ECE 6560 Final Project - Image Smoothing

Alexander Domagala

Spring 2024

1 Problem Description

Although image processing techniques began to appear around the 1960s, the proliferation of inexpensive digital computing power has greatly widened the application pool. Image processing techniques can now be found in areas such as digital photography, medical imaging, and object detection/tracking.

All these applications can be affected by a very common problem: image noise. Noise can render further processing ineffective, thus there exist a variety of approaches by which we can attempt to smooth and denoise images. More traditional denoising techniques may involve fixed convolutional kernels, eg. box blur [2], that indicate how a specific pixel can be updated as a weighted sum of its neighbors. Approaches of this nature will reduce the noise in exchange for the image appearing blurred.

We can instead leverage PDEs to describe how an image should be updated depending on its local characteristics. Approaches that leverage PDEs offer noise reduction while also lessening the blurring that is done to edges.

In this project, we will explore how we can tailor a PDE-based filter to suit a specific image. The goal is that by understanding the characteristics of the image's edges and noise, we will be able to outperform a more generalized PDE-based filter. Our filter will be developed using techniques falling under Anisotropic Diffusion.

2 Mathematical Modeling

2.1 Introduction

Before explicitly developing our PDE, we must first understand the behavior that we want our system to have. We will leverage the Calculus of Variations. This is a powerful framework that will help arrive at a PDE that reflects some desired behavior. Examine the following equation

$$E(I) = \int_0^1 \int_0^1 L(I, I_x, I_y, x, y) dx dy$$

$E(I)$ is the energy functional. Similar to the common understanding of a function, it can output a single value. The key difference is that the input to the energy functional, is another entire function that will depend on traditional variables like x, y, etc.

Note that all notation before section (4) is for continuous space: $x, y \in \mathbb{R}$. Thus, the input to the energy functional can be thought of as an image with an infinite amount of pixels: $I(x, y)$.

During lecture, the following example was introduced. If we want to measure how noisy a certain image is, we could measure the average value of the image's gradient squared.

$$E(I) = \int_0^1 \int_0^1 \frac{1}{2} (\|\nabla_I\|)^2 dx$$

Given two copies of the same image with different amounts of noise, the noiser image will have larger values for $E(I)$ while smoother image will have a smaller values for $E(I)$. The Calculus of Variations says that we can utilize the Euler Lagrange equation $L_I - \frac{\partial}{\partial x} L_{I_x} - \frac{\partial}{\partial y} L_{I_y} = 0$ to solve an optimization problem that minimizes the energy functional $E(I)$. Since we defined the energy functional to measure the average gradient squared, minimization will yield a method by which we can most efficiently reduce this criterion.

In the case of the example, minimization yields the Linear Heat equation

$$I_t = \Delta I$$

The Linear Heat equation optimally reduces the average squared value of the gradient in an image. This means that areas of high gradient in the image will be aggresively smoothed. Hence, the edges (which are areas of higher gradient) are not very well preserved with this technique.

Quadratic Penalty: x^2
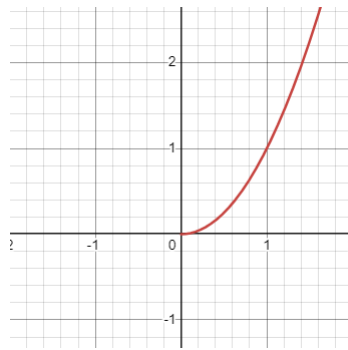
A major improvement is to instead use a linear penalty. Smoothing that takes place at the edges will be less extreme than the previous case. This means that enough diffusion could take place in areas of low gradient such that the image appears to be denoised. Then, the diffusion can be stopped. Although the edges also experience diffusion, the penalty is much more lenient than the previous case. This method is known as Total Variation Diffusion.

