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[1], that indicate how a specific pixel can be updated as a weighted sum of its neighbors. Approaches of this nature can 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 us arrive at a PDE that reflects some desired behavior. Examine the following equation

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

$E(I)$ 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 can measure the average value of the image's magnitude gradient squared.

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

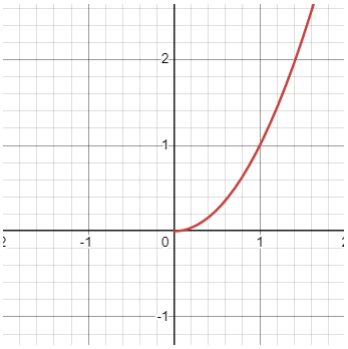
Given two copies of the same image with different amounts of noise, the noiser image will have larger values for $E(I)$ while smoother image will have a smaller values for $E(I)$. The Calculus of Variations says that we can utilize the Euler Lagrange equation $L_I - \frac{\partial}{\partial x}L_{I_x}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

$$I_t = \Delta I$$

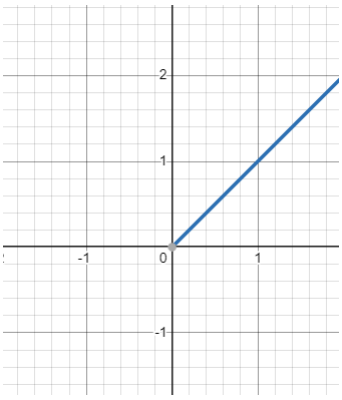
The Linear Heat equation optimally reduces the average squared value of the gradient in an image. This means that areas of high gradient in the image will be aggresively smoothed. Hence, the edges (which are areas of higher gradient) are not very well preserved with this technique.

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 arrive at our PDE via gradient descent

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

$$L_I = 0$$

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

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

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

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

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

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

2.3 Image Profiling

Consider the following images

(a) Original Image

(b) Corrputed Image

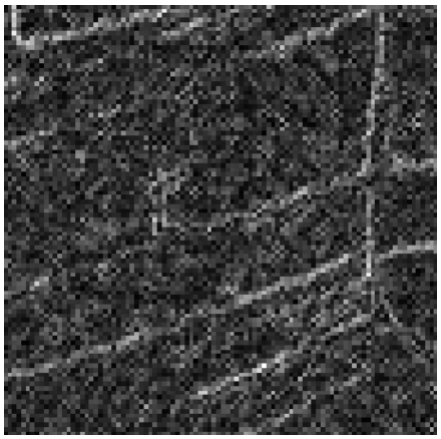
We can compute the magnitude gradient at each pixel in the images by using the following approximations. Note that, these are explained in further detail in section (4):

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

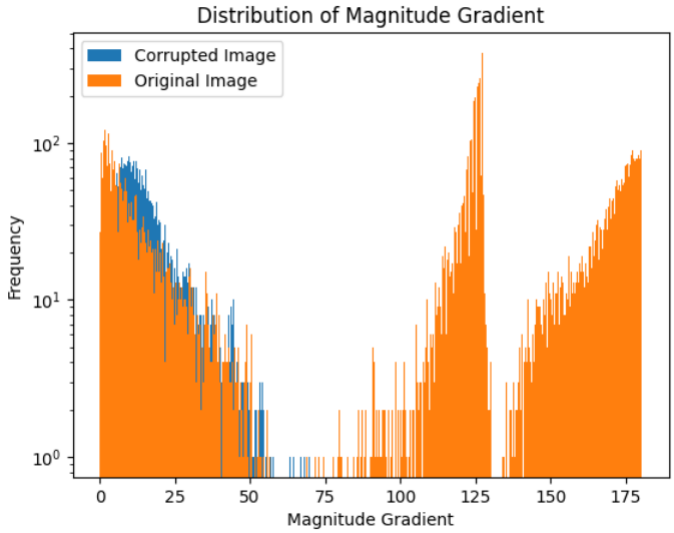
$$I_y(x,y,t) = \frac{I(x,y+\Delta y,t)-I(x,y-\Delta y,t)}{2\Delta y}$$

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

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.

