Image Denoising Using Anisotropic Diffusion PDE

Vats, Tavish
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
tvats3@gatech.edu

April 26, 2023

1 Introduction

Image Denoising is a fundamental concept in the field of computer vision. This method is an essential step when it comes to improving image quality. Images are an everyday part of our lives. In the digital age we live in, factors such as image acquisition and transmission can cause image degradation. Image degradation can be thought of as noise being added to the image, decreasing its quality and leading to loss of information. The presence of noise in an image can have a negative impact on various applications such as medical image processing and object recognition. Due to this, work needs to be done to develop methods of image denoising to solve this problem.

Solving the problem of noise in an image is a well-researched task and there are many solutions proposed in the literature for noise reduction. A common and classic method for image denoising doesn't involve any machine learning. This method involves convolving the image with a low-pass filter kernel such as the Gaussian Kernel. The Gaussian kernel has an isotropic distribution meaning that the behavior of the function is the same in all directions. This method can reduce high-frequency noise and blur the edges of an image effectively. However, blurring of the image may be undesirable, and can reduce the final image detail leading to information loss.

In this report, we will be looking into non-linear anisotropic diffusion also known as Perona-Malik diffusion. This technique applies anisotropic filtering to reduce noise in an image while preserving the significant parts of the image such as the edges. In the following sections, I will explain in detail how the Perno-Malik diffusion PDE is formulated mathematically and conduct experiments on noisy images to analyze the results and performance of this denoising technique using various metrics.

2 Problem Setup

In this section, I will be explaining the problem mathematically and deriving the Peronal-Malik diffusion PDE. Let's suppose we have an image I that can be represented as a function $I(x, y)$, where the two unknown variables x and y represent the coordinates of a pixel in an image.

Using the Calculus of Variations method, we can define an energy functional which takes in the function $I(x, y)$. This energy functional acts as the measure of noise in an image. So, our objective is to define the energy functional and to minimize it, which reduces the noisiness of an image. So we are going to establish a gradient descent PDE using the energy function which will ideally reduce the noise in an image. Let's express the general form of the energy functional as follows:

$$E(I) = \int_{\Omega} c(\|\nabla I\|)dxdy \tag{1}$$

The energy functional shown above takes in a simple function c that takes an argument which is the gradient of the multi-variable function I. The function $c(\|\nabla I\|)$ is a non-decreasing function ($\dot{c}(\|\nabla I\|) \geq 0$) and can be expressed as the Lagrangian:

$$c(\|\nabla I\|) = c(\sqrt{I_x^2 + I_y^2}) = L\left(I, I_x, I_y\right) \tag{2}$$

Next, we can find the gradient of the energy functional with respect to image I using the equations [1] and [2]:

$$\nabla_I E = \nabla_I (L\,(I, I_x, I_y)) = L_I - \frac{\partial L_{I_x}}{\partial x} - \frac{\partial L_{I_y}}{\partial y} \tag{3}$$

Since the equation $c(\sqrt{I_x^2 + I_y^2})$ doesn't have an I term in it, we can say that $L_I = 0$. So we end up with:

$$\nabla_I E = -(\frac{\partial L_{I_x}}{\partial x} + \frac{\partial L_{I_y}}{\partial y}) \tag{4}$$

Now, if we put L_{I_x} and L_{I_y} together in a vector, we create a vector field in 2D. Then if we take the x derivative of the x component and the y derivative of the y component, the sum corresponds to the divergence of the vector field. Keeping that in mind, we can write equation [4] as follows:

$$\nabla_I E = -\nabla \begin{pmatrix} L_{I_x} \\ L_{I_y} \end{pmatrix} \tag{5}$$

Now, let's solve for L_{I_x} and L_{I_y} as follows:

$$L_{I_x} = \dot{c}(\|\nabla I\|) \frac{I_x}{\sqrt{I_x^2 + I_y^2}} \tag{6}$$

$$L_{I_y} = \dot{c}(\|\nabla I\|) \frac{I_y}{\sqrt{I_x^2 + I_y^2}} \tag{7}$$

Now, using equations [6] and [7] we can write the gradient $\nabla_I E$ as:

$$\nabla_I E = -\nabla \cdot \left[\begin{pmatrix} I_x \\ I_y \end{pmatrix} \frac{\dot{c}(\|\nabla I\|)}{\sqrt{I_x^2 + I_y^2}} \right] = -\nabla \cdot \left[\begin{pmatrix} I_x \\ I_y \end{pmatrix} \frac{\dot{c}(\|\nabla I\|)}{\|\nabla I\|} \right] = -\nabla \cdot \left(\frac{\nabla I}{\|\nabla I\|} \dot{c}(\|\nabla I\|) \right) \tag{8}$$