Linear Penalty: x

2.2 Anisotropic Diffusion

The previous examples showed that by selecting a certain penalty and then minimizing the associated energy functional, we can control the smoothing behavior of a particular system. The examples can be generalized to what is known as Anisotropic or Perona-Malik Diffusion. We can continue towards the project goal of tailoring a penalty to a specific image by working under this framework.

$$E(I) = \iint C(\|\nabla_I\|)dxdy$$

Note that the penalty $C(\|\nabla_I\|)$ must be strictly increasing for the diffusion to be well-posed. We can then arrive at our PDE via gradient descent

$$L(I, I_x, I_y, x, y) = C(\|\nabla_I\|)$$

$$L_I = 0$$

$$L_{I_x} = \dot{C}(\|\nabla_I\|)\frac{I_x}{\|\nabla_I\|}$$

$$L_{I_y} = \dot{C}(\|\nabla_I\|)\frac{I_y}{\|\nabla_I\|}$$

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

$$I_t = -\nabla_I E$$

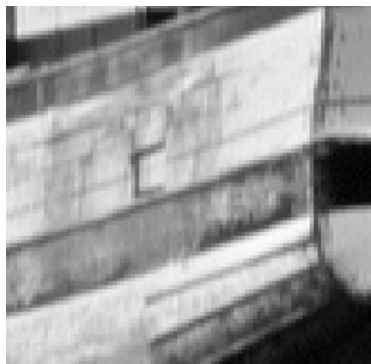$$I_t = -L_I + \frac{\partial}{\partial x}L_{I_x} + \frac{\partial}{\partial y}L_{I_y}$$

$$I_t = \frac{\partial}{\partial x}(\dot{C}(\|\nabla_I\|)\frac{I_x}{\|\nabla_I\|}) + \frac{\partial}{\partial y}(\dot{C}(\|\nabla_I\|)\frac{I_y}{\|\nabla_I\|})$$

$$I_t = \nabla \cdot (\frac{\dot{C}(\|\nabla_I\|)}{\|\nabla_I\|}\nabla I)$$

Now we will work to develop a custom penalty $C(\|\nabla_I\|)$ that supports enhanced edge preservation for specific images.

2.3   Image Profiling

Consider the following images


(a) Original Image

(b) Corrputed Image

We can compute the magnitude gradient at each pixel in the image using the following approximations that are derived in section (4):

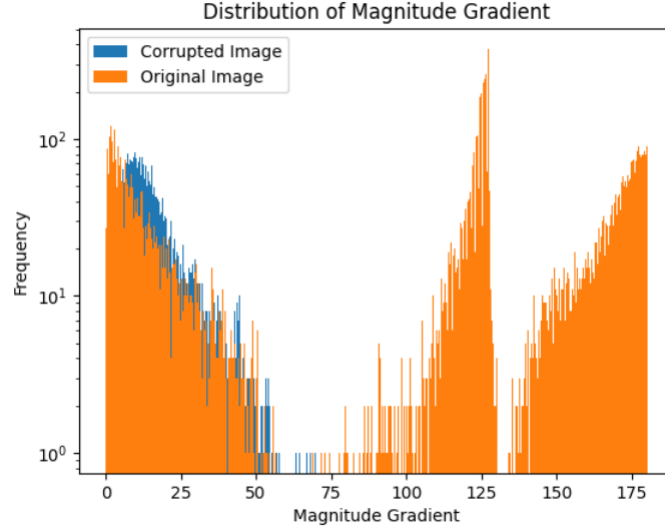$$I_x(x,y,t) = \frac{I(x+\Delta x,y,t)-I(x-\Delta x,y,t)}{2\Delta x}$$

$$I_y(x,y,t) = \frac{I(x,y+\Delta y,t)-I(x,y-\Delta y,t)}{2\Delta y}$$

$$\|\nabla_I\| = \sqrt{I_x^2 + I_y^2}$$

The following image shows the magnitude gradient at each pixel in the noisy image. Edges take on a lighter color while darker sections reflect more uniform areas of the image.
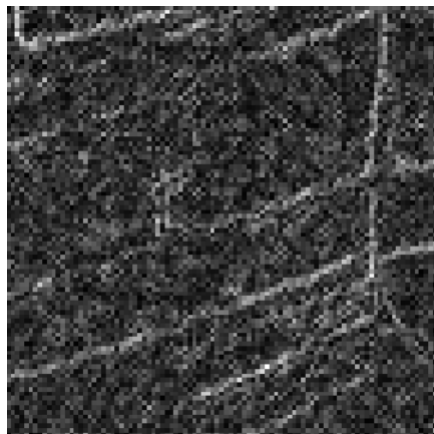
The distribution of the magnitude gradient can be visualized using a histogram.