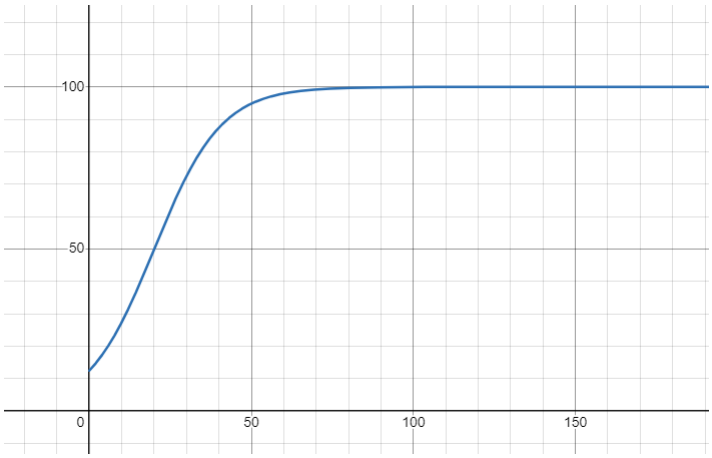


The distribution of the magnitude gradient can be visualized using a histogram.



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

$$\text{Sigmoidal Penalty: } \frac{\lambda}{1+e^{-\frac{1}{\beta}(\|\nabla_I\|-c)}}$$



The goal of this penalty is to only have smoothing at the gradients that are corrupting the image. By keeping a nearly flat penalty for the higher gradients, we will 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 $\alpha = -\frac{1}{\beta}(\|\nabla_I\| - c)$
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.

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

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

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

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

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

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

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

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

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

Supplemental PDE Derivations

The experiment utilizes the Linear Heat and TV Diffusion PDEs for comparisons with the custom PDE. Brief derivations will be shown here.

Linear Heat Equation

$$L(I, I_x, I_y, x, y) = \frac{1}{2}(\|\nabla_I\|)^2$$

$$L_I = 0$$

$$L_{I_x} = I_x$$

$$L_{I_y} = I_y$$

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

$$I_t = I_{xx} + I_{yy}$$

TV Diffusion

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

$$L_I = 0$$

$$L_{I_x} = \frac{I_x}{\sqrt{I_x^2 + I_y^2 + \epsilon^2}}$$

$$L_{I_y} = \frac{I_y}{\sqrt{I_x^2 + I_y^2 + \epsilon^2}}$$

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

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

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

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

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

$f(x,t)$ can be eliminated on the right side of the equation by subtracting the first equation from the second. This introduces $-f(x,t)$ on the left hand side. Finally, we can divide by Δt. Note that the approximation also 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

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

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

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 ensure numeric stability.

Typically, a CFL condition can be found by performing Von Neumann Analysis on a finite difference equation. Due to the complexity of the derived PDE, Von Neumann Analysis is not feasible.

Instead, we can attempt to analyze for what value of $\|\nabla_I\|$ is $\frac{\dot{C}(\|\nabla_I\|)}{\|\nabla_I\|}$ maximized. At this point, we will obtain the strictest CFL condition. Recall that the penalty function $C(\|\nabla_I\|)$ had the form $\frac{\lambda}{1+e^{-\frac{1}{\beta}(\|\nabla_I\|-c)}}$. Taking the derivative with respect to $\|\nabla_I\|$ and multiplying by $\frac{1}{\|\nabla_I\|}$ yields

$$\frac{\lambda e^{-\frac{1}{\beta}(\|\nabla_I\|-c)}}{\beta(1+e^{-\frac{1}{\beta}(\|\nabla_I\|-c)})^2}\frac{1}{\|\nabla_I\|}$$

Graphing the expression as a function of $\|\nabla_I\|$ reveals that as $\|\nabla_I\|$ approaches zero, the value of $\frac{\dot{C}(\|\nabla_I\|)}{\|\nabla_I\|}$ tends towards its maximum of infinity.

Now, 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 a scaled version of the heat equation

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

In lecture, it was shown that the linear heat equation has a CFL condition

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

Thus, we can include our constants to obtain a new CFL condition

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

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

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

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

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

This scheme will be followed for all the spatial derivatives.

We will now be able to 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 $\alpha = -\frac{1}{\beta}(\|\nabla_I\| - c)$ 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

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

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 paremeters, 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. All combinations of parameters satisfy the previously derived CFL condition.

