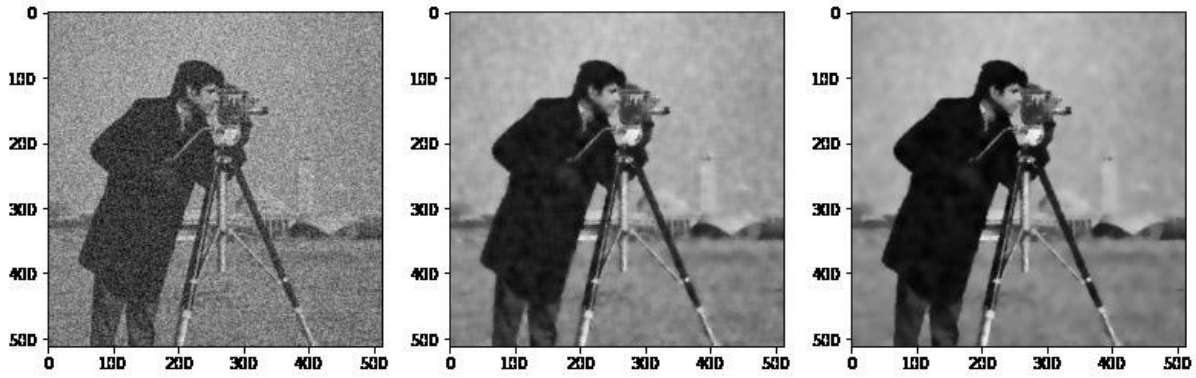
4. Results

In this project we use the peak signal to noise ratio PSNR [8] for evaluation of the image obtained after denoising. The below images show the results obtained by applying NLM with MSE (centre), NLM with SSIM (rightmost) on the noisy images (leftmost).

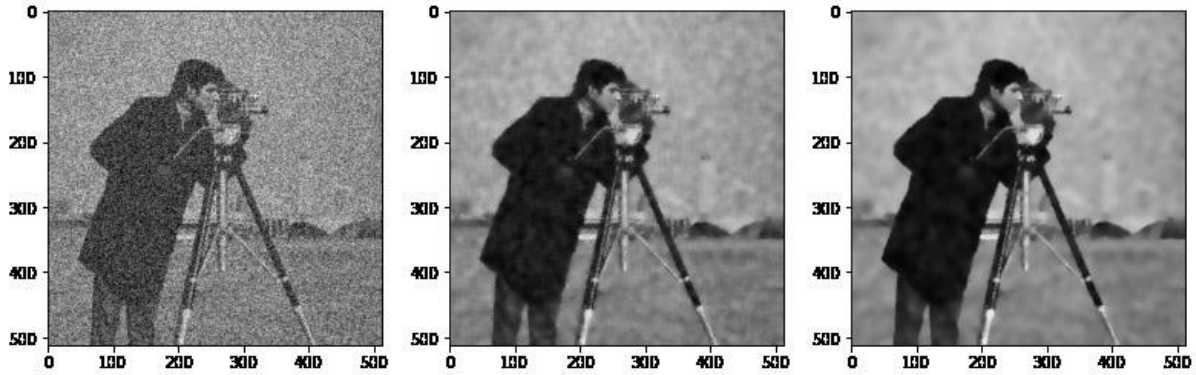
Sigma=0.2 or 20%



Sigma=0.3 or 30%



Sigma=0.4 or 40%



	Sigma values		
PSNR values for all this combinations	**Sigma = 0.2**	**Sigma = 0.3**	**Sigma = 0.4**
Noisy Image	13.98	10.46	7.96
NLM + MSE	27.53	25.19	23.88
NLM + SSIM	27.86	25.93	25.17

5. Contributions:

Ankit Rai: Contributed in making video, code and edited the report

Bhuvan Balaji: Gathered base papers for research and wrote the whole report after getting content

Sachin Pandey: Contributed in making video, code and edited the report

Saurabh Madhan: Gathered content for the report and researched about the implementation of the project

Vishesh Kodwani: Gathered base papers for research and wrote the whole report after getting content

6. Conclusions:

In this project, we have essentially tried to substitute the responsibility of MSE with SSIM in the framework of NLM. Here we are proposing a vigorous technique to overcome the difficulty of using SSIM by altering the mean and contrast of the image patches before calculating the weighted average for NLM. The results obtained above suggest that using SSIM when the noise level is high usually yields better results and hence it can be used as an alternative to MSE for denoising images.

