Image De-noising using Perona-Malik Diffusion

Siddhi Vishwas Pandare

April 26, 2023

1 Problem Statement

Images are a form of multimedia that conveys information using visual representations. The process of picture de-noising involves taking out the noise from images, which is necessary to enhance their visual quality, make them more usable for analysis and interpretation, and make it possible to accurately extract information. Several factors, including sensor noise, environmental noise, processor noise, and transmission noise, can impart noise into a picture. Image de-noising algorithms must be used to remove noise from images while maintaining the image's key characteristics. These algorithms use a variety of approaches, including filtering, thresholding, and wavelet transforms. This activity is crucial in numerous applications, including remote sensing, computer vision, and medical imaging. In this project, we will perform image de-noising using Perona-Malik diffusion.

2 Mathematical Formulation

Let us assume that the time-evolving image that we have is represented by *I(x,y,t)* where x and y are spatial co-ordinates, and t is the time variable. Now, we want to minimize the noise in the image. Let us focus on removing high-frequency noise such as salt and pepper or line drop noise for now. For penalizing high-frequency noise, we form an energy function that quantifies the gradient of the image. The intuition behind this is that as the image becomes smoother and smoother (de-noising), the gradient or the high-frequency components in the image decrease. Thus, the energy function that we are trying to minimize is given in equation (1)

$$E(z) = \int C(z)dz \tag{1}$$

where z = $||\nabla I||$ and $||.||$ is the L2 norm.

Now, this is the generic form of the energy function. An ideal de-noising would preserve the edges of the image while also smoothing the image. Thus, we want to smooth along the edges but not across them. For this purpose,

we would formulate anisotropic diffusion called Perona Malik diffusion. The intuition is that the gradient at the edge would be large and should ideally be not diffused. Thus, the modification to the heat equation which diffuses homogeneously would be to add a variable diffusion coefficient. This diffusion coefficient is naturally a monotonically decreasing function of ∇I which should be ideally 0 (no diffusion) at the edge and greater than zero at the non-edge parts of the image. The Perona-Malik anisotropic diffusion equation is given in equation(2)

$$I_t = \nabla.(F(||\nabla I||)\nabla I) \tag{2}$$

where $F(.)$ is the diffusion coefficient. Note: In literature, the diffusion coefficient is usually denoted by C however to avoid confusion with equation(1) I have denoted it by F.

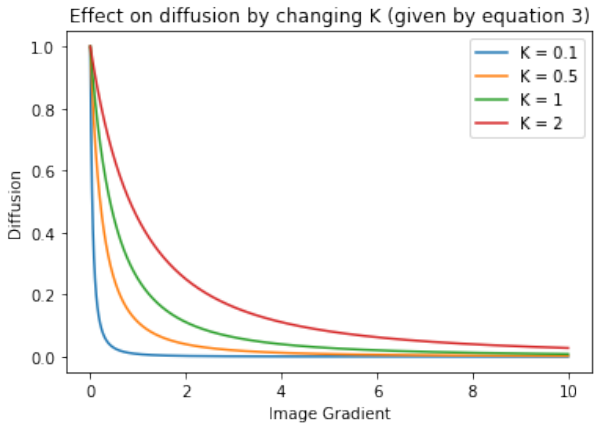
In [1], two diffusion coefficients functions are proposed:

$$F(||\nabla I||) = \frac{1}{1 + (\frac{||\nabla I||}{K})^2} \tag{3}$$

and

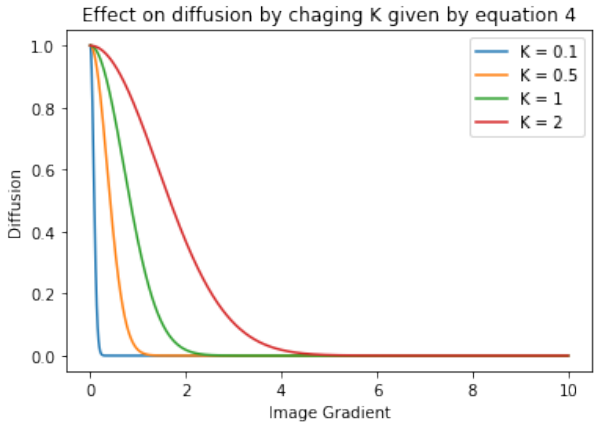
$$F(||\nabla I||) = e^{-(||\nabla I||/K)^2} \tag{4}$$

