ECE 6560 - Final Project Report
Image De-noising with Perona Malik (Anistropic) Diffusion

Prasanth Bathala

Spring 2024

1 Problem Statement

Images are a fundamental medium for conveying information visually. Within the realm of image processing, a common challenge arises in the form of image denoising. This process involves the removal of unwanted noise to restore an image's original clarity. This noise can stem from various sources such as sensor interference, environmental factors, or disruptions during transmission. While efforts have been made to mitigate noise without compromising crucial image features like edges and corners, traditional methods like median filtering may fall short of preserving these details. Hence, isotropic smoothing techniques often prove ineffective for comprehensive denoising. In this project, I aim to explore anisotropic methods, particularly focusing on the Perona-Malik Diffusion approach, to ascertain their efficacy in achieving robust denoising results.

The code for this project is available at:
`https://github.com/prasbathala/Perona-Malik-Diffusion-ECE6560`

2 Mathematical Formulation

Assume an image $I(x, y, t)$ where x and y are spatial coordinates, and t is the time variable. Our objective is to mitigate the noise within this image. Given that noise typically occurs as high-frequency signals, we will formulate an energy function that quantifies the gradient of the image. The energy function, denoted by equation 1, encapsulates our minimization goal.

$$E(I(x,y)) = \iint C(||\nabla I||)dxdy \tag{1}$$

where $||.||$ is the L2 norm and $C(||\nabla I|| = L(I, I_x, I_y, x, y)$

As discussed in the class, the Euler-Lagrange equation for equation 1:

$$L_I - \frac{\partial}{\partial x}L_{I_x} - \frac{\partial}{\partial y}L_{I_y} = 0 \tag{2}$$

For L_{I_x}:

$$L_{I_x} = \frac{\partial}{\partial x}C(||\nabla I||)$$

$$= \frac{\partial}{\partial x} C(\sqrt{I_x^2 + I_y^2})$$

$$L_{I_x} = C'(||\nabla I||)\frac{I_x}{||\nabla I||} \tag{3}$$

Similarly, L_{I_y}:

$$L_{I_y} = C'(||\nabla I||)\frac{I_x}{||\nabla I||} \tag{4}$$

and

$$L_I = 0 \tag{5}$$

The gradient of the energy function is given by:

$$\nabla E = -I_t \tag{6}$$

Combining equations (3), (4), (5), we have:

$$I_t = \frac{\partial}{\partial x} L_{I_x} + \frac{\partial}{\partial y} L_{I_y}$$

$$I_t = \nabla . (\frac{C'(||\nabla I||)}{||\nabla I||} \nabla I) \tag{7}$$

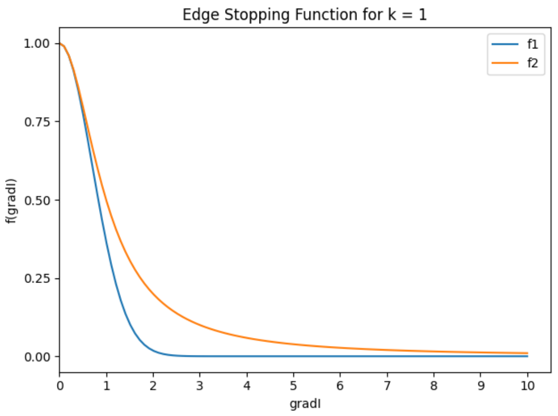

Figure 1: Effect of diffusion for different edge functions for K = 1

This is the generic form of energy function, and the derivation of the PDE from the energy function. Now, let's focus on the Perona-Malik anisotropic diffusion [1], where the diffusion coefficient is chosen as an edge-stopping function. The edge-stopping is naturally a monotonic decreasing function of ∇I which should be ideally $\lim_{\nabla I \to 0} c(\nabla I) = 1$ - that rate of diffusion is high within the uniform or non-edge parts of the image and $\lim_{\nabla I \to 0} c(\nabla I) = 0$ - that the diffusion is 0 at the edges.