Now to calculate the gradient descent, we use the following equation:

$$I_t = -\nabla_I E \tag{9}$$

Here, the above equation represents the time derivative of the negative (descent) of the gradient of the energy functional. The time variable in the gradient descent PDE is the physical evolution parameter of the gradient descent process. So, now we can use the results from [9] to show the generic form of the non-linear diffusion PDE.

$$I_t = \nabla \cdot \left(\frac{\dot{c}(\|\nabla I\|)}{\|\nabla I\|} \nabla I \right) \quad \text{where} \quad \frac{\dot{c}(\|\nabla I\|)}{\|\nabla I\|} = \text{positive diffusion coefficient} \tag{10}$$

So far, we have shown a generalized form of the non-linear diffusion PDE which has a positive non-decreasing diffusion coefficient. We can call this diffusion coefficient as $C(\|\nabla I\|)$ which depends on the norm of the gradient of I. Keep in mind, the C in the diffusion coefficient is not the same as the c in the equation [1]. So let's rewrite the PDE as follows:

$$I_t = \nabla \cdot (C(\|\nabla I\|) \nabla I) \tag{11}$$

In terms of the diffusion coefficient $C(\|\nabla I\|)$, if we treat it as a constant C then it becomes an isotropic diffusion PDE and corresponds to the linear heat equation $I_t = \nabla(C\nabla I) = \Delta I$ with a constant C. As explained before, isotropic distribution denoises of the images uniformly in all directions, which

can lead to the blurring of edges and reducing the fine details of the image. However, if $C(\|\nabla I\|)$ is non-linear we can denoise the image while preserving the edges to prevent loss of information. Perona-Malik presents two non-linear diffusion coefficient functions [1] that can be used as $C(\|\nabla I\|)$ as follows:

$$C(\|\nabla I\|) = e^{-(\frac{\|\nabla I\|}{K})^2} \tag{12}$$

$$C(\|\nabla I\|) = \frac{1}{1 + (\frac{\|\nabla I\|}{K})^2} \tag{13}$$

The two non-linear diffusion coefficients defined in equation [12] and [13] have a constant K, which controls the sensitivity to the edges of the image. In the following sections, I will describe how I implemented the PDE using the Perona-Malik diffusion coefficient to denoise images with different types of noise. Also, I will conduct experiments on how well the noise from the image is removed using metrics such as Peak Signal-to-Noise ratio (PSNR), Mean-Squared-Error (MSE), and structural similarity index measure (SSIM).

3 Implementation

In order to discretize the PDE and translate it to code, I used the book Geometry-Driven Diffusion in Computer Vision by Bart M. Haar Romeny as a reference point [1]. Firstly, I will go through the process of discretizing equation [11] using Perona-Malik diffusion coefficient [13] to get the final Peronal-Malik Diffusion PDE. We start by expanding equation [11] as follows:

$$I_t = \nabla \cdot (C(\|\nabla I\|)\nabla I) = \nabla C(\|\nabla I\|)\nabla I + C(\|\nabla I\|)\Delta I \tag{14}$$

Then we substitute the Perona-Malik diffusion coefficient [13] in equation [14] and expand the equation further:

$$\begin{aligned} I_t &= \nabla\left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)\nabla I + \left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)\Delta I \\ &= \frac{\partial}{\partial x}\left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)I_x + \frac{\partial}{\partial y}\left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)I_y + \frac{I_{xx} + I_{yy}}{1 + (\frac{\|\nabla I\|}{K})^2} \end{aligned} \tag{15}$$

