Anisotropic Diffusion

Da Teng

April 23, 2024

1 Problem Introduction

In the realm of image processing, a common challenge arises when images are corrupted by noise, obscuring the clarity and fidelity of the visual data. Traditional methods of noise reduction often employ frequency-based filtering techniques drawn from classic Digital Signal Processing (DSP). However, these methods often pose a significant drawback: while they effectively mitigate noise, they tend to indiscriminately blur or degrade the edges present within the image, resulting in a loss of crucial detail and structure. To address this issue, there's a need for innovative approaches that strike a balance between noise reduction and edge preservation. One promising avenue involves devising algorithms or methodologies that can intelligently identify and distinguish between noise artifacts and genuine image features, thereby allowing for targeted noise suppression while minimizing the impact on edges. Such methods might leverage advanced signal processing techniques, machine learning algorithms, or a combination thereof, to adjust the filtering process based on the local characteristics of the image. By prioritizing edge preservation alongside noise reduction, these approaches aim to produce denoised images that retain sharper, more faithful representations of the original visual content, catering to a wide range of applications in fields such as photography, medical imaging, and computer vision.

2 Mathematcal Translation

In the context of image denoising, an energy functional represents a mathematical framework used to quantify the quality or "smoothness" of an image. It consists of a mathematical expression that evaluates how well an image conforms to certain desired properties, such as minimizing noise or preserving important features.

$$\int\int L(I(x,y), I_x(x,y), I_y(x,y), x, y)\, dy\, dx \tag{1}$$

Consider g as another image or perturbation beside the image I, where g's endpoints, namely the four edges of the image, are fixed. The derivative the of the energy functional can be calculated as the following

$$\left.\frac{d}{d\epsilon}\right|_{\epsilon=0} E(I+\epsilon g) = \int\int L(I+\epsilon g, I_x+\epsilon g_x, I_y+\epsilon g_y, x, y)\, dy\, dx \tag{2}$$

We have shown in class, through a series of calculation, that

$$\frac{\partial E}{\partial g}(I) = \int\int (L_I - \frac{\partial}{\partial x}L_{I_x} - \frac{\partial}{\partial y}L_{I_y}) g\, dy\, dx =< L_I - \frac{\partial}{\partial x}L_{I_x} - \frac{\partial}{\partial y}L_{I_y}, g > \tag{3}$$

where the term $L_I - \frac{\partial}{\partial x}L_{I_x} - \frac{\partial}{\partial y}L_{I_y}$ can be taken as $\nabla_I E$, the gradient of the energy functional with respect to I.

One way to measure the noise is to look at the value of the derivative. The issue with that is that we cannot account properly for the part of the image where the derivative is large but negative. To avoid this problem, we take the square of the derivative. Adding $\frac{1}{2}$ as the coefficient, the energy functional becomes

$$E(I) = \int\int \frac{1}{2}\|\nabla I\|^2 \, dy \, dx = \int\int \frac{1}{2}(I_x^2 + I_y^2) \, dy \, dx \tag{4}$$

where the term $\int\int \frac{1}{2}(I_x^2 + I_y^2) \, dy \, dx$ represents the function L. Therefore, the gradient of the energy functional with respect to I has become

$$\nabla_I E = L_I - \frac{\partial}{\partial x} L_{I_x} - \frac{\partial}{\partial y} L_{I_y} = -I_{xx} - I_{yy} = -\Delta I \tag{5}$$

3 PDE Derivation

Now we apply the gradient descent and introduce the time variable

$$I_t = -\nabla_I E = \Delta I \tag{6}$$

If we apply this directly to the images that we want to denoise, the results will be disastrous to the edges of the image. The cost/loss function is proportional to the square of the magnitude of the gradient of the image. this means that we are paying an enormous cost for having edges in the image.