Distribution of Magnitude Gradient

The corrputed image appears to differ in the range of 10 to 75. Let's shape the energy functional so that pixels in the image that fall in this range of gradients are smoothed. Thus, we can propose the following form for the penalty

$$\text{Sigmoidal Penalty: } \frac{\lambda}{1+e^{-x}}$$
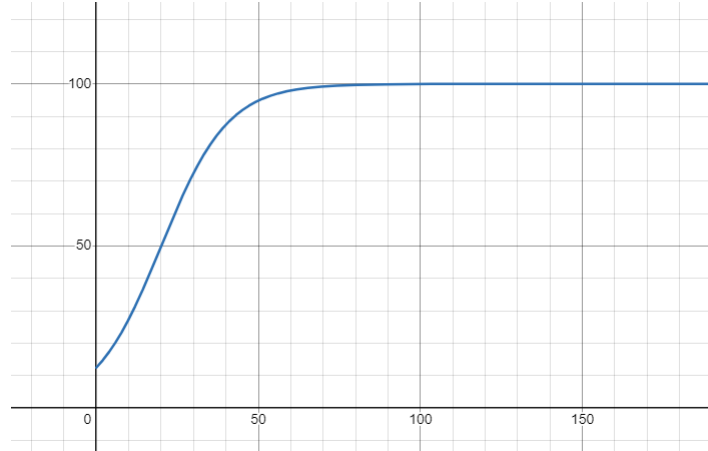
The goal of this penalty is to only have smoothing at the gradients that are corrputing the image. By keeping a nearly flat penalty in the for the higher gradient, we will greatly reduce the amount of diffusion that takes place along the edges.

3   Derivation of PDE

Now, recall the Euler-Lagrange equation

$$L_f - \frac{\partial}{\partial x} L_{f'} = 0 \text{ (1-D)}$$
$$L_I - \frac{\partial}{\partial x} L_{I_x} - \frac{\partial}{\partial y} L_{I_y} = 0 \text{ (2-D)}$$

We can begin working towards obtaining our PDE by setting up a gradient descent

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

We will now have to compute terms L_I, L_{I_x}, L_{I_y} using the previously obtained energy functional

$L(I, I_x, I_y, x, y) = \frac{\lambda}{1+e^\alpha}$, where $\alpha = -\frac{1}{\beta}(\|\nabla_I\|)$

We will also include a term ϵ such that $\|\nabla_I\| = \sqrt{I_x^2 + I_y^2 + \epsilon^2}$

ϵ is a constant used to prevent stability issues and will be examined more closely in section (4).

$L_I = \frac{\partial}{\partial I}(L)$
$L_I = 0$
$L_{I_x} = \frac{\partial}{\partial I_x}(L)$
$L_{I_x} = \frac{\lambda}{\beta} \frac{e^\alpha}{(1+e^\alpha)^2} \frac{I_x}{\sqrt{I_x^2+I_y^2+\epsilon^2}}$
$L_{I_y} = \frac{\partial}{\partial I_y}(L)$
$L_{I_y} = \frac{\lambda}{\beta} \frac{e^\alpha}{(1+e^\alpha)^2} \frac{I_y}{\sqrt{I_x^2+I_y^2+\epsilon^2}}$

Now that we have obtained our expressions for L_{I_x} and L_{I_y}, we must compute their partial derivatives as shown by the Euler-Lagrange equation. This will be shown for $\frac{\partial}{\partial x} L_{I_x}$. $\frac{\partial}{\partial y} L_{I_y}$ will be obtained by examining the expression of $\frac{\partial}{\partial x} L_{I_x}$.

Let ϕ denote $\frac{e^\alpha}{(1+e^\alpha)^2}$ and let γ denote $\frac{I_x}{\sqrt{I_x^2+I_y^2+\epsilon^2}}$. We can begin finding $\frac{\partial}{\partial x} L_{I_x}$ by using the product-rule $\frac{\partial}{\partial x}(\gamma)\phi + \frac{\partial}{\partial x}(\phi)\gamma$. We will start with the left side of the sum. Note that we must include $\frac{\lambda}{\beta}$ in the final expression.