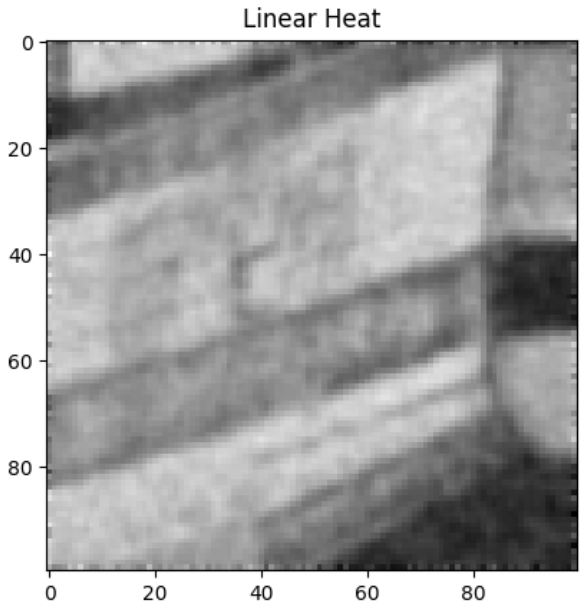
Test Number	Linear Heat	TV Diffusion		Sigmoidal				
1	$\Delta t = 0.05$	$\Delta t = 0.1$	$\epsilon = 4$	$\Delta t = 0.1$	$\epsilon = 4$	$\lambda = 1$	$\beta = 1$	$c = 0$
2	$\Delta t = 0.05$	$\Delta t = 0.1$	$\epsilon = 4$	$\Delta t = 0.1$	$\epsilon = 4$	$\lambda = 87$	$\beta = 20$	$c = 43$
3	$\Delta t = 0.05$	$\Delta t = 0.1$	$\epsilon = 4$	$\Delta t = 0.1$	$\epsilon = 4$	$\lambda = 400$	$\beta = 55$	$c = 150$
4	$\Delta t = 0.05$	$\Delta t = 0.1$	$\epsilon = 4$	$\Delta t = 0.1$	$\epsilon = 4$	$\lambda = 50$	$\beta = 27$	$c = 46$

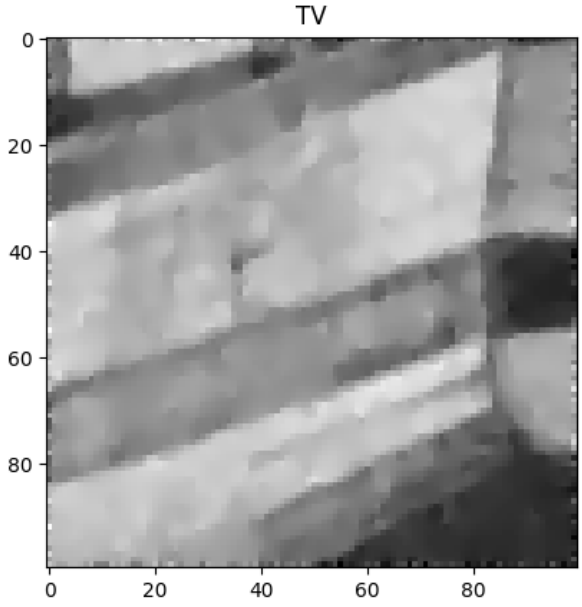
5.2 Control Results

In addition to the original and corrupted versions of the images. Image results are provided for the Linear Heat Equation and TV Diffusion on the following page.