Another proposal is to use a linear function. This means that we set the energy functional to be the integral of the magnitude of the gradient itself over the region of the image.

$$E = \int\int \|\nabla I\| \, dy \, dx = \int\int \sqrt{I_x^2 + I_y^2} \, dy \, dx \tag{7}$$

$$\nabla_I E = L_I - \frac{\partial}{\partial x} L_{I_x} - \frac{\partial}{\partial y} L_{I_y} \tag{8}$$

$$= -\frac{\partial}{\partial x}\left(\frac{I_x}{\sqrt{I_x^2 + I_y^2}}\right) - \frac{\partial}{\partial y}\left(\frac{I_y}{\sqrt{I_x^2 + I_y^2}}\right) \tag{9}$$

$$= -\frac{I_y^2 I_{xx} - 2I_x I_y I_{xy} + I_x^2 I_{yy}}{(I_x^2 + I_y^2)^{\frac{3}{2}}} \tag{10}$$

The gradient descent function now becomes

$$I_t = \frac{I_y^2 I_{xx} - 2I_x I_y I_{xy} + I_x^2 I_{yy}}{(I_x^2 + I_y^2)^{\frac{3}{2}}} \tag{11}$$

This closely resembles the geometric heat diffusion process. As we know, geometric heat diffusion selectively eliminates undesirable directions by extracting the linear diffusion component from the heat equation and filtering out the portion diffusing across edges, retaining only the segment aligning with the edge. Essentially, it incorporates an edge-dependent diffusion coefficient into the geometric heat equation. This not only fosters favorable diffusion pathways but also curbs diffusion rates adjacent to edges, as high edges lead to near-zero gradients. Consequently, it regulates both diffusion orientation and intensity, preserving edge details. Unlike the isotropic diffusion of the standard heat equation, which diffuses equally in two orthogonal directions, the geometric heat equation exclusively diffuses along the direction tangent to the image's level curve. Therefore, when diffusing near an edge, it adheres to the level curve contour, enabling diffusion along the edge without crossing it. This targeted diffusion approach significantly mitigates blurring issues.

However, this uncovers a numerical challenge: when the magnitude of the gradient reaches zero, the time step (Δt) also diminishes to zero, halting the diffusion process. This occurs because the norm of the gradient of I continually decreases over time. Consequently, the maximum allowable time step diminishes progressively. This systematic reduction in the time step inevitably culminates in instability at some point.

With that, we will propose the Perona-Malik Diffusion (Anisotropic Diffusion), where

$$E(I) = \int \int c(\|\nabla I\|)\, dy\, dx \tag{12}$$

where $c(\|\nabla I\|)$ should be a function of our choice. This effectively solves the numeric issue that the we had previously. It is import to note that the function c has to be increasing. If we have a negative diffusion coefficient, the problem will be ill-posed. More specifically, slight perturbation in the condition can lead to radically different backward results.

Similarly to the calculations presented above, the gradient descent of anisotropic function now becomes

$$I_t = \nabla \cdot (\frac{c(\|\nabla I\|)}{\|\nabla I\|} \nabla I) \tag{13}$$

The two functions presented in the paper by Pietro Perona and Jitendra Malik are

$$c(\|\nabla I\|) = e^{-(\|\nabla I\|/K)^2} \tag{14}$$

$$c(\|\nabla I\|) = \frac{1}{1 + (\frac{\|\nabla I\|}{K})^2} \tag{15}$$

These functions will also be used in the implementation in this project.

4 Discretization and Implementation

To discretize and implement the provided code for anisotropic diffusion (Perona-Malik filter), several choices must be made regarding the discretization of spatial derivatives, the selection of the time step, and the choice of diffusion coefficients.

Firstly, for the spatial derivatives, central differences are used to approximate gradients in both the x and y directions. Central differences provide a balanced approximation, taking into account information from both neighboring pixels, and offer smoother results compared to forward or backward differences. This choice ensures that the discretization accurately captures the gradient information needed for diffusion while minimizing numerical artifacts.

Regarding the time step (gamma), a smaller value slows down the diffusion process, while a larger value accelerates it. The selection of the time step is crucial to ensure stability and convergence of the numerical scheme. A balance must be struck between computational efficiency and accuracy. Here, a relatively small value of gamma(0.1) is chosen, providing a conservative time step that ensures stability while allowing for efficient computation.

Next, the diffusion coefficient (K) controls the sensitivity to edges. In the Perona-Malik filter, two options are provided for the diffusion coefficient: exponential or quadratic. The exponential option (option 1) smooths edges more aggressively, while the quadratic option (option 2) preserves sharper edges. The choice between these options depends on the desired level of edge preservation versus noise reduction. Here, the exponential option is selected, which is commonly used for noise reduction tasks.