$$\frac{\partial}{\partial x}(\gamma)\phi$$

$$\frac{\partial}{\partial x}\left(\frac{I_x}{\sqrt{I_x^2+I_y^2+\epsilon^2}}\right)\phi$$

$$\left(\frac{I_{xx}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}} + \frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}}(I_x I_{xx} + I_y I_{xy})\right)\phi$$

We can now examine the right side of $\frac{\partial}{\partial x}(\gamma)\phi + \frac{\partial}{\partial x}(\phi)\gamma$.

$$\frac{\partial}{\partial x}(\phi)\gamma$$
$$\frac{\partial}{\partial x}(\frac{e^\alpha}{(1+e^\alpha)^2})\gamma$$
$$\frac{\partial}{\partial x}((e^\alpha)(1+e^\alpha)^{-2})\gamma$$

We see that we will need to again perform the product-rule between (e^α) and $(1+e^\alpha)^{-2}$. Taking the partial derivative of (e^α)

$$-\frac{1}{\beta}e^\alpha \frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}(I_xI_{xx}+I_yI_{xy})$$

Taking the partial derivative of $(1+e^\alpha)^{-2}$

$$-2(1+e^\alpha)^{-3}(-\frac{1}{\beta}e^\alpha \frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}(I_xI_{xx}+I_yI_{xy}))$$

Thus, after factoring common terms, $\frac{\partial}{\partial x}((e^\alpha)(1+e^\alpha)^{-2})$ yields

$$[-\frac{1}{\beta}(e^\alpha)(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})(I_xI_{xx}+I_yI_{xy})][(1+e^\alpha)^{-2}+(e^\alpha)(-2(1+e^\alpha)^{-3})]$$

We have reached the final expression for $\frac{\partial}{\partial x}L_{I_x}$

$$\frac{\partial}{\partial x}L_{I_x}=$$

$$\frac{\lambda}{\beta}[(\frac{I_{xx}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})-(\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}})(I_xI_{xx}+I_yI_{xy})(\frac{e^\alpha}{(1+e^\alpha)^2})+$$

$$(\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})[-\frac{1}{\beta}(e^\alpha)(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})(I_xI_{xx}+I_yI_{xy})][(1+e^\alpha)^{-2}+(e^\alpha)(-2(1+e^\alpha)^{-3})]]$$

$\frac{\partial}{\partial y}L_{I_y}$ can be obtained by modifying $\frac{\partial}{\partial x}L_{I_x}$

$$\frac{\partial}{\partial y}L_{I_y}=$$

$$\frac{\lambda}{\beta}[(\frac{I_{yy}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})-(\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}})(I_xI_{xy}+I_yI_{yy})(\frac{e^\alpha}{(1+e^\alpha)^2})+$$

$$(\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})[-\frac{1}{\beta}(e^\alpha)(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})(I_xI_{xy}+I_yI_{yy})][(1+e^\alpha)^{-2}+(e^\alpha)(-2(1+e^\alpha)^{-3})]]$$

Our final gradient-descent PDE is

$$I_t = \frac{\lambda}{\beta}\left[\left(\frac{I_{xx}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right) - \left(\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}}\right)(I_xI_{xx}+I_yI_{xy})\left(\frac{e^\alpha}{(1+e^\alpha)^2}\right)+\right.$$

$$\left(\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right)\left[-\frac{1}{\beta}(e^\alpha)\left(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right)(I_xI_{xx}+I_yI_{xy})\right][(1+e^\alpha)^{-2}+(e^\alpha)(-2(1+e^\alpha)^{-3})]+$$

$$\left(\frac{I_{yy}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right) - \left(\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}}\right)(I_xI_{xy}+I_yI_{yy})\left(\frac{e^\alpha}{(1+e^\alpha)^2}\right)+$$

$$\left.\left(\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right)\left[-\frac{1}{\beta}(e^\alpha)\left(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}\right)(I_xI_{xy}+I_yI_{yy})\right][(1+e^\alpha)^{-2}+(e^\alpha)(-2(1+e^\alpha)^{-3})]\right]$$

Where $\alpha = -\frac{1}{\beta}(\|\nabla_I\|)$ and $\|\nabla_I\| = \sqrt{I_x^2+I_y^2+\epsilon^2}$

4 Discretization and Implementation

The PDE that was obtained in the previous section is only applicable for continuous time and space variables. We must discretize the PDE so that it can actually be implemented in software.