(a) Original Image

(b) Corrputed Image

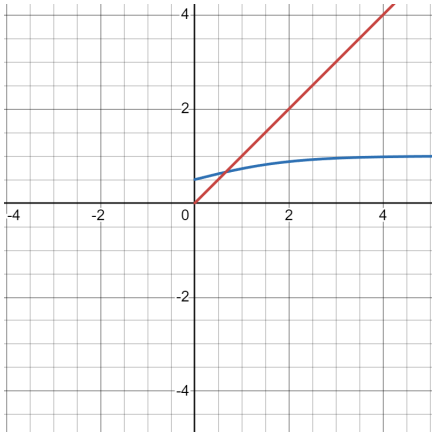

Linear Heat
0
20
40
60
80
0
20
40
60
80


TV
0
20
40
60
80
0
20
40
60
80

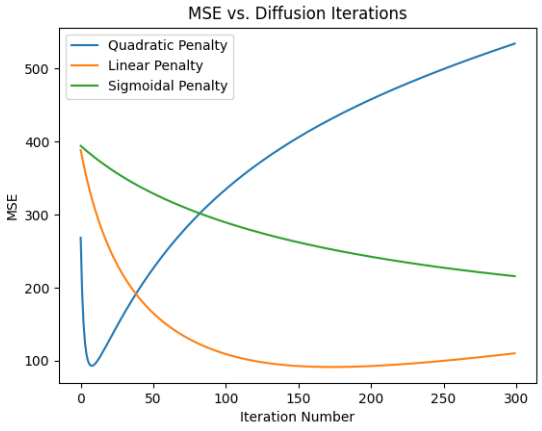
5.3 Experimental Results

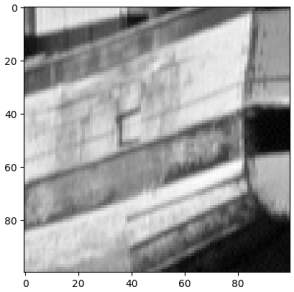
All results were generated by running the Evaluation.py script.

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.

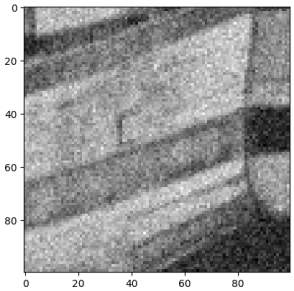


In its base configuration, we see that the sigmoidal penalty is still able to apply some smoothing to the image. However, the MSE is dropping much more slowly than the MSEs associated with the other penalties.

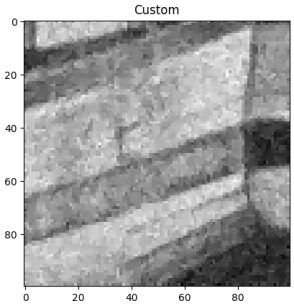




(a) Original Image



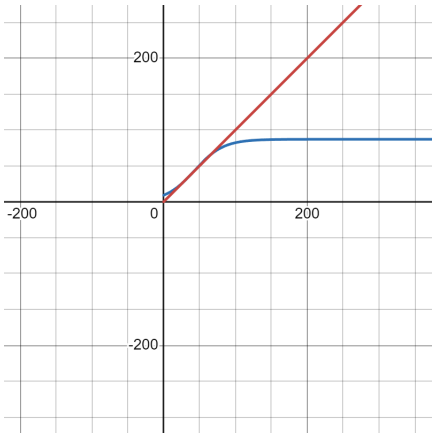
(b) Corrputed Image



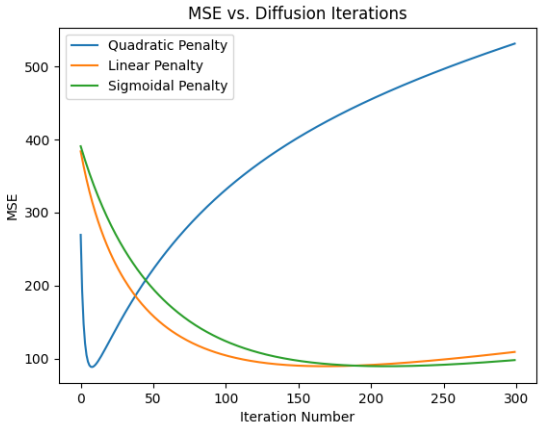
(c) Smoothed Image

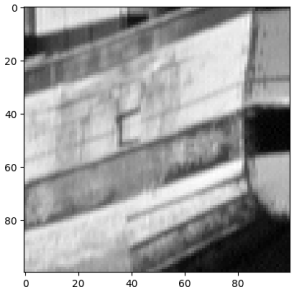
2. "Linear" Smoothing

In this test, the custom PDE parameters are set such that the sigmoidal penalty traces the linear penalty before going flat.

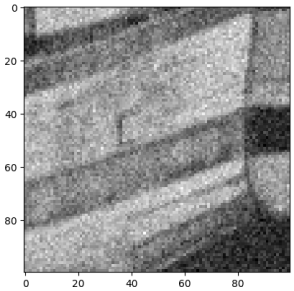


It is very interesting to see how the MSE associated with the sigmoidal penalty closey follows the MSE of the linear penalty. Examining the smoothing result on the next page, we can see that we obtained a much better result than the previous test.