In the implementation, the discretized diffusion equation is iteratively applied to the input image for a specified number of iterations (n_iter). Each iteration updates the image based on the calculated

diffusion coefficients and gradients. The diffusion process is isotropic, affecting all directions equally, and is driven by the combination of the diffusion coefficients and gradients.

In summary, the discretization and implementation of the provided code for anisotropic diffusion involve carefully selecting central differences for spatial derivatives, choosing a conservative time step to ensure stability, and determining the diffusion coefficient option based on the desired balance between edge preservation and noise reduction. These choices are made to ensure accurate and efficient diffusion while minimizing numerical instabilities and artifacts.

5 Experiment

5.1 Adding Noise

Four methods were employed to add noise to the original image, each serving a distinct purpose. The *add_gaussian_noise* function introduced Gaussian noise by generating random samples from a Gaussian distribution with a specified mean and variance. The resulting noise, resembling a bell curve distribution, was added to the original image to simulate random fluctuations in pixel intensity. The *add_salt_pepper_noise* function simulated salt-and-pepper noise by randomly selecting pixels in the image and setting them to either maximum or minimum intensity values (typically 1 and 0, respectively), thereby creating sporadic bright and dark spots reminiscent of salt and pepper sprinkles. On the other hand, the *add_poisson_noise* function modeled Poisson noise, common in low-light imaging scenarios, by generating random samples from a Poisson distribution based on the original image intensity values. This noise type often arises from photon count variations and manifests as grainy texture in images. Lastly, the *add_speckle_noise* function introduced speckle noise, characteristic of radar and ultrasound imaging, by multiplying the original image with random Gaussian noise. This resulted in a grainy or granular appearance, akin to interference patterns observed in such imaging modalities. Each noise addition method was carefully chosen to simulate real-world noise sources encountered in diverse imaging applications, thereby enabling robust evaluation and comparison of denoising algorithms. The images after applying noise can be seen below.

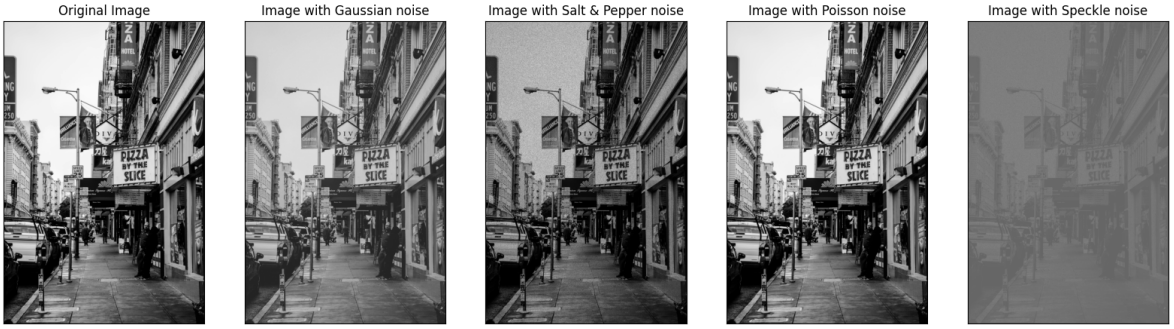

Figure 1: Noisy Images

5.2 Optimization

For the parameter optimization of anisotropic diffusion, a grid search method was employed to systematically explore various combinations of parameters. The parameter space consisted of the number of iterations (n_iter), the conduction coefficient (kappa), the time step (gamma), and the diffusion equation option (option). Ranges were predefined for each parameter, including n_iter_range = [20, 50, 100], kappa_range = [10, 20], gamma_range = [0.05, 0.1, 0.2], and option_range = [1, 2]. For each combination of parameters, the anisotropic diffusion algorithm was applied to the original image, and the Mean Squared Error (MSE) between the original and diffused images was computed. The combination

yielding the lowest MSE was considered the optimal set of parameters for denoising the image. This exhaustive search approach ensured that the chosen parameters provided the most effective denoising results while considering a diverse range of parameter configurations.