4.1 Numerical Differentiation

Before explicitly discretizing the PDE, we must also obtain numeric approximations for its partial derivatives. Beginning with the left side of the PDE, let's approximate I_t. It is important to understand that the PDE is defining how the image will be smoothed over successive iterations. In other words, it specifies how the image is updated as time increases. Thus, it is appropriate to use a forward difference.

The forward difference for a single (space) variable can be obtained using the numeric differentiation method shown in lecture

$$f(x,t) = f(x,t)$$

$$f(x,t+\Delta t) = f(x,t) + \Delta t f'(x,t) + O(\Delta t^2)$$

After eliminating $f(x,t)$ terms on the right side of the above equations, we obtain our approximation. Note that the approximation includes an error term $O(\Delta t)$.

$$f'(x,t) = \frac{f(x,t+\Delta t) - f(x,t)}{\Delta t}$$

Expanding this process to two (space) dimensions yields

$$I_t(x,y,t) = \frac{I(x,y,t+\Delta t) - I(x,y,t)}{\Delta t}$$

Continuing with the right side of the PDE, approximations are needed for the following partial derivatives: I_x, I_y, I_{xx}, I_{yy}, and I_{xy}. Since these approximations capture how the image is changing with respect to space, it is more appropriate to use a central difference rather than a forward difference.

Suppose we are trying to approximate I_x. A forward difference would introduce some bias because the value of I_x at a certain point is dependent on an adjacent image value in the positive x-direction only. The central difference will balance out the value of I_x since both directions are considered. The intent is to make the approximation more robust to variations in image content.

The central difference for a single (space) variable can be obtained using the numeric differentiation method shown in lecture

$$f(x,t) = f(x,t)$$

$$f(x + \Delta x, t) = f(x,t) + \Delta x f'(x,t) + O(\Delta x^2)$$

$$f(x - \Delta x, t) = f(x,t) - \Delta x f'(x,t) + O(\Delta x^2)$$

After eliminating $f(x,t)$ terms on the right side of the above equations, we obtain our approximation. Note that the approximation includes an error term $O(\Delta x)$.

$$f'(x,t) = \frac{f(x+\Delta x,t) - f(x-\Delta x,t)}{2\Delta x}$$

Expanding this process to two (space) dimensions yields

$$I_x(x,y,t) = \frac{I(x+\Delta x,y,t) - I(x-\Delta x,y,t)}{2\Delta x}$$

$$I_y(x,y,t) = \frac{I(x,y+\Delta y,t) - I(x,y-\Delta y,t)}{2\Delta y}$$

In order to obtain approximations for I_{xx} and I_{yy}, we can simply add additional terms to the Taylor Series expansion and then repeat the elimination process

$$I_{xx}(x,y,t) = \frac{I(x+\Delta x,y,t) - 2I(x,y,t) + I(x-\Delta x,y,t)}{\Delta x^2}$$

$$I_{yy}(x,y,t) = \frac{I(x,y+\Delta y,t) - 2I(x,y,t) + I(x,y-\Delta y,t)}{\Delta y^2}$$

Finally, we need to obtain an approximation for the mixed partial derivative I_{xy}. This is achieved by taking the central difference approximation for I_x and applying another central difference with respect to y

$$I_x(x,y,t) = \frac{I(x+\Delta x,y,t)}{2\Delta x} - \frac{I(x-\Delta x,y,t)}{2\Delta x}$$

$$I_{xy}(x,y,t) = \frac{\partial}{\partial y}\left(\frac{I(x+\Delta x,y,t)}{2\Delta x}\right) - \frac{\partial}{\partial y}\left(\frac{I(x-\Delta x,y,t)}{2\Delta x}\right)$$

$$I_{xy}(x,y,t) = \frac{1}{2\Delta y}\left(\frac{I(x+\Delta x,y+\Delta y,t)}{2\Delta x} - \frac{I(x+\Delta x,y-\Delta y,t)}{2\Delta x}\right) - \frac{1}{2\Delta y}\left(\frac{I(x-\Delta x,y+\Delta y,t)}{2\Delta x} - \frac{I(x-\Delta x,y-\Delta y,t)}{2\Delta x}\right)$$

$$I_{xy}(x,y,t) = \frac{I(x+\Delta x,y+\Delta y,t) - I(x+\Delta x,y-\Delta y,t) - I(x-\Delta x,y+\Delta y,t) + I(x-\Delta x,y-\Delta y,t)}{4\Delta x\Delta y}$$

4.2 Parameter Selection

Now that we have obtained all the necessary numeric approximations of the partial derivatives, we can move towards defining a scheme that discretizes the PDE. This involves selecting some important parameters.