Perona and Malik [1] proposed two diffusion coefficient functions given by:

$$C(||\nabla I||) = e^{-(||\nabla I||/K)^2} \tag{8}$$

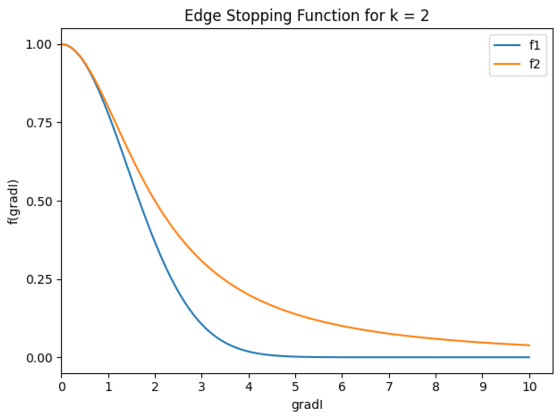

Figure 2: Effect of diffusion for different edge functions for K = 2

and

$$C(||\nabla I||) = \frac{1}{1 + (\frac{||\nabla I||}{K})^2} \tag{9}$$

where K is the edge sensitivity factor. Equation (8) is $f1$ and equation (9) is $f2$ in the Figure 1 and 2. We can observe from Figure 1 and Figure 2, that edge functions penalize high for higher image gradients and low for lower image gradients. For this project, I chose 9 for all the experiments carried out.

3 Discretization and Implementation

3.1 Discretization

From equation 7, The Partial Differential Equation for Perona Malik diffusion can further be computed as:

$$\begin{aligned} I_t &= \nabla.(C(||\nabla I||)\nabla I) \\ &= \nabla C(\|\nabla I\|)\nabla I + C(\|\nabla I\|)\Delta I \end{aligned}$$

For computational implementation, we discretize this equation. Let $C = C(\|\nabla I\|)$ for simplicity in the derivation,

$$\begin{aligned} I_t &= \nabla C \cdot \nabla I + C\Delta I \\ &= (C_x + C_y)(I_x + I_y) + C(I_{xx} + I_{yy}) \\ &= C_x I_x + C_y I_y + C(I_{xx} + I_{yy})] \end{aligned} \tag{10}$$

Now, let us use the diffusion coefficients in equations (8) and compute the derivatives

$$
\begin{aligned}
C_x I_x &= \frac{\partial}{\partial x}\left(e^{-\frac{I_x^2+I_y^2}{K^2}}\right) I_x \\
&= e^{-\frac{I_x^2+I_y^2}{K^2}}\left(-\frac{2I_x}{K^2}(2I_x I_{xx} + 2I_x I_y I_{xy})\right) \qquad (11)
\end{aligned}
$$

Similarly,$C_y I_y$:

$$
\begin{aligned}
C_y I_y &= \frac{\partial}{\partial y}\left(e^{-\frac{I_x^2+I_y^2}{K^2}}\right) I_y \\
&= e^{-\frac{I_x^2+I_y^2}{K^2}}\left(-\frac{2I_y}{K^2}(2I_y I_{yy} + 2I_x I_y I_{xy})\right) \qquad (12)
\end{aligned}
$$

Substituting equations (11 & 12) in equation (10):

$$
I_t = \frac{1}{e^{\frac{I_x^2+I_y^2}{K^2}}} \frac{K^2(I_{xx}+I_{yy}) - 2I_x^2 I_{xx} - 2I_y^2 I_{yy} - 4I_x I_y I_{xy}}{K^2} \qquad (12)
$$

Similarly for diffusion coefficient equation 9:

$$
\begin{aligned}
C_x I_x &= \frac{\partial}{\partial x}\left(\frac{1}{1+\frac{I_x^2}{K^2}}\right) I_x \\
&= \frac{-K^2}{K^2+I_x^2+I_y^2}(2I_x I_{xx} + 2I_x I_y I_{xy}) \qquad (13)
\end{aligned}
$$

Similarly, $C_y I_y$:

$$
\begin{aligned}
C_y I_y &= \frac{\partial}{\partial y}\left(\frac{1}{1+\frac{I_y^2}{K^2}}\right) I_y \\
&= \frac{-K^2}{K^2+I_x^2+I_y^2}(2I_y I_{yy} + 2I_x I_y I_{xy}) \qquad (14)
\end{aligned}
$$

Substituting equations (13 and 14) in equation (10), we get:

$$
I_t = \frac{K^2(I_{xx}+I_{yy}) - 2I_x^2 I_{xx} - 2I_y^2 I_{yy} - 4I_x I_y I_{xy}}{K^2+I_x^2+I_y^2} \qquad (9)
$$

To program this, we discretize partial derivatives using forward and central differences. The forward difference for time approximation, requiring only one previous time step, is simple and computationally efficient. The central difference is employed for space derivative approximation to minimize error.