Figure 1:



where K is the edge sensitivity factor. In this project, we will see how the choice of diffusion coefficient as well as the value of K influences the de-noising of the image.

Figure 2:



3 Discretization & Stability Analysis

3.1 Discretization

In order to run computer algorithms, we will have to discretize the PDE given in equation(2). Let us now simplify equation(2)

$$\begin{aligned} I_t &= \nabla(F(||\nabla I||)\nabla I) \\ &= \nabla F(||\nabla I||).\nabla I + F(||\nabla I||)\Delta I \end{aligned} \tag{5}$$

Let $F = F(||\nabla I||)$ be a shorthand for simplicity,

$$\begin{aligned} I_t &= \nabla F.\nabla I + F\Delta I \\ &= (F_x + F_y).(I_x + I_y) + F(I_x x + I_y y) \\ &= F_x I_x + F_y I_y + F(I_{xx} + I_{yy}) \end{aligned} \tag{6}$$

Now, let us split each addition term and simplify it separately. First by substituting F as in equation (3).

$$\begin{aligned} F_x I_x &= \frac{\partial F}{\partial x} I_x \\ &= \frac{\partial \frac{1}{1+\frac{(I_x+I_y)}{K^2}}}{\partial x} I_x \\ &= \frac{-K^2}{K^2 + I_x^2 + I_y^2}(2I_x^2 I_{xx} + 2I_x I_y I_{xy}) \end{aligned} \tag{7}$$

Similarly, by symmetry, $F_y I_y$ is given below.

$$
\begin{aligned}
F_y I_y &= \frac{\partial F}{\partial y} I_y \\
&= \frac{\partial \frac{1}{1+\frac{(I_x+I_y)}{K^2}}}{\partial y} I_y \\
&= \frac{-K^2}{K^2 + I_x^2 + I_y^2} (2I_y^2 I_{yy} + 2I_x I_y I_{xy})
\end{aligned} \tag{8}
$$

From equations (6,7,8)

$$
\begin{aligned}
I_t &= \frac{(2I_x^2 I_{xx} + 2I_x I_y I_{xy}) + (2I_y^2 I_{yy} + 2I_x I_y I_{xy}) + (K^2 + I_x^2 + I_y^2)(I_{xx} + I_{yy})}{(K^2 + I_x^2 + I_y^2)} \\
&= \frac{(I_{xx} - I_{yy})(I_y^2 - I_x^2) + K^2(I_{xx} + I_{yy}) - 4I_x I_y I_{xy}}{(K^2 + I_x^2 + I_y^2)^2}
\end{aligned} \tag{9}
$$

Similarly, we substitute F as in equation (4).

$$
\begin{aligned}
F_x I_x &= \frac{\partial F}{\partial x} I_x \\
&= \frac{\partial e^{\frac{-(I_x^2+I_y^2)}{K^2}}}{\partial x} I_x \\
&= \frac{-e^{\frac{-(I_x^2+I_y^2)}{K^2}}}{K^2} (2I_x^2 I_{xx} + 2I_x I_y I_{xy})
\end{aligned} \tag{10}
$$

Similarly, by symmetry, $F_y I_y$ is given below.

$$
\begin{aligned}
F_y I_y &= \frac{\partial F}{\partial y} I_y \\
&= \frac{\partial e^{\frac{-(I_x^2+I_y^2)}{K^2}}}{\partial y} I_y \\
&= \frac{-e^{\frac{-(I_x^2+I_y^2)}{K^2}}}{K^2} (2I_y^2 I_{yy} + 2I_x I_y I_{xy})
\end{aligned} \tag{11}
$$