First, we will examine what is an appropriate time-step Δt. Previously, we defined $I_t(x, y, t) = \frac{I(x,y,t+\Delta t) - I(x,y,t)}{\Delta t}$. In our implementation, Δt will multiply the right side of the PDE. The product will then be added to the current image to obtain the updated image. Thus, we must find a CFL condition so that we can find an appropriate time-step. Recall that during the derivation of the PDE, a term ϵ was included so that $\|\nabla_I\| = \sqrt{I_x^2 + I_y^2 + \epsilon^2}$. Suppose the gradient became very small, all terms I_x and I_y in the PDE would go to zero and we would be left with $I_t = \frac{\lambda}{\beta}[(\frac{I_{xx}+I_{yy}}{(\epsilon^2)^{\frac{1}{2}}})]$. Then we simply have the heat equation

$$I_t = \tfrac{\lambda}{\beta\epsilon}\Delta I$$

We can include our constants with the form of the CFL condition for the two-dimensional linear heat equation as shown in lecture

$$\tfrac{\lambda}{\beta\epsilon} \leq \tfrac{1}{4}\Delta x^2$$

$$\Delta t \leq \tfrac{1}{4}\tfrac{\beta\epsilon}{\lambda}\Delta x^2$$

With a bound set for Δt, we can now shift our focus to Δx and Δy. Since a digital image is composed of a finite number of individual pixels, we will compute the different partial derivative approximations by simply taking our current pixel and applying an offset of +1 or -1 to the current pixel's x and y indices. For example, for I_x

$$I_x(x, y, t) = \tfrac{I(x+\Delta x, y, t)}{2\Delta x} - \tfrac{I(x-\Delta x, y, t)}{2\Delta x}, \text{ we can compute } I_x \text{ at pixel } (a, b) \text{ as}$$

$$I_x(a, b, t) = \tfrac{I(a+1, b, t)}{2(1)} - \tfrac{I(a-1, b, t)}{2(1)}$$

Finally, we will iterate through each pixel in the image and compute how the pixel should be updated based on the PDE.

4.3 Implementation Summary

PDE

$I_t = \frac{\lambda}{\beta}[(\frac{I_{xx}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}) - (\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}})(I_xI_{xx} + I_yI_{xy})(\frac{e^\alpha}{(1+e^\alpha)^2})+$

$(\frac{I_x}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})[-\frac{1}{\beta}(e^\alpha)(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})(I_xI_{xx} + I_yI_{xy})][(1+e^\alpha)^{-2} + (e^\alpha)(-2(1+e^\alpha)^{-3})]+$

$(\frac{I_{yy}}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}}) - (\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{3}{2}}})(I_xI_{xy} + I_yI_{yy})(\frac{e^\alpha}{(1+e^\alpha)^2})+$

$(\frac{I_y}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})[-\frac{1}{\beta}(e^\alpha)(\frac{1}{(I_x^2+I_y^2+\epsilon^2)^{\frac{1}{2}}})(I_xI_{xy} + I_yI_{yy})][(1+e^\alpha)^{-2} + (e^\alpha)(-2(1+e^\alpha)^{-3})]]$

Where $\alpha = -\frac{1}{\beta}(\|\nabla_I\|)$ and $\|\nabla_I\| = \sqrt{I_x^2 + I_y^2 + \epsilon^2}$

Partial Derivatives

$I_t(x,y,t) = \frac{I(x,y,t+\Delta t) - I(x,y,t)}{\Delta t}$

$I_x(x,y,t) = \frac{I(x+\Delta x,y,t)}{2\Delta x} - \frac{I(x-\Delta x,y,t)}{2\Delta x}$

$I_y(x,y,t) = \frac{I(x,y+\Delta y,t) - I(x,y-\Delta y,t)}{2\Delta y}$

$I_{xx}(x,y,t) = \frac{I(x+\Delta x,y,t) - 2I(x,y,t) + I(x-\Delta x,y,t)}{\Delta x^2}$

$I_{yy}(x,y,t) = \frac{I(x,y+\Delta y,t) - 2I(x,y,t) + I(x,y-\Delta y,t)}{\Delta y^2}$

$I_{xy}(x,y,t) = \frac{I(x+\Delta x,y+\Delta y,t) - I(x+\Delta x,y-\Delta y,t) - I(x-\Delta x,y+\Delta y,t) + I(x-\Delta x,y-\Delta y,t)}{4\Delta x \Delta y}$

CFL Condition

$\Delta t \leq \frac{1}{4} \frac{\beta \epsilon}{\lambda} \Delta x^2$

Parameters

$\Delta t, \Delta x, \Delta y, \beta, \epsilon, \lambda$

Format of Implemented PDE

$I(x, y, t + \Delta t) = I(x, y, t) + \Delta t(I_t)$

5 Experiments

Since there are several different hyperparemeters, we will conduct multiple tests to characterize how the custom PDE functions. Additionally, we will include two other PDEs for comparison, the Linear Heat equation and TV Diffusion.

5.1 Parameters

Recall the work in section (2), for the custom PDE, we want to adjust the smoothing so that magnitude gradients in the range 10 to 75 are most affected. The following table summarizes the different parameters that will be used for testing the custom PDE.

Test Number	Linear Heat	TV Diffusion		Sigmoidal				
1	$\Delta t = 0.01$	$\Delta t =$	$\epsilon =$	$\Delta t =$	$\epsilon =$	$\lambda = 1$	$\beta = 4$	$c = 0$
2	$\Delta t = 0.01$	$\Delta t =$	$\epsilon =$	$\Delta t =$	$\epsilon =$	$\lambda = 1$	$\beta = 4$	$c = 0$
3	$\Delta t = 0.01$	$\Delta t =$	$\epsilon =$	$\Delta t =$	$\epsilon =$	$\lambda = 1$	$\beta = 4$	$c = 0$
4	$\Delta t = 0.01$	$\Delta t =$	$\epsilon =$	$\Delta t =$	$\epsilon =$	$\lambda = 1$	$\beta = 4$	$c = 0$

5.2 Control Results
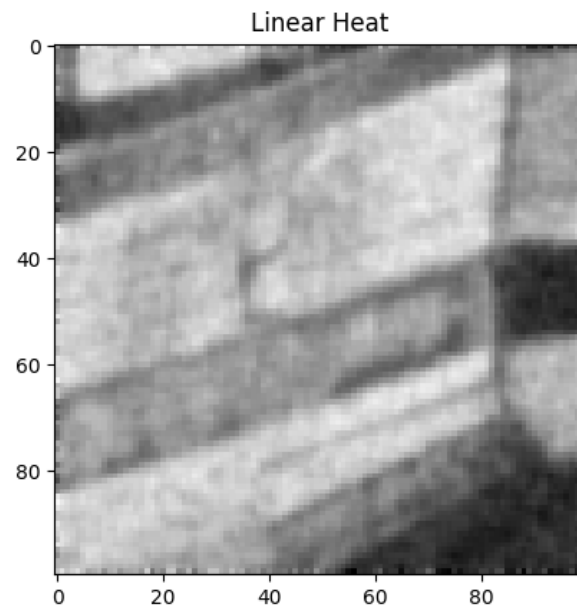
Image results are provided for the Linear Heat Equation and TV Diffusion as a baseline.
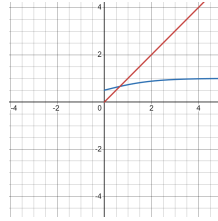