$$I_t = \frac{I(t+\Delta t) - I(t)}{\Delta t} \tag{15}$$

$$I_x = \frac{I(x+\Delta x) - I(x-\Delta x)}{2\Delta x} \tag{16}$$

$$I_y = \frac{I(y+\Delta y) - I(y-\Delta y)}{2\Delta y} \tag{17}$$

$$I_{xx} = \frac{I(x+\Delta x) - 2I_x + I(x-\Delta x)}{\Delta x^2} \tag{18}$$

$$I_{yy} = \frac{I(y+\Delta y) - 2I_y + I(y-\Delta y)}{\Delta y^2} \tag{19}$$

$$\begin{aligned} I_{xy} = \frac{1}{4\Delta x \Delta y}\big[&I(x+\Delta x, y+\Delta y) + I(x-\Delta x, y-\Delta y) \\ &- I(x+\Delta x, y-\Delta y) - I(x-\Delta x, y+\Delta y)\big] \end{aligned} \tag{20}$$

3.2 Stability Analysis

In this project, I will be implementing Perona-Malik diffusion using the above discretization scheme because of its stability. Consider $\Delta x = \Delta y = 1$ be the spatial steps and Δt be the time step.

$$\begin{aligned} I(x,y,t+1) = I(x,y,t) + \Delta t\big[&C_N \nabla_N I(x,y,t) + C_S \nabla_S I(x,y,t) \\ &+ C_E \nabla_E I(x,y,t) + C_W \nabla_W I(x,y,t)\big] \end{aligned} \tag{21}$$

where gradients are:

$$\nabla_N I(x,y,t) = I(x-1,y,t) - I(x,y,t) \tag{22}$$

$$\nabla_S I(x,y,t) = I(x+1,y,t) - I(x,y,t) \tag{23}$$

$$\nabla_E I(x,y,t) = I(x,y+1,t) - I(x,y,t) \tag{24}$$

$$\nabla_W I(x,y,t) = I(x,y-1,t) - I(x,y,t) \tag{25}$$

with $0 < \Delta t \leq \frac{1}{4}$ for the scheme to be stable.

A more efficient approach to determining the CFL condition, which avoids the computationally intensive task of computing the 2D Discrete Fourier Transform (DFT) of equations (15-20), involves initially calculating the CFL condition for the 1D heat equation. This condition is then extended to the 2D case due to its inherent symmetry. This streamlined method simplifies the computational process and conserves resources.

Define $C(\|\nabla I\|) = C$.

$$\Delta t \leq \frac{(\Delta x)^2}{4C} \tag{26}$$

$$\Delta t \leq \frac{(\Delta y)^2}{4C} \tag{27}$$

Consider $\Delta x = \Delta y = 1$

$$\Delta t \leq \frac{1}{4C} \tag{28}$$

We observe diffusivity constrained within the range of 0 to 1. Therefore, regardless of the value of K, C will be bounded by $0 \leq C \leq 1$ and $\Delta t \leq 0.25$ will ensure the convergence of the scheme.

4 Experiments

This section discusses about the different experiments conducted for image denoising with Perona Malik diffusion.

4.1 Evaluation Criteria

We evaluate the quality of the denoised images based on the following metrics:

Notations:

- $\hat{I}$ - Denoised image,
- I - Original Image
- MSE - Mean Square Error

4.1.1 Peak Signal-to-Noise Ratio (PSNR)

The Peak Signal-to-Ratio (PSNR) is calculated as the ratio of the power of the maximum potential image intensity within a volume to the power of the distortions of the noise. Higher PSNR indicate better denoising.

$$PSNR(I, \hat{I}) = 10 \log \frac{max(I)^2}{MSE(\hat{I}, I)} \tag{29}$$

4.1.2 Mean Square Error

Mean Square Error (MSE) typically prioritizes smoothness over sharpness. Lower MSE indicate better denoising quality.

$$MSE(I, \hat{I}) = \frac{\sum ||\hat{I} - I||^2}{\sum ||I||^2} \tag{30}$$

4.2 Experimental Setup

The primary objective of this experiment is to evaluate the efficacy of denoising with Perona Malik diffusion on image corrupted with different types of noise:

Salt and Pepper Noise: This form of noise introduces random, isolated bright and dark pixels throughout the image.

Poisson Noise: This type of noise is modeled based on poisson distribution generated from the data.

Figure 4 illustrates the original image alongside versions corrupted by Salt & Pepper and Poisson noise.

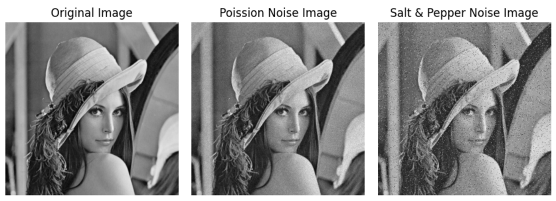

Figure 3: Image with Salt and Pepper and Poisson noise added

4.3 Median Filtering

As a basline method, the median filtering technique was implemented, which serves as a reference for comparison with Perona Malik Diffusion. In Median Filtering, the aim is to suppress the noise by replacing each pixel value with median value within a local neighborhood.

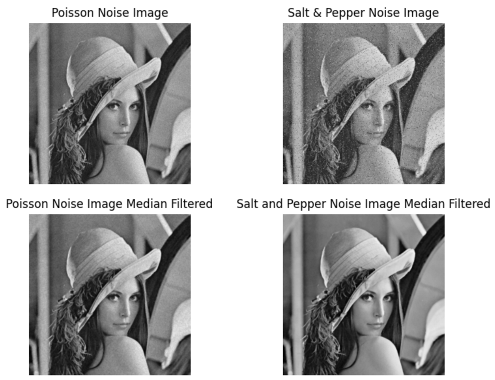
Poisson Noise Image
Salt & Pepper Noise Image
Poisson Noise Image Median Filtered
Salt and Pepper Noise Image Median Filtered

Figure 4: Denoised images with Median filtering Technique

Table 1 presents the PSNR and NMSE values for images corrupted with Poisson and Salt & Pepper noise. Lower PSNR values indicate greater noise distortion, with Poisson noise exhibiting a PSNR of -15.26 and Salt & Pepper noise measuring -11.25.

4.4 Perona Malik Diffusion

We'll implement Perona Malik Diffusion for image denoising. Initially, let's examine the fluctuation of diffusion coefficients concerning the parameter K. I utilized the diffusion coefficient equations referenced in 8 and 9, computing the diffusion coefficients across a range of K values. Additionally, I generated graphs depicting the relationship between different K values and F.

Table 1: Denoising Performance Metrics with Median Filtering

Metric	**Poisson Noise**	**Salt & Pepper Noise**
PSNR	-15.26	-11.25
NMSE	8002.13	12389.63

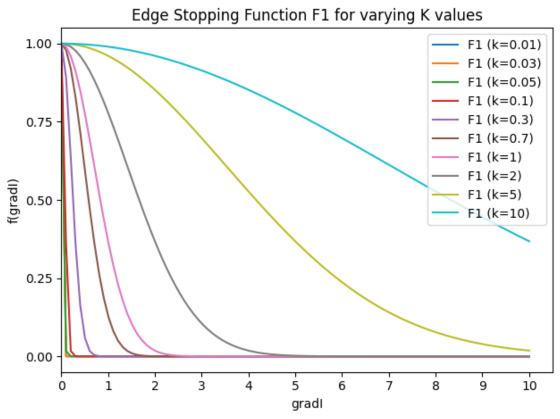

Figure 5: Sensitivity of F1 (8) for varying K values

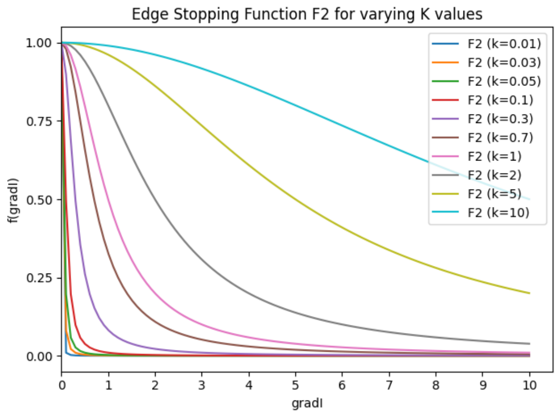

Figure 6: Sensitivity of F2 (9) for varying K values

The analysis from figures 5 and 6 illustrates a clear trend: as the K value increases in Perona-Malik diffusion, the diffusion's impact becomes more pronounced, especially on areas with higher gradient values. This heightened diffusion tends to smooth out image details, particularly affecting edges and corners, resulting in a loss of fine features. Thus, it is very crucial to strike a balance between noise reduction and preservation of the image structure.

4.4.1 Temporal Analysis of Perona-Malik Diffusion Process: Evolution Across Different Time Steps.

In this experiment, we explored the temporal evolution of the Perona-Malik diffusion process applied to images afflicted with both Poisson and Salt & Pepper noise. Our investigation involved visually tracking the changes in image appearance over successive iterations of the diffusion process and quantifying its performance using Peak Signal-to-Noise Ratio (PSNR) and Normalized Mean Square Error (NMSE).

Throughout all experiments conducted on the Perona-Malik Diffusion, F2 (9) has been selected as the edge-stopping function.

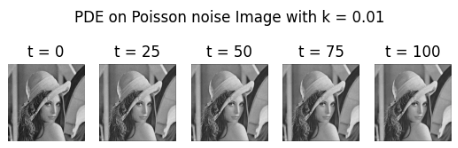

Figure 7: PDE for Poisson Image for K = 0.01

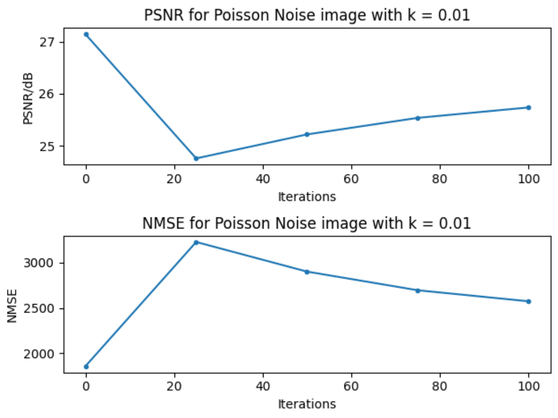

Figure 8: Variation of PSNR and NMSE for PDE on Poisson noise Image with K = 0.01

For Poisson Noise Image: Figure 7 illustrates the evolving dynamics of the Perona-Malik diffusion process applied to an image corrupted with Poisson Noise. The corresponding performance plots in Figure 8 detail the analysis of PSNR and NMSE across the iterations. Notably, the PSNR exhibits an initial decline until the 22nd iteration, indicative of an early phase of information loss. However, subsequent iterations reveal a significant improvement in PSNR, signifying iterative refinement and ultimately enhancing image quality.

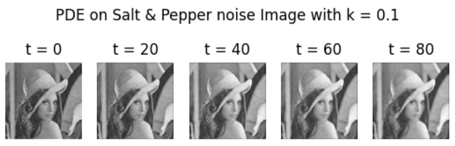

Figure 9: PDE for Salt & Pepper Image for K = 0.1

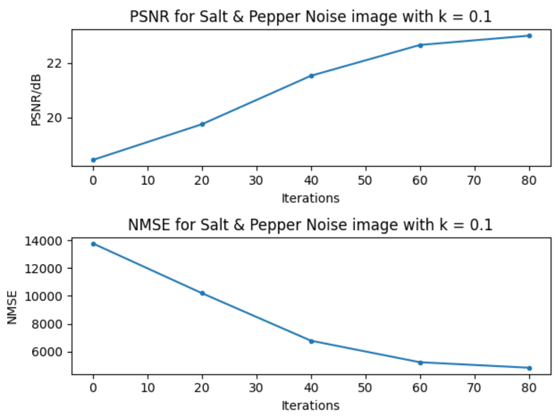

Figure 10: Variation of PSNR and NMSE for PDE on Salt & Pepper noise Image with K = 0.1

For Salt & Pepper Image: Figure 9 showcases the evolving denoising process with Perona-Malik diffusion on an image affected by Salt & Pepper noise. Evaluating performance metrics, as illustrated in Figure 10, involved analyzing PSNR and NMSE across iterations. Remarkably, for the Salt & Pepper noise-corrupted image, there is no initial decline observed in PSNR across iterations; instead, a consistent increment in PSNR is observed throughout the iterative process. The sustained increase in PSNR underscores the efficacy of iterative refinement in steadily enhancing image quality without notable information loss.