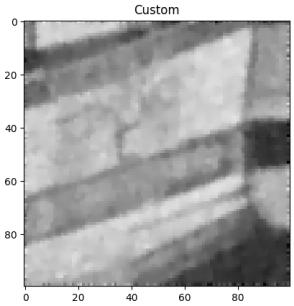




(a) Original Image



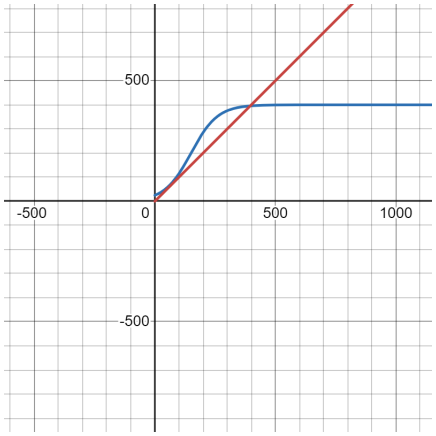
(b) Corrputed Image



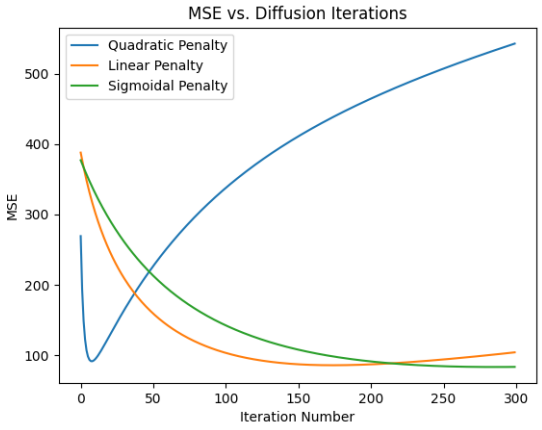
(c) Smoothed Image

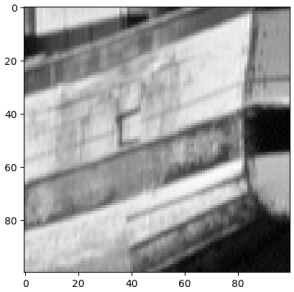
3. Aggressive Smoothing

In this test, the parameters were selected in such a way that the sigmoidal penalty will only begin to go flat around a magnitude gradient value of 200. This coincides with the end of the magnitude gradient values we found during the image analysis.

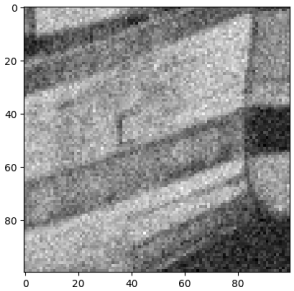


This configuration yielded an excellent result. The MSE associated with the sigmoidal penalty continues to decrease even as we approach the 300th iteration.

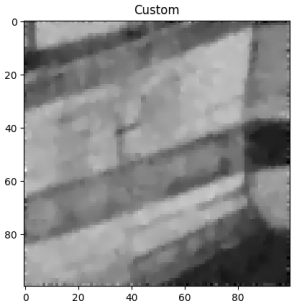




(a) Original Image



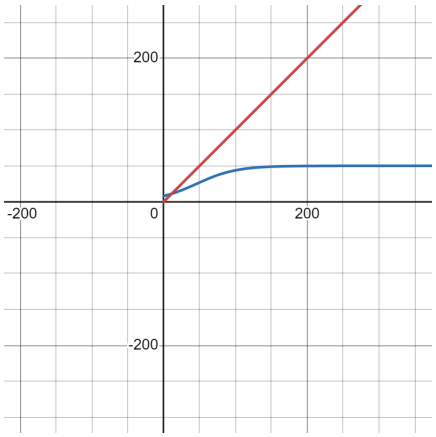
(b) Corrputed Image



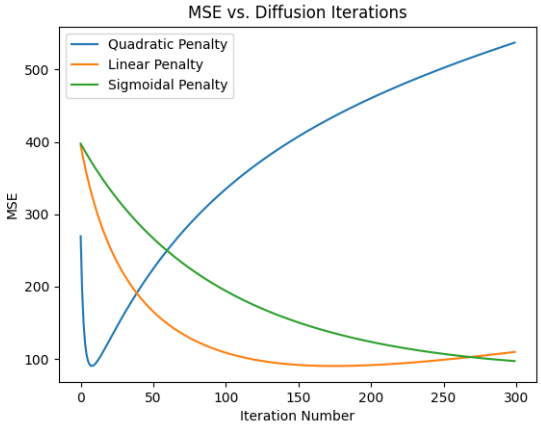
(c) Smoothed Image

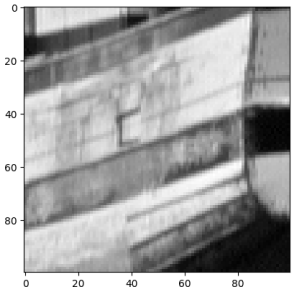
4. Tempered Smoothing

In the final test, we relax the parameters so that the increasing region of the sigmoidal penalty function has a lower slope than the linear penalty.