From equations(6,9,10)

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

To code this in a program we now use the forward and central differences to discretize the partial derivatives. With just one previous time step needed to

determine the present time step, the forward difference for time approximation has the advantages of being simple to use and computationally efficient. While to minimize error, the central difference to approximate space derivatives is used.

$$
\begin{aligned}
I_t &= \frac{I(t+\Delta t) - I(t)}{\Delta t} \\
I_x &= \frac{I(x+\Delta x) - I(x-\Delta x)}{2\Delta x} \\
I_y &= \frac{I(y+\Delta y) - I(y-\Delta y)}{2\Delta y} \\
I_{xx} &= \frac{I(x+\Delta x) - 2\Delta I x + I(x-\Delta x)}{\Delta x^2} \\
I_{yy} &= \frac{I(y+\Delta y) - 2\Delta I y + I(y-\Delta y)}{\Delta y^2} \\
I_{xy} &= \frac{I(x+\Delta x, y+\Delta y) + I(x-\Delta x, y-\Delta y) - I(x+\Delta x, y-\Delta y) - I(x-\Delta x, y+\Delta y}{4\Delta x \Delta y}
\end{aligned}
\tag{13}
$$

3.2 Stability Analysis

I will now use the Perona - Malik scheme of discretization given in [1] for this project since doing it by equations (9, 12 and 13) will be too complicated. Also, this method of discretization is widely popular and proved to be stable. I will closely follow the literature [1] to write the code.

Let Δx and Δy be the spatial steps and Δt be the time step. Now, with $\Delta x = \Delta y = 1$ the discretization of Perona - Malik equation is given by,

$$
\begin{aligned}
I(x,y,t+1) = I(x,y,t) + \Delta t[&C_N \nabla_N I(x,y,t) + C_S \nabla_S I(x,y,t) + C_E \nabla_E I(x,y,t) \\
&+ C_W \nabla_W I(x,y,t)]
\end{aligned}
\tag{14}
$$

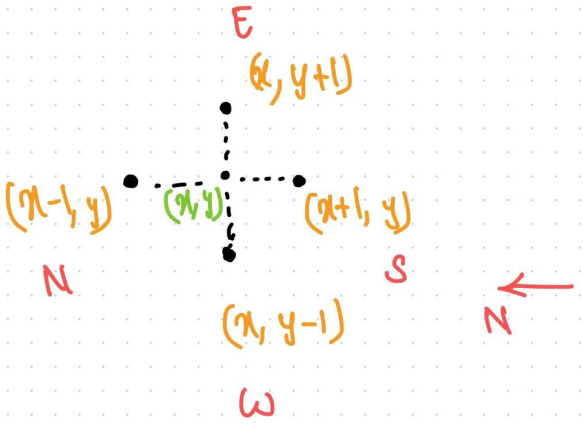
where,

$$
\begin{aligned}
\nabla_N I(x,y,t) &= I(x-1,y,t) - I(x,y,t) \\
\nabla_S I(x,y,t) &= I(x+1,y,t) - I(x,y,t) \\
\nabla_E I(x,y,t) &= I(x,y+1,t) - I(x,y,t) \\
\nabla_W I(x,y,t) &= I(x,y-1,t) - I(x,y,t)
\end{aligned}
\tag{15}
$$

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

An efficient method for determining the CFL condition that avoids the need to compute the 2D DFT of equations (13) is to first calculate the CFL condition

Figure 3: Pictorial representation for equation (15)



for the 1D heat equation, and then extend it to the 2D case due to its symmetry. This approach simplifies the process and saves computational resources. For our case let us define $F(||\nabla I||) = F$ which is the real-valued diffusivity coefficient. Now, the stability condition for the 1D heat equation $I_t = F * I_{xx}$ as derived in class is $\Delta t \leq \frac{(\Delta x)^2}{2F}$. Substituting $\Delta x = \frac{\Delta x}{\sqrt{2}}$, we get the CFL condition for 2D case as

$$\begin{aligned} \Delta t &\leq \frac{(\Delta x)^2}{4F} \\ \Delta t &\leq \frac{(\Delta y)^2}{4F} \end{aligned} \tag{16}$$

Let $\Delta x = \Delta y = 1$

$$\Delta t \leq \frac{1}{4F} \tag{17}$$

Now, from Figures 2 and 3, we see that the diffusivity coefficient is bounded between 0 and 1, thus for any value of K, F will be bounded by $0 \leq F \leq 1$ and $\Delta t \leq 0.25$ will ensure the converge of the scheme. The code for the update equation is given below.

4 Experiments

4.1 Experiment 1: Comparison of Diffusion Performance with varying contrast in images

In this experiment, the objective was to determine when to use equations 3 and 4 as the diffusion coefficient and what properties of the image affect the diffusion

Figure 4: Code Snippet

```
[7]: def F1(dt,K):
         return np.exp(-1* (np.power(dt,2))/(np.power(K,2)))

     def F2(dt,K):
         func = 1/(1 + ((dt/K)**2))
         return func
```

Define anisotropic diffusion function:

```
[8]: def anisodiff_f1(img, steps, K, del_t = 0.25):
         upgrade_img = np.zeros(img.shape, dtype=img.dtype)

         for t in range(steps):
             dn = img[:-2,1:-1] - img[1:-1,1:-1]
             ds = img[2:,1:-1]  - img[1:-1,1:-1]
             de = img[1:-1,2:]  - img[1:-1,1:-1]
             dw = img[1:-1,:-2] - img[1:-1,1:-1]
             upgrade_img[1:-1,1:-1] = img[1:-1,1:-1] + del_t * (F1(dn,K)*dn + F1(ds,K)*ds + F1(de,K)*de + F1(dw,K)*dw)
             img = upgrade_img
         return img
```

```
[9]: def anisodiff_f2(img, steps, K, del_t = 0.25):
         upgrade_img = np.zeros(img.shape, dtype=img.dtype)
         for t in range(steps):
             dn = img[:-2,1:-1] - img[1:-1,1:-1]
             ds = img[2:,1:-1]  - img[1:-1,1:-1]
             de = img[1:-1,2:]  -  img[1:-1,1:-1]
             dw = img[1:-1,:-2] - img[1:-1,1:-1]
             upgrade_img[1:-1,1:-1] = img[1:-1,1:-1] + del_t * (F2(dn,K)*dn + F2(ds,K)*ds + F2(de,K)*de + F2(dw,K)*dw)
             img = upgrade_img
         return img
```

performance. To achieve this, I applied diffusion using the values of F given by equations(3) and (4) to a low and high-contrast image shown below.

Figure 5: Low contrast Image

The high-contrast image was formed by performing histogram equalization

Figure 6: High contrast Image

on the low-contrast image. Some of the features that were enhanced were the eyes, the feather on the hat, and the arc behind the girl.

4.2 Experiment 2: Comparison of Diffusion Performance by Varying values of K

In this experiment, I added salt and pepper noise to both low and high-contrast images (given below) and diffused them using K values ranging from 0 to 10. The objective was to observe the effect of varying K on image smoothing. To quantify the results, I used Peak Signal-to-Noise Ratio to compare the diffused images with the original ones.

5 Results

5.1 Experiment 1

The image below shows the result of diffusing high-contrast and low-contrast images using the two diffusion coefficients while keeping K constant, K = 0.1 for 100 iterations. We can observe that when $\frac{1}{1+(\frac{||\nabla I||}{K})^2}$ is used as the diffusion coefficient, the edges in the high contrast image are still preserved while the edges start to blur for the low contrast image. Thus, we can see that the diffusion coefficient preserves the edges in a high-contrast image.

Figure 7: Low contrast Image with Salt and Pepper Noise

Figure 8: High contrast Image with Salt and Pepper Noise

5.2 Experiment 2

The figures below display the PSNR values obtained by varying the K value for the two images. For low contrast images, F given by equation(3, 4) yields

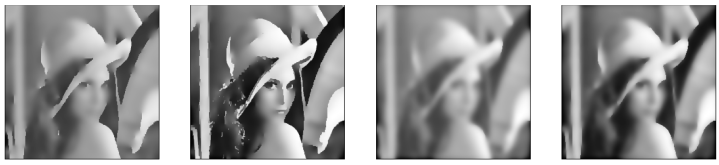


Figure 9: (left to right) Diffusion using equation (3) on low contrast image, Diffusion using equation (3) on high contrast image, Diffusion using equation (4) on low contrast image, Diffusion using equation (4) on high contrast image

high PSNR values for K < 2, while high contrast images result in high PSNR values for K > 1. Notably, the use of two diffusion coefficients demonstrates a difference in smoothing for high-contrast images.

Figure 10:

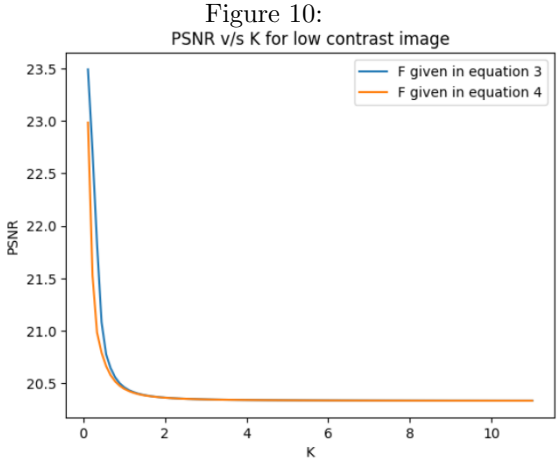
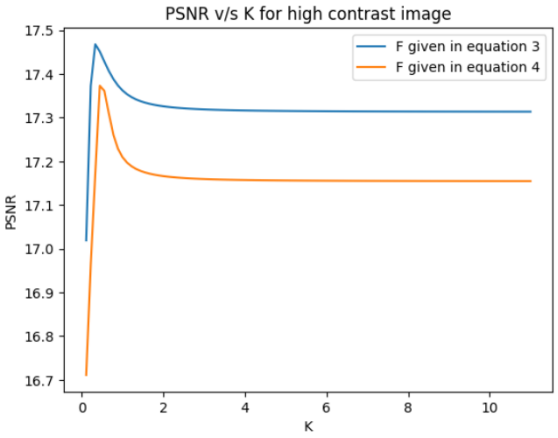


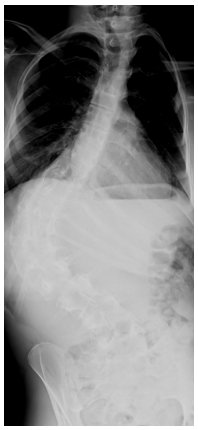
Figure 11:



6 Discussion & Future Work

The Perona Malik equation is a highly effective algorithm for removing high-frequency noise in images, but to preserve edges, it is essential to enhance the image's contrast before applying the de-noising algorithm. The choice of diffusion coefficient is also crucial as it influences the result of the smoothing process. Furthermore, the optimal value of K depends on the image's contrast quality, highlighting the need to investigate its local properties for efficient de-noising. As a potential future work, automating the process of quantifying local properties of images and selecting the best parameters for each use case would be beneficial. For instance, in Figure 12, where the lower half of the image has low contrast and the upper half has high contrast, regional properties of the image could be identified and considered in the de-noising process.

Figure 12: Uneven contrast image



References

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.