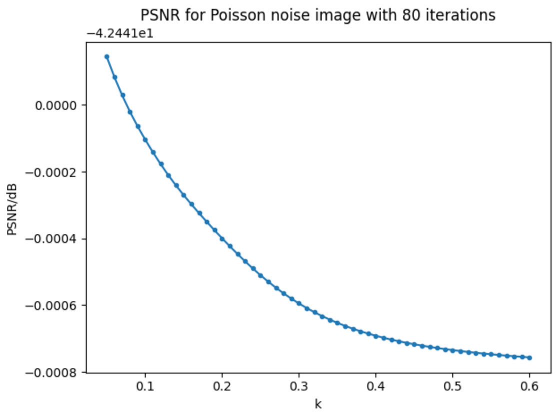

Figure 11: PSNR for Poisson noise image with different K

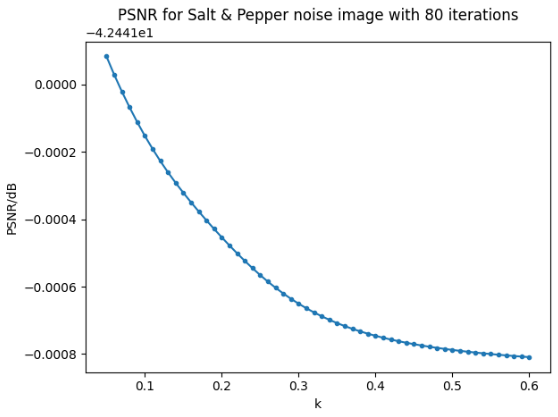

Figure 12: PSNR for Salt & Pepper noise image with different K

4.4.2 Exploring PSNR Variation with Varying K Values

In this experiment, we investigate how varying K values impact PSNR performance for both noisy images. Our goal is to understand how changes in the diffusion parameter K affect the denoising efficacy as measured by PSNR metrics. Figure 11 illustrates the PSNR performance variation with K for the Poisson noise image, while Figure 12 depicts the same for the Salt & Pepper image. Across both images, we observe a consistent trend: as the K value increases, PSNR performance decreases. This decline aligns with expectations, as higher K values are known to result in increased loss of information around edges and corners, as discussed in section 4.4.

4.5 Comparative Analysis

Figure 13 presents a visual comparison of different image denoising methods. Even though there's still some noise visible in the Perona Malik diffusion image compared to the one filtered with median filtering, the Perona Malik method actually has a higher PSNR (Peak Signal-to-Noise Ratio) and MSR (Mean Square Error) when measured as shown in that table 2. Notably Perona-Malik diffusion often achieves higher PSNR values, demonstrating its robustness.

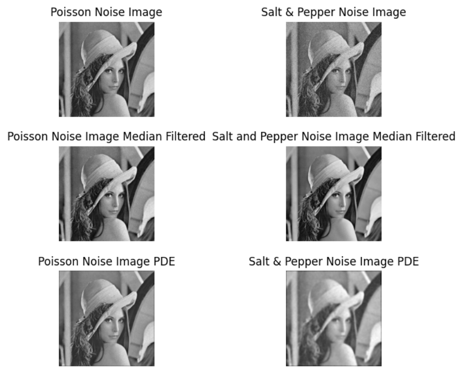
Poisson Noise Image
Salt & Pepper Noise Image
Poisson Noise Image Median Filtered
Salt and Pepper Noise Image Median Filtered
Poisson Noise Image PDE
Salt & Pepper Noise Image PDE

Figure 13: PSNR for Salt & Pepper noise image with different K

Table 2: Optimal Parameters and Corresponding PSNR and MSE values

Noise	Method	Optimal K	PSNR (dB)	NMSE
Salt & Pepper	Median Filtering	N/A	-11.25	12389.63
Salt & Pepper	Perona Malik F2 (9)	0.01	22.95	5500
Poisson	Median Filtering	N/A	-15.26	8002.13
Poisson	Median Filtering	0.1	23.44	2625.56

5 Discussion and Future Work

The experimental results demonstrate notable differences in denoising effectiveness across various types of noise. Particularly, the Perona-Malik diffusion method demonstrated superior performance for both noise types, underscoring its suitability for applications requiring preservation of fine details. In contrast, certain techniques such as median filtering exhibited poor performance under specific noise conditions.

Future research directions encompass the optimization of the Perona-Malik algorithm to accommodate dynamic noise profiles, the exploration of deep learning methodologies for data-centric denoising approaches, the enhancement of algorithmic efficiency for real-time implementation, and the comprehensive validation of our findings across real-world datasets.

References

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