Looking at equation [15], we can compute each of the partial derivatives, simplify the function, and expand to get the final expression of the diffusion PDE as follows:

$$\begin{aligned} I_t &= \frac{\partial}{\partial x}\left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)I_x + \frac{\partial}{\partial y}\left(\frac{1}{1 + (\frac{\|\nabla I\|}{K})^2}\right)I_y + \frac{I_{xx} + I_{yy}}{1 + (\frac{\|\nabla I\|}{K})^2} \\ &= \left(-\frac{2I_xI_{xx} + 2I_yI_{xy}}{(1 + \frac{I_x^2+I_y^2}{K^2})^2K^2}\right)I_x + \left(-\frac{2I_xI_{yx} + 2I_yI_{yy}}{(1 + \frac{I_x^2+I_y^2}{K^2})^2K^2}\right)I_y + \frac{I_{xx} + I_{yy}}{1 + (\frac{I_x^2+I_y^2}{K^2})} \\ &= \frac{-2I_x^2I_{xx} - 4I_xI_yI_{xy} - 2I_y^2I_{yy}}{(1 + \frac{I_x^2+I_y^2}{K^2})^2K^2} + \frac{K^2(I_{xx} + I_{yy}) + I_x^2I_{xx} + I_x^2I_{yy} + I_y^2I_{xx} + I_y^2I_{yy}}{(1 + \frac{I_x^2+I_y^2}{K^2})^2K^2} \\ I_t &= \frac{K^2(I_{xx} + I_{yy}) + (I_x^2 - I_y^2)(I_{yy} - I_{xx}) - 4I_xI_yI_{xy}}{(K + \frac{I_x^2+I_y^2}{K})^2} \end{aligned} \tag{16}$$

So, we obtained the final PDE in equation [16]. Computing the update scheme and translating it to code can be a difficult task, so I decided to use the update scheme provided by the book which discretizes equation [14] using a method known as 4-nearest-neighbors discretization [1]. This discretization method can be best represented by the figure below:

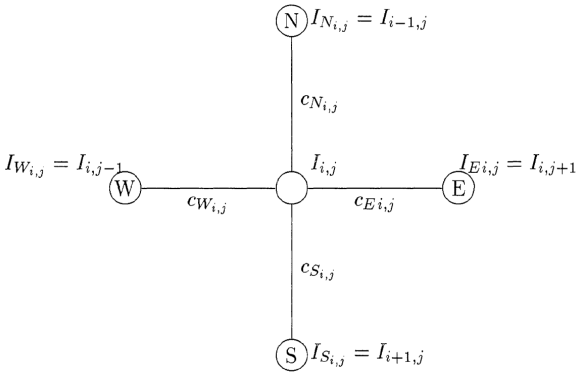

Figure 1: The structure of the discrete computation scheme for the Anisotropic Diffusion PDE [1].

In figure 1 above we can see a presentation of a node $I_{i,j}$ in a square lattice. The four nearest neighbors here are the 4 nodes in the lattice in the North (N), South (S), East (E), and West (W) directions. The nodes in the lattice have brightness values represented as $I_{i,j}$, $I_{i-1,j}$, $I_{i+1,j}$, $I_{i,j+1}$, and $I_{i,j-1}$ and the conduction coefficients c represented by the arcs.

The difference between the brightness values of the nearest neighbors is computed as follows:

$$\Delta_\Omega I_{i,j} = I_{\Omega_{i,j}} - I_{i,j} \tag{17}$$

In equation [17], Ω represents the 4 nearest neighbors in the N, S, E, W direction and Δ represents the difference between the brightness values. The conduction coefficient used for this implementation is the equation [13] as is presented as follows:

$$C^t_{\Omega_{i,j}} = C(|\Delta_\Omega I^t_{i,j}|) = \frac{1}{1 + (\frac{|\Delta_\Omega I^t_{i,j}|}{K})^2} \tag{18}$$

Taking the absolute value is the simplest choice for approximating the norm of the gradient at each arc location. So in equation [18], we take the absolute value of the difference between brightness found from equation [16] as the argument to the function C. The t variable represents the updated conduction coefficient at a particular iteration.

The flux of the brightness through each arc is calculated by multiplying the conduction coefficient of each arc with their corresponding nearest neighbor differences. Using equations [17] and [18], we can represent the flux as follows:

$$\phi_\Omega = C^t_{\Omega_{i,j}} * \Delta_\Omega I_{i,j} \tag{19}$$

Now, we can define the update scheme using the 4-nearest-neighbors discretization, which uses equation [19] as follows:

$$I^{t+1}_{i,j} = I^t_{i,j} + \lambda[\phi_N + \phi_S + \phi_E + \phi_W] \tag{20}$$

In equation [20], the $I^{t+1}_{i,j}$ term stands for the brightness value of a pixel in the image at the iteration $t+1$ and $I^t_{i,j}$ term stands for the brightness value of a pixel in the image at the iteration t. The value of λ varies from $0 \leq \lambda \leq \frac{1}{4}$, which represents the time-step for the update scheme to be stable [1].

So, let's assume that the diffusion coefficient $C^t_{\Omega_{i,j}}$ is a constant 1. So we could express the update scheme as follows [1]:

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda[\Delta_N I_{i,j} + \Delta_S I_{i,j} + \Delta_E I_{i,j} + \Delta_W I_{i,j}] \tag{21}$$

So, in equation [21], we can rewrite the equations using the forward and central difference equations covered in class. Looking at $\Delta_\Omega I_{i,j}$, we can represent all of them as follows [1]:

$$\Delta_N I_{i,j} = I_{i-1,j}^{t} - I_{i,j}^{t} = I(x - \Delta x, y, t) - I(x, y, t) \tag{22}$$

$$\Delta_S I_{i,j} = I_{i+1,j}^{t} - I_{i,j}^{t} = I(x + \Delta x, y, t) - I(x, y, t) \tag{23}$$

$$\Delta_E I_{i,j} = I_{i,j+1}^{t} - I_{i,j}^{t} = I(x, y + \Delta y, t) - I(x, y, t) \tag{24}$$

$$\Delta_W I_{i,j} = I_{i,j-1}^{t} - I_{i,j}^{t} = I(x, y - \Delta y, t) - I(x, y, t) \tag{25}$$

So using equations [21], [22], [23], [24], and [25], we can write the update scheme equation with a constant diffusion coefficient 1 as:

$$I(x,y,t+\Delta t) - I(x,y,t) = \lambda \frac{I(x-\Delta x,y,t) + I(x+\Delta x,y,t) + I(x,y+\Delta y,t) + I(x,y-\Delta y,t) - 4I(x,y,t)}{\Delta x^2} \tag{26}$$

From the equation [26] we can see that it is the same as the Forward Time Central Space Difference equation covered in class that is expressed as:

$$I(x,y,t+\Delta t) - I(x,y,t) = \Delta t \frac{I(x-\Delta x,y,t) + I(x+\Delta x,y,t) + I(x,y+\Delta y,t) + I(x,y-\Delta y,t) - 4I(x,y,t)}{\Delta x^2} \tag{27}$$

Comparing equation [26] and [27], we can see that λ represents the time step Δt, which as explained in the book is $0 \leq \lambda \leq \frac{1}{4}$ [1]. So we can find the CFL condition for equation [27] which represents the update scheme of 2-D Linear Heat equation. The CFL condition can be computed using the following equation:

$$\frac{2b\Delta t}{\Delta x^2} < 1 \rightarrow b\Delta t < \frac{1}{2}\Delta x^2 \tag{28}$$

In equation [28] the variable b which is the diffusion coefficient is set to 1 (constant) for the linear heat equation. The CFL condition would be correct as is for the 1D linear heat equation. However, for the 2D case we have two x and y neighbors so a total of 4 neighbors, which corresponds to the $-4I(x,y,t)$ in equation [27]. So, assuming Δx and Δy are the same, instead of $\frac{1}{2}$, we will get $\frac{1}{4}$ as the coefficient Δx^2. So the CFL condition for 2D linear heat equation will be as follows:

$$\lambda = \Delta t < \frac{1}{4}\Delta x^2 \tag{29}$$

The equation [29] shows the CFL condition computed with a constant (1) diffusion coefficient for the linear heat equation, which is analogous to the CFL condition shown for the case where diffusion coefficient b in equation [18]. So a general represent of the CFL condition for our PDE is as follows:

$$b\Delta t < \frac{1}{4}\Delta x^2 \tag{30}$$

So, for the computational implementation of image denoising, we will generate different types of noise that can be added to an image. Then we will use the update scheme to update the brightness values of the pixels in the image by specifying parameters such as the number of iterations, λ, and K. After running this algorithm, we will be able to denoise the image and evaluate how well the Anisotropic Diffusion PDE reduces the noise in an image using metrics such as PSNR, MSE, and SSIM.