(a) Original Image   (b) Corrputed Image

Linear Heat


TV
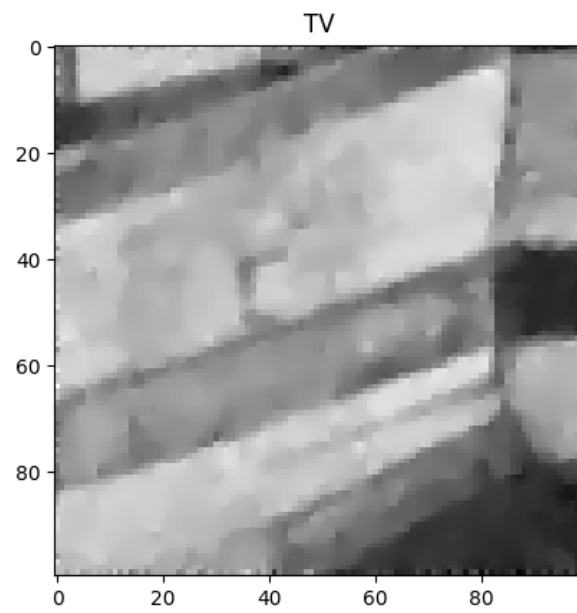
5.3   Experimental Results

All results were generated by running the Evaluation.py script.

1. Baseline Smoothing
In this test, the Custom PDE parameters are all set to one or zero to use a penalty associated with a typical sigmoid.
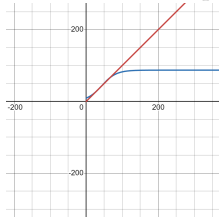

The mean square error is reduced more quickly for the sigmoidal penalty compared to the linear penalty, however the result still shows acceptable edge preservation characteristics.
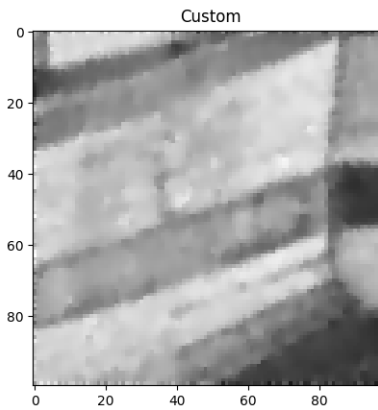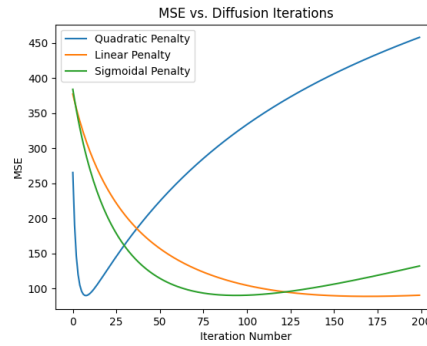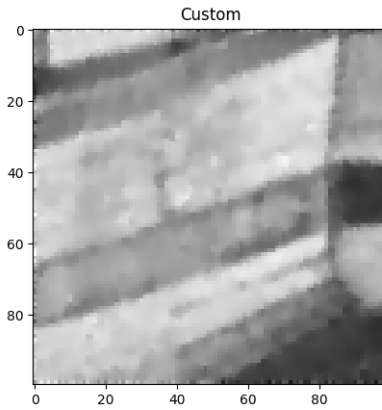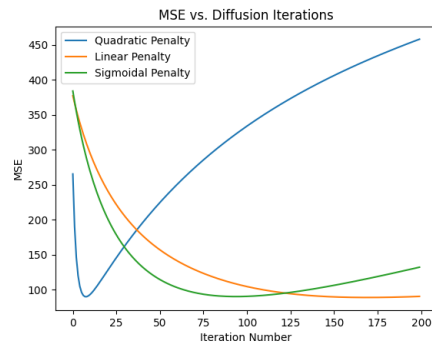
2. Linear Smoothing

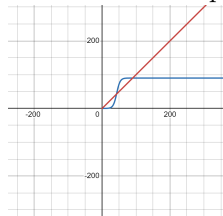Custom PDE slope matching linear penalty for range 10 to 75.

noindent The mean square error is reduced more quickly for the sigmoidal penalty compared to the linear penalty, however the result still shows acceptable edge preservation characteristics.
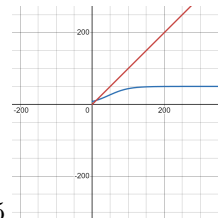
3. Aggressive Smoothing
Custom PDE slope greater than linear penalty for portion of range 10 to 75

4. Tempered Smoothing

Custom PDE slope less than linear penalty for range 10 to 75

6 Conclusion

References

[1] Thomas, L. & Ari, R. d. *Biological Feedback* (CRC Press, USA, 1990).

[2] Thomas, L. & Ari, R. d. *Biological Feedback* (CRC Press, USA, 1990).