After performing a series of optimization on the diffusion parameters, the results obtained are as following:

Gaussian Noise: Best parameters: (20, 20, 0.05, 2) Best MSE: 22.07942157447247

S&P Noise: Best parameters: (100, 20, 0.2, 2) Best MSE: 1909.433660163865

Poisson Noise: Best parameters: (20, 10, 0.05, 1) Best MSE: 5.820018973643632

Speckle Noise: Best parameters: (100, 20, 0.2, 2) Best MSE: 9653.34029181819

5.3 Performance Evalutaion

To evaluate the performance of the algorithms, the project uses several metrics functions used to evaluate the performance of the anisotropic diffusion algorithm. The Mean Squared Error (MSE) function calculates the average squared difference between corresponding pixel intensities of two images, providing a measure of overall image dissimilarity. The Peak Signal to Noise Ratio (PSNR) function utilizes MSE to compute the ratio of the maximum possible signal strength to the noise introduced during image denoising, facilitating quantitative comparison between the original and denoised images. The Structural Similarity Index (SSIM) function assesses image quality by quantifying the similarity in luminance, contrast, and structure between two images. Additionally, the Signal to Noise Ratio (SNR) function quantifies the ratio of signal power to noise power, offering insight into the quality of the denoising process. Lastly, the Edge Preservation Index (EPI) function evaluates the preservation of image edges by comparing the gradient magnitudes of the original and denoised images using the Sobel operator. Together, these metrics provide a comprehensive evaluation of the denoising algorithm's performance in terms of fidelity to the original image, noise reduction, structural similarity, edge preservation, and overall image quality.

5.4 Results

With the optimized parameters selected, we applied the anisotropic diffusion to denoise all four noisy images. To have a more intuitive idea on how anisotropic diffusion performed on these noisy images. We present the comparison results below.

Figure 2: Gaussian Images

Figure 3: S&P Images

Figure 4: Poisson Images

Figure 5: Speckle Images

To evaluate the performance of the diffusion in a more quantitative manner, we apply the error metrics defined earlier on these denoised images.

Table for MSE:

Noise Type	MSE
Poisson Noise	5.820018973643632
Gaussian Noise	22.07942157447247
Salt & Pepper Noise	1909.433660163865
Speckle Noise	9653.34029181819

Table for PSNR:

Noise Type	PSNR
Speckle Noise	8.284027450068193
Salt & Pepper Noise	15.321757866766422
Gaussian Noise	34.69092669089067
Poisson Noise	40.48155960387214

Table for SSIM_INDEX:

Noise Type	SSIM_INDEX
Salt & Pepper Noise	0.4879098420367284
Speckle Noise	0.6476503772672839
Gaussian Noise	0.9200095965690045
Poisson Noise	0.9697172996357983

Table for SNR:

Noise Type	SNR
Speckle Noise	1.7067387833356853
Salt & Pepper Noise	9.157907005247402
Gaussian Noise	28.11641000842709
Poisson Noise	33.906778281812684

Table for EPI:

Noise Type	EPI
Speckle Noise	-234089.95718381912
Salt & Pepper Noise	-39035.3417469472
Poisson Noise	-34178.44591395587
Gaussian Noise	-33104.608311453216

5.5 Analysis

Gaussian Noise
Mean Squared Error (MSE): 22.079 - This relatively low MSE value suggests that anisotropic diffusion is effective at reducing Gaussian noise, indicating minor average squared differences between the original and denoised image pixel intensities. Peak Signal-to-Noise Ratio (PSNR): 34.691 - A high PSNR value implies a higher quality of the denoised image, which correlates with the low MSE, showing effective noise suppression. Structural Similarity Index (SSIM): 0.920 - A near-one SSIM value indicates that structural information and perceived quality are well-preserved post-filtering, affirming the suitability of anisotropic diffusion for Gaussian noise. Signal-to-Noise Ratio (SNR): 28.116 - This robust SNR indicates that the signal has been significantly clarified relative to the noise level, further validating the method's effectiveness. Edge Preservation Index (EPI): -33104.608 - This negative value, although ostensibly counterintuitive, requires context as EPI calculations can vary in sign based on implementation specifics or edge definition in the context of the method used.