4 Evaluation

For evaluation the performance for the anisotropic diffudion PDE, we first use a sample image and add noise to create a noisy image. For this experiment, I choose to add Salt and Pepper (SP) Noise and Speckle Noise. Then we use the 4-nearest-neighbors discretization in equation [20] referenced from the Geometry-Driven Diffusion in Computer Vision book [1]. Implementing the update equation as code, we then denoise the SP and Speckle noisy images. Then we evaluation the best value of K for a fixed number of iterations (20) and $\lambda = 0.1$ for the denoising of the noisy images. We do this evaluation based on metrics such as Mean-Squared-Error (MSE), Peak-to-Signal-Noise-Ratio (PSNR), and Structural Similarity Index Measure (SSIM).

4.1 Salt and Pepper Noise (SP)

SP noise is also known as an impulse noise. It presents itself in the form or black and white pixel randomly scattered on a clean image making it noisy. It simulates the effect of sprinkling salt and pepper on an image, hence the name "Salt and Pepper Noise". Below is a sample image with and without SP noise added to it:

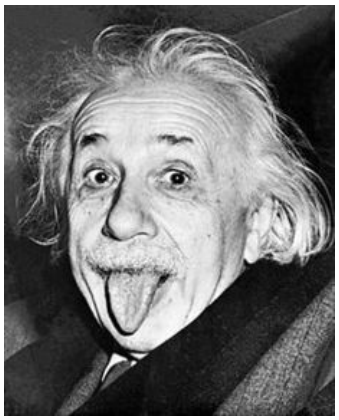

Figure 2: Original Image

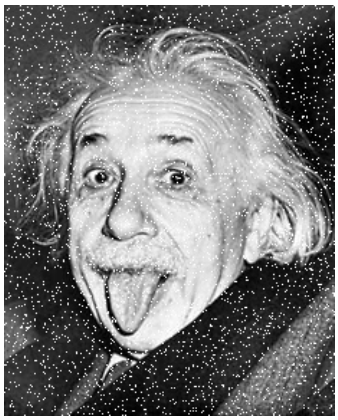

Figure 3: Image with SP noise

4.2 Speckle Noise

Speckle noise, also known as multiplicative noise, is a common noise that occurs in medical images. The noise component is multiplied to each pixel of the image and it has doesn't follow a normal distribution making it difficult to remove from images. This noise is close to the Rayliegh and Gamma Distributions [2]. Below is a sample image with and without Speckle noise added to it:

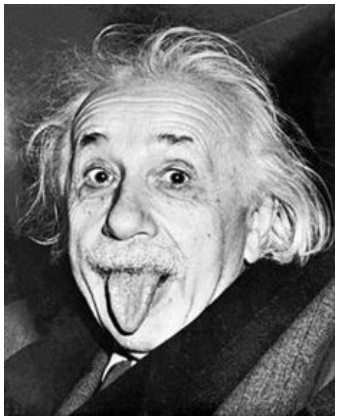

Figure 4: Original Image

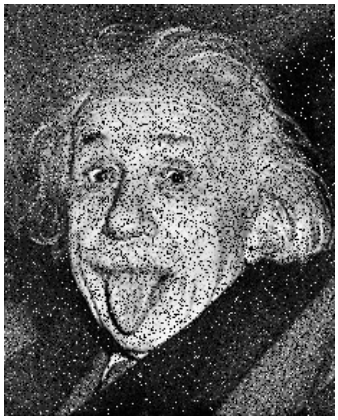

Figure 5: Image with Speckle noise

4.3 MSE

The metric mean squared error, measures the average squared difference between the estimated value and the actual value. The formula for mean squared error can be expressed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(I_{original} - I_{noisy})^2 \tag{31}$$

Using $iterations = 20$ and $\lambda = 0.1$, we can plot the parameter K versus the MSE to identify the best K range for image denoising of both SP and Speckle noisy images. Below are two plots for both noisy images:

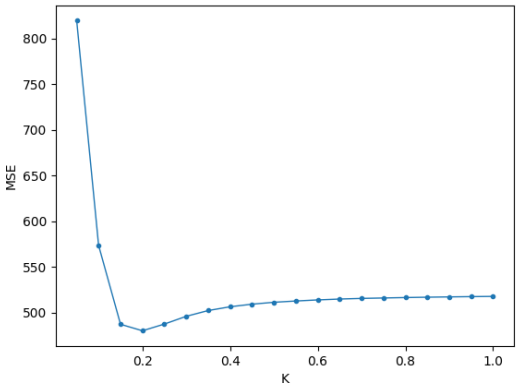

Figure 6: MSE with SP noise for 20 Iterations

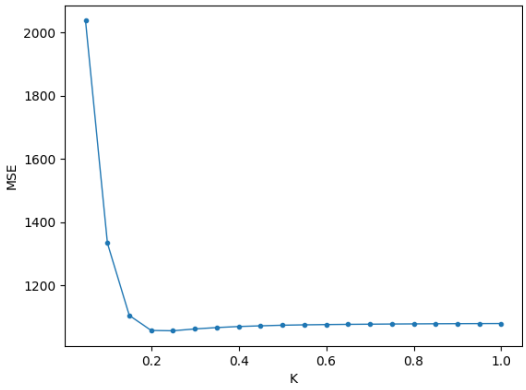

Figure 7: MSE with Speckle noise for 20 Iterations

Based on the two plots above, we can see that MSE for both the SP and Speckle noise image have the best K where the MSE is the lowest between the $0.15 \leq K \leq 0.25$.

4.4 PSNR

The metric peak to signal noise ratio, is a ratio of the maximum power of a signal to the power of the corrupting noise that effects the quality of an image leading to information loss. The formula for peak to signal noise ratio in dB can be expressed as follows:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \tag{32}$$

In equation [32], MAX_I is the maximum pixel value of an image and MSE is defined in equation [31]. Below are two plots for both noisy images using the same parameters as before:

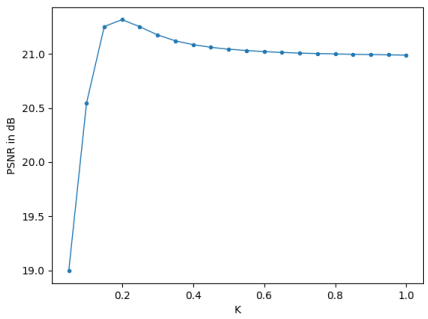

Figure 8: PSNR with SP noise for 20 Iterations

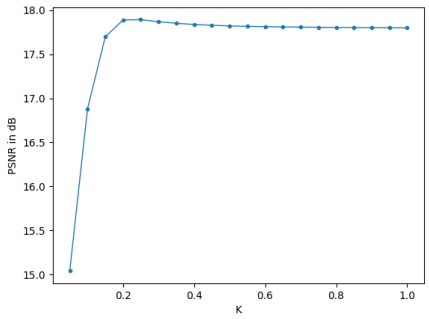

Figure 9: PSNR with Speckle noise for 20 Iterations

Based on the two plots above, we can see that PSNR for both the SP and Speckle noise image have the best K where the PSNR in dB is the highest between $0.15 \leq K \leq 0.25$, same as MSE.

4.5 SSIM

The metric structural similarity index measure, is a method to understand the quality of an image. The formula for structural similarity index measure takes in μ_x, μ_y that represent the pixel mean and σ_x, σ_y that represents the variance of x, y. The covariance of the x and y is $\sigma_{x,y}$, and $c_1 = (k_1 L)^2$, $c_2 = k_2 L)^2$ where L is the dynamic range of the image and k_1, k_2 and 0.01 and 0.03 respectively. The formula of SSIM is expressed as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{33}$$

Below are two plots for both noisy images using the same parameters as before:

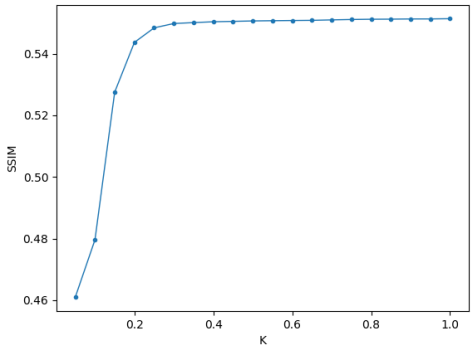

Figure 10: SSIM with SP noise for 20 Iterations

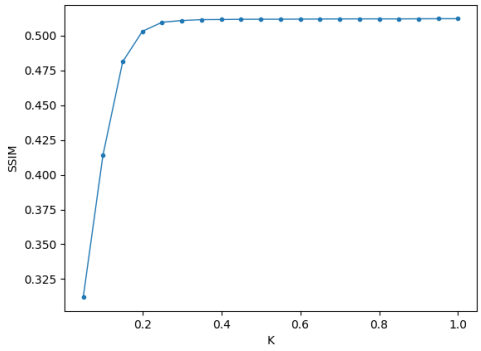

Figure 11: SSIM with Speckle noise for 20 Iterations

Based on the two plots above, we can see that SSIM for both the SP and Speckle noise image have the best K where the SSIM is the highest between $0.25 \leq K \leq 0.35$.

4.6 Denoised Image

Using the results from the plots shown in the previous section, we can see that for the majority of the metrics used for evaluating the best K value, we get a range between $0.15 \leq K \leq 0.25$. So using the $K = 0.2$, $iterations = 20$, and $\lambda = 0.1$, we can see both the denoised image for the noisy SP image and the noisy Speckle image:

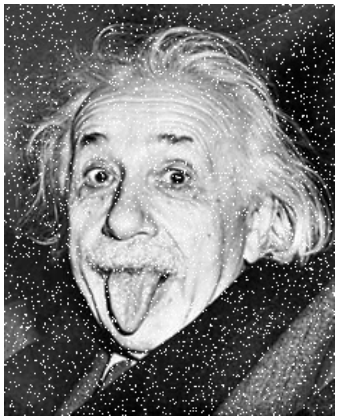

Figure 12: Image with SP Noise

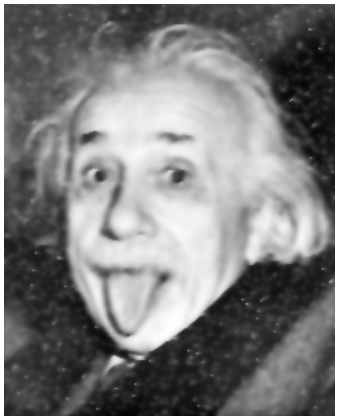

Figure 13: Denoising of SP Noisy Image

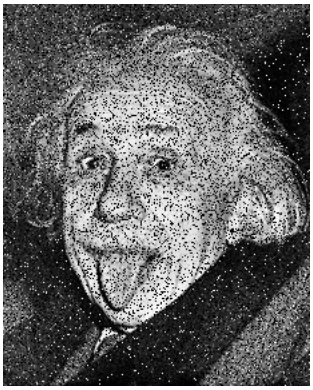

Figure 14: Image with Speckle Noise

Figure 15: Denoising of Speckle Noisy Image

5 Conclusion

In conclusion, we can observe that the anisotropic diffusion PDE is an effective method to remove noise from an image while preserving the fine details of an image. Looking at the denoised images in detail, we can see that the anisotropic diffusion PDE does a good of remove Salt and Pepper noise from an image, while it doesn't perform as well for Speckle Noisy image. As Speckle noise is hard to remove and doesn't have a normal distribution, other advanced noise reduction techniques can be use to improve the noise reduction for a Speckle noisy image. The metrics used to evaluate the best K parameters for the anisotropic diffusion coefficient equation [13] helped in understand what range of values of K are ideal for the update equation. The Geometry-Driven Diffusion in Computer Vision book by Bart M. Haar Romeny [1] helped in understand how to discretize the PDE and gave a good reference point for the computational implementation of the PDE. Some experiments that I could have implemented was to try to generate other types of noise to add to an image such as Random Noise and Poisson Noise as this would have helped in getting a better understand on what kinds of noisy images the anisotropic diffusion PDE performs best.

6 Code

Link to the code on GitHub

References

[1] Proesmans, M., Pauwels, E., van Gool, L. (1994). Coupled Geometry-Driven Diffusion Equations for Low-Level Vision. In: ter Haar Romeny, B.M. (eds) Geometry-Driven Diffusion in Computer Vision. Computational Imaging and Vision, vol 1. Springer, Dordrecht.

[2] Yadav, S. (2020, June 17). Speckle vs gaussian noise? Medium. Retrieved April 25, 2023, from https://medium.com/@sunil7545/speckle-vs-gaussian-noise-7f4f47230d82