References:

[1] Z. Wang, A. Bovik, H. Sheikh, and E.P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, April 2004.

[2] A. Buades, B. Coll, and J. M. Morel, “A non-local algorithm for image denoising,”in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2005.

[3] A. Buades, B. Coll, and J. M. Morel, “A review of image denoising algorithms, with a new one,” Multiscale Modelling & Simulation, 2005.

[4] Yan Nana, “Research on Degraded Image Restoration Technology Based on Non-local Mean”,Yanshan University, 2011.

[5] Baozhong Liu and Jianbin Liu, “Overview of image noise reduction based on non-local mean algorithm”, MATEC Web of Conferences, 2018.

[6] A. Buades, B. Coll, and J.M. Morel, “Non local means denoising implementation” http://ipol.im/pub/algo/bcm non local means denoising.

[7] V. Bruni, D. Panella, and D. Vitulano, “Non local means image denoising using noise-adaptive SSIM” in 23rd European Signal Processing Conference (EUSIPCO), 2015.

[8] Hore A, Ziou D. Image quality metrics: PSNR vs. SSIM, International Conference on Pattern Recognition. IEEE, 2010

Appendix:

```
import numpy as np
import matplotlib.pyplot as plt
import math
import random
import skimage as ski
from skimage import data
from skimage.util import random_noise
from skimage.transform import resize
from skimage import data, img_as_float
from skimage.metrics import mean_squared_error
from skimage.restoration import denoise_nl_means, estimate_sigma
from skimage.metrics import structural_similarity as ssim
from skimage.metrics import peak_signal_noise_ratio as psnr
orig_img = data.camera()
print("orig_img.shape: ", orig_img.shape)
print("orig_img.max: ", np.max(orig_img))
print("orig_img.min: ", np.min(orig_img))
plt.imshow(orig_img, cmap='gray')
plt.show()
sigma = 0.2
noise = np.ones_like(orig_img) * sigma * (orig_img.max() -
orig_img.min())
noise[np.random.random(size=noise.shape) > 0.5] *= -1
noise_img = orig_img + noise
print("noise_img.shape: ", noise_img.shape)
print("noise_img.max: ", np.max(noise_img))
print("noise_img.min: ", np.min(noise_img))
plt.imshow(noise_img, cmap='gray')
plt.show()

# estimate the noise standard deviation from the noisy image
sigma_est = np.mean(estimate_sigma(noise_img, multichannel=True))

patch_kw = dict(patch_size=5,      # 5x5 patches
                patch_distance=6,  # 13x13 search area
                multichannel=True)
```

```
# just nlm with mse
denoise = denoise_nl_means(noise_img, h=0.8 * sigma_est,
sigma=sigma_est,
                           fast_mode=False, **patch_kw)
plt.imshow(denoise, cmap='gray')
plt.show()

#nlm and ssim used together
ssim1 = ssim(noise_img, denoise, data_range=denoise.max() -
denoise.min(), multichannel=True)

sigma_est_1 = np.mean(estimate_sigma(denoise, multichannel=True))
denoise1 = denoise_nl_means(denoise, h=0.8*(sigma_est_1)/ssim1,
sigma=ssim1,fast_mode=False, **patch_kw)

plt.imshow(denoise1, cmap='gray')
plt.show()

psnr_noisy = psnr(orig_img, noise_img)
psnr_denoise = psnr(orig_img, denoise)
psnr_denoise1 = psnr(orig_img, denoise1)


print(f"PSNR (noisy) = {psnr_noisy:0.2f}")
print(f"PSNR (without SSIM) = {psnr_denoise:0.2f}")
print(f"PSNR (with SSIM) = {psnr_denoise1:0.2f}")
```