Salt & Pepper Noise
MSE: 1909.434 - The high MSE indicates a lesser degree of effectiveness in dealing with Salt & Pepper noise, which features sharp, random disturbances. PSNR: 15.322 - The low PSNR suggests that the denoised image quality is significantly compromised. SSIM: 0.488 - A much lower SSIM indicates poor preservation of structural features, likely due to the random nature of Salt & Pepper noise that challenges smoothing algorithms like anisotropic diffusion. SNR: 9.158 - The low SNR reflects an inadequacy in clearly distinguishing signal from noise. EPI: -39035.342 - Similar to the Gaussian case, the negative EPI value needs contextual understanding.

Poisson Noise
MSE: 5.820 - The very low MSE suggests excellent performance in reducing Poisson noise, which is typical for photon-counting in imaging. PSNR: 40.482 - This high PSNR confirms superior image quality restoration. SSIM: 0.970 - A very high SSIM value reflects outstanding maintenance of texture and structure. SNR: 33.907 - Indicates that the signal strength post-denoising is robust compared to the noise. EPI: -34178.446 - As with other noise types, interpreting EPI requires specific knowledge of the implementation details.

Speckle Noise
MSE: 9653.340 - Extremely high MSE indicates very poor performance in reducing Speckle noise, which often affects granular data. PSNR: 8.284 - This very low PSNR is indicative of very poor image quality post-denoising. SSIM: 0.648 - This mediocre SSIM score shows that while some structural integrity is preserved, a lot is lost. SNR: 1.707 - The nearly negligible SNR underscores a failure in effective noise suppression. EPI: -234089.957 - The highly negative value here significantly underscores the challenges in edge preservation with Speckle noise.

5.6 Other Denoising Methods

In this project, I also implemented two distinct image denoising techniques: Non-Local Means Denoising (NLMeans) and Median Filtering, to enhance the quality of images corrupted by Gaussian and salt & pepper noise. The NLMeans approach involves replacing each pixel's value with a weighted average of other pixels' values. This weight is calculated based on the similarity between small patches centered at the pixels being compared, controlled by parameters such as the filtering strength (h), the size of the search window, and the patch size. For this method, I chose a filtering strength of 1, a search window of 3, and a patch size of 5. This technique effectively preserves edges while reducing noise by averaging similar patches.

Conversely, the Median Filtering method works by sliding a kernel over the image and replacing each pixel with the median value of the pixels within the kernel's neighborhood. I applied a kernel size of 3, which is robust against outliers, particularly in images with salt & pepper noise, making it effective in maintaining the structural integrity of the image while removing noise.

To assess the performance of these methods, I computed three metrics: Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Edge Preservation Index (EPI). These metrics were calculated for images denoised using both methods against the original grayscale images. The results indicated that NLMeans was particularly effective in handling Gaussian noise, showing improvements in all three metrics, suggesting superior edge preservation and overall similarity to the original image. Similarly, median filtering showed significant effectiveness against both types of noise, with a notable reduction in MSE and improvements in SSIM and EPI, confirming its utility in noise reduction and edge preservation.

The image comparison results for these two methods on two noisy images are shown below.

Again, to evaluate the results in a more quantitative fashion, we present the tables for error metrics for these two methods below.

NLMEANS Method:

Noise Type	MSE	SSIM_INDEX	EPI
Gaussian Noise	99.39259675292453	0.6817348237817769	-27602.296594969506
Salt & Pepper Noise	5489.215943150156	0.27272691869149857	-79197.02692412655

Median Method:

Noise Type	MSE	SSIM_INDEX	EPI
Gaussian Noise	34.49632231099796	0.8716861102166145	-31034.78301922595
Salt & Pepper Noise	1274.9083214214384	0.5449820347083472	-50909.510504680315

Comparing the performance of the Non-Local Means (NLMeans) method, the Median method, and Anisotropic Diffusion across two noise types—Gaussian Noise and Salt & Pepper Noise—reveals varying degrees of efficacy in noise reduction and image restoration, as reflected by metrics such as Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Edge Preservation Index (EPI).

Gaussian Noise

Anisotropic Diffusion shows strong performance with the lowest MSE (22.08) and highest SSIM (0.920) compared to the NLMeans and Median methods. This indicates superior noise reduction and better preservation of structural details.
NLMeans yields a higher MSE (99.39) and a lower SSIM (0.682) than Anisotropic Diffusion, suggesting less effective noise handling for Gaussian noise.
Median Method performs better than NLMeans with an MSE of 34.50 and an SSIM of 0.872, showing good noise reduction and structural preservation, though not as effective as Anisotropic Diffusion.

Salt & Pepper Noise
Anisotropic Diffusion struggles with this noise type, with a very high MSE (1909.43) and low SSIM (0.488). However, it still outperforms NLMeans in terms of SSIM, indicating slightly better structural integrity retention.
NLMeans is significantly less effective, having the highest MSE (5489.22) and the lowest SSIM (0.273), suggesting it is poorly suited for handling sharp, random disturbances like Salt & Pepper noise.
Median Method significantly improves upon NLMeans with a lower MSE (1274.91) and higher SSIM (0.545). The nature of the Median filter, which is specifically effective against such noise, helps it outperform the other two methods in both metrics for this noise type.

6 What We Learned

Through the execution of this project on anisotropic diffusion for image denoising, several significant learnings were garnered, particularly in understanding the intricate balance between theory and practical implementation in mathematical modeling and computational simulations. The strengths and weaknesses of the Partial Differential Equation (PDE) approach were elucidated through rigorous experimental testing.

One of the most enlightening aspects of this project was the revelation of how anisotropic diffusion can be both a robust solution and a potential source of complication in specific scenarios. The theoretical premise of anisotropic diffusion promises enhanced noise reduction while preserving edge details, which is critically important in practical applications like medical imaging and photography where detail retention is paramount. However, during experimental application, it became evident that the performance of anisotropic diffusion varied significantly with different types of noise. While it showed excellent results in reducing Gaussian and Poisson noise, it struggled with speckle and salt & pepper noise, which are characterized by their sharp and random disturbances. This discrepancy highlighted a weakness in the PDE-based model's ability to adapt its diffusion coefficients effectively across different noise patterns.

The assumption that anisotropic diffusion uniformly handles various noise types without adjusting the diffusion coefficients accordingly was too simplistic. This assumption failed to accommodate the stochastic nature of noise like salt & pepper, which requires a more dynamic adaptation strategy to effectively distinguish between noise and true edge information. The experiments conducted provided empirical evidence that adjusting the diffusion coefficient based on the local gradient magnitude is essential but not always sufficient for optimal noise reduction.

Given more time and resources, several modifications could be explored to refine the model. For instance, integrating a machine learning approach to dynamically adjust diffusion coefficients in real-time based on the detected noise characteristics could potentially enhance the model's versatility. Additionally, exploring a hybrid approach that combines the strengths of anisotropic diffusion with other denoising techniques, such as Non-Local Means or Median Filtering, could address the limitations observed with harsher noise types. Such hybrid models could leverage the structural advantages of each method, offering a more comprehensive solution to diverse noise problems.

Moreover, the project illuminated the importance of discretization and implementation choices in the performance of computational algorithms. The selection of central differences for spatial derivatives and the conservative approach to the time step in the numerical scheme were crucial in ensuring

the stability of the solution but also introduced limitations in terms of computational efficiency and responsiveness to rapidly changing image features. This experience underscores the need for a careful balance between accuracy, stability, and computational overhead in the design of numerical algorithms for real-world applications.

References

1. Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7), 629-639. *IEEE Xplore.* [Accessed April 20, 2024].

2. Jain, V. (2024). NLMeans Denoising Implementation. *GitHub repository.* Available: https://github.com/varunjain3/NLMeansDenoising/blob/main/NLmeans.py [Accessed April 23, 2024].

3. Unknown Author. (2014). How to add noise (Gaussian, salt and pepper, etc.) to image in Python with OpenCV. *Stack Overflow.* Available: https://stackoverflow.com/questions/22937589/how-to-add-noise-gaussian-salt-and-pepper-etc-to-image-in-python-with-opencv [Accessed April 23, 2024].

4. HanC, M. (n.d.). Python Median Filter. *GitHub repository.* Available: https://github.com/MeteHanC/Python-Median-Filter/blob/master/MedianFilter.py [Accessed April 23, 2024].