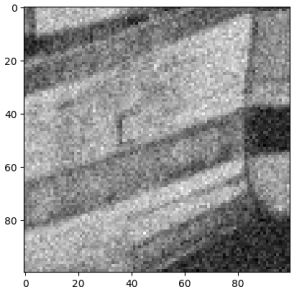


We again see the MSE associated with the sigmoidal penalty decrease more slowly than that of the linear penalty.

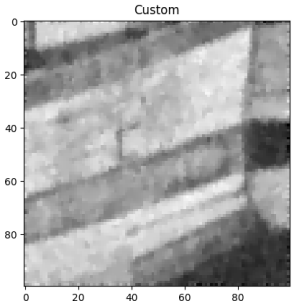




(a) Original Image



(b) Corrputed Image

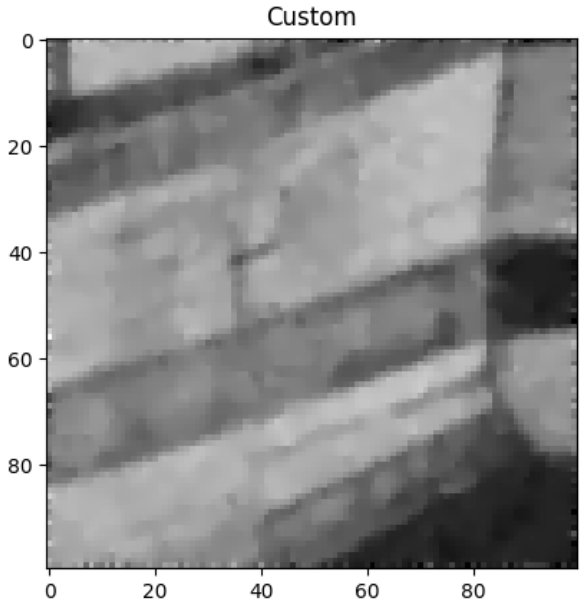


(c) Smoothed Image

Minimum Computed MSE

Test Number	Linear Heat MSE	TV Diffusion MSE	Sigmoidal MSE
1	93.17	91.47	215.79
2	88.33	89.41	89.39
3	91.62	85.91	83.40
4	90.41	90.30	97.13

Test 3 - Best smoothing result:



6 Conclusion

The experiments showed that we were able to successfully denoise images using the custom PDE. From a qualitative perspective, the best smoothing result for the custom PDE has noticeably less blurring than the result obtained from smoothing via the Linear Heat equation. From a quantitative perspective, tests 2 and 3 showed that the custom PDE was able to outperform TV Diffusion on the MSE metric.

Although the PDE showed very good smoothing performance, every single test showed that reaching the minimum MSE required more iterations than the other methods. This was expected as the sigmoidal penalty goes nearly flat after a certain point. This causes the reduced rate of smoothing. Furthermore, the coding implementation requires many more operations than the other methods. Thus, the custom PDE might not be suitable for applications that require the smoothing to take place in a short amount of time.

One challenge that arose while working with the custom PDE was deciding how to balance the various parameters including λ, β, c, etc. The selection of values for the parameters was performed heuristically in order to examine how the varying forms of the penalty function can impact smoothing performance. In the future, an option to sweep parameters and observe the impact on a greater array of quality metrics could lead to new combinations of parameters that improve the smoothing performance of the PDE.

Overall, this project showcased the powerful role PDEs can play in image processing. Previously, my perspective was limited to the idea that image smoothing is simply conducted via convolution and that the smoothing can only be paid for by accepting blurring to the image. Instead, smoothing can be thought of as an iterative diffusion process. The designer of the PDE is able to control how the smoothing effect will behave in different regions of an image.

References

[1] W. Jarosz, "Fast Image Convolutions", 2001. https://web.archive.org/web/20060718054020/http://www.acm.uiuc.edu/siggraph/workshops/wjarosz_convolution_2001.pdf

[2] Harris, C.R., Millman, K.J., van der Walt, S.J. "Array programming with NumPy", Nature 585, 357–362 2020.

[3] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[4] Clark A., "Pillow (PIL Fork) Documentation", 2015. https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf