ECE 6560 : Final Project

Image Denoising with Anisotropic Diffusion

Prajwal Mavinkere Revanna

903386056

Table of Contents

PROBLEM STATEMENT

People capture images all the time for personal, professional. medical or scientific purposes. They are captured using multiple devices such as DSLRs, smartphones, X-Ray machines, and many other. During the process of acquisition from the sensor to the device display, there are a lot of points where noise gets added to the image. This might be because of low light during capture, movement, transmission noise or some other ways.

Noise Removal techniques are used to remove these noises that alter the image. It is a hard task because there are multiple types noises that affect the image, and also because it is extremely important to preserve the details in the image. Noise Removal is a very important part of image analysis and computer vision problems. Preserving the image sharpness on the edges, sharp structures and corners while smoothing is a huge challenge.

In this project, we are exploring the application of Anisotropic Diffusion to do feature preserving image denoising. The Anisotropic Diffusion technique used is also referred to as Perona-Malik Diffusion. It is also based on Finite Differences Method.

We are also comparing the output to other techniques such as FFT based and Gaussian blur based image denoising. We are finding the better of the three based on the contrast of the image (image histogram). We will be basing the better image denoising technique for a particular noise on an image with a particular contrast map on the PSNR value generated when the original and denoised image are compared.

MATHEMATICAL CONCEPTS

Let Ω, belong to R^2 denote the subset of the plane and I(.,t) denote the family of gray scale images. Then anisotropic diffusion is,

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(c(x,y,t)\nabla I\right) = \nabla c \cdot \nabla I + c(x,y,t)\Delta I$$

Δ = Laplacian

∇ = Gradient

div (…) = divergence

c(x, y, t) = diffusion coefficient to control rate of diffusion

Pietro Perona and Jitendra Malik implemented their idea on anisotropic diffusion and proposed their two functions for the diffusion coefficient:

$$c\left(\|\nabla I\|\right) = e^{-(\|\nabla I\|/K)^2}$$

And

$$c\left(\|\nabla I\|\right) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{K}\right)^2}$$

Constant K is used to control the sensitivity to the edges and chosen experimentally or sometimes as a function of the image noise.

DERIVATIONS

Anistropic Diffusion:

The diffusion equation is a PDE usually arising from physical phenomena. It is generally presented in a general form:

$$\partial_t u(x, t) = \nabla(C(u, x, t)\nabla u(x, t))$$

C is the diffusivity, can be scalar, scalar function or scalar (anisotropy).
The equation becomes nonlinear if C depends on u's solution.

If C = 1, it creates a simple linear model of the heat equation, which leads to a Cauchy problem,

$$u_t = \Delta u \text{ for } x \in Rn,\ t > 0$$
$$u(x, 0) = f(x) \text{ for } x \in R_n$$

The idea behind the Perona-Malik equation is a modified heat equation with addition of the diffusivity coefficient depending on spacial activity in a given section of a picture, measured by the norm of the gradient of the local picture. For small gradient norm (homogeneous regions) large diffusivity values are expected, to perform smoothing strongly. In regions with large gradient norm (inhomogeneity) smaller diffusivity is expected, to slow down the diffusion process with delicate image features protection. Hence, the Perona-Malik problem formulation with a Neumann boundary condition is

$$\begin{cases} u_t = \nabla(c(|\nabla u|^2)\nabla u) & \text{in } \Omega \times (0, +\infty) \\ \frac{\partial u}{\partial n} = 0 & \text{in } \partial\Omega \times (0, +\infty) \\ u(x, 0) = u_0(x) & \text{in } \Omega \end{cases}$$

Ω denotes the picture domain. Generally, Ω is a bounded subset of R_n with boundary of class C^1. When looking for C^1 solutions, we assume that the derivatives of u_0 vanish at the Ω boundary. We restrict to diffusivity functions $c(s^2)$ monotonically decreasing from 1 to 0 while s^2 changes from 0 to $+\infty$, infinitely smooth and such that function $\phi(s) = s*c(s^2)$ has one maximum in R. The typical choices are as follows

$$c(s^2) = \frac{1}{1 + \frac{s^2}{\lambda^2}}$$

And

$$c(s^2) = \exp\left(-\frac{s^2}{2\lambda^2}\right)$$

with the parameter $\lambda > 0$.

Flux function is

$$\phi(s) = s*c(s^2).$$

In the one-dimensional case, Perona-Malik equation simplifies to,

$$u_t = (c(u_x^2) + 2 * u_x^2 * c'(u_x^2))\, u_{xx} = \phi'(u_x)*u_{xx}$$

Now it is visible, that for large values of u_x the diffusion coefficient $\phi'(u_x)$ becomes negative, which leads to backward diffusion.

In the two-dimensional case, Perona-Malik equation can be rewritten as:

$$u_t = \phi'(|\nabla u|)u_{\eta\eta} + c(|\nabla u|^2)u_{\xi\xi}$$

Here, the coordinates η and ξ denote the directions parallel and perpendicular to ∇u respectively.

Finite Differences Method:

partial derivative	finite difference approximation	type	order
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_i^n}{\Delta x}$	forward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_i^n - U_{i-1}^n}{\Delta x}$	backward	first in x
$\frac{\partial U}{\partial x} = U_x$	$\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}$	central	second in x
$\frac{\partial^2 U}{\partial x^2} = U_{xx}$	$\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}$	symmetric	second in x
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n+1} - U_i^n}{\Delta t}$	forward	first in t
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^n - U_i^{n-1}}{\Delta t}$	backward	first in t
$\frac{\partial U}{\partial t} = U_t$	$\frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t}$	central	second in t
$\frac{\partial^2 U}{\partial t^2} = U_{tt}$	$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2}$	symmetric	second in t

DISCRETIZATION OF THE PROBLEM:

Consider the energy,

$$E[I] = \int_{\Omega} c(\|\nabla I\|)\, dx\, dy$$

where c : R -> R and is increasing. You can look at it as a penalty: the higher the gradient, the greater the energy. Note that even though the increasing nature of c restricts the class of functions that can be allowed, this model gives more flexibility than the previous.

$$EL = L_I - \frac{\partial}{\partial x}L_x - \frac{\partial}{\partial y}L_y$$

$$= -\frac{\partial}{\partial x}\frac{\partial}{\partial I_x}c\left(\sqrt{I_x^2 + I_y^2}\right) - \frac{\partial}{\partial y}\frac{\partial}{\partial I_y}c\left(\sqrt{I_x^2 + I_y^2}\right) - \frac{\partial}{\partial x}\left(\frac{c(\|\nabla I\|)I_x}{\sqrt{I_x^2 + I_y^2}}\right) - \frac{\partial}{\partial y}\left(\frac{c(\|\nabla I\|)I_y}{\sqrt{I_x^2 + I_y^2}}\right)$$

After regrouping some terms, we end up with the expression

$$\nabla\left(\frac{c'(\|\nabla I\|)}{\|\nabla I\|}\nabla I\right) = 0$$

where the ∇ is the divergence operator. Therefore, the gradient descent PDE becomes

$$I_t = \nabla\left(\frac{c'(\|\nabla I\|)}{\|\nabla I\|}\nabla I\right)$$

Note that if $c(x) = x^2$; $c'(x) = 2x$, you get the heat equation, which will destroy the noise, but also destroys the edges. But we want to preserve the edges. One solution is to let c be the identity operator, i.e. $c(||\nabla I||) = ||\nabla I||$ (actually, it is a modulus operator, but as the norms are always positive, it becomes identity). Then

$$I_t = \nabla\left(\frac{I_{xx}I_y^2 - 2I_xI_yI_{xy} + I_{yy}I_x^2}{(I_x^2 + I_y^2)^{\frac{3}{2}}}\right)$$

This is similar to the geometric heat equation. Note that numerically it is unstable if the derivatives go to zero, so one usually adds an epsilon term.

NOISES ADDED TO THE IMAGE:

Poisson Noise:

The appearance of the Poisson or Shot noise is seen because of the statistical nature of electromagnetic waves such as x-rays, visible lights and gamma rays. The x-ray and gamma ray sources emit a number of photons per unit time.

Gaussian Noise:

Gaussian Noise is a kind of statistical noise with a probability density function equal to the normal distribution, also known as Gaussian Distribution. Random Gaussian function is mixed to Image function to generate the noise. It is called as electronic noise because it forms in amplifiers or detectors.

Salt and Pepper Noise:

Salt and Pepper noise is introduced to an image by adding both random bright (with 255) and random dark (with 0) all across the image. This model is also called as data drop noise because it drops the original data values where random white and dark points are introduced.

Speckle Noise:

Speckle is a kind of granular noise that inherently exists in an image which degrades the image's quality. It can be generated by multiplying random values with different pixels across the image.

OTHER DENOISING TECHNIQUES COMPARED:

FFT based Image Denoising:

The Fast Fourier transform is applied to the noisy image to transform the image from the spatial to frequency domain. A LPF and a HPF are designed. Then the transformed image is filtered with LPF and HPF. Now the inverse FFT is used to the low-pass filtered image and high-pass filtered image. Soft thresholding is then applied to the inverse of high-pass image to enhance the sharpness of the image. The PWL filter or Lagrange or spline interpolated PWL filter is applied to the inversion of the low-pass image, to enhance the image smoothness. Then the resulting images are combined to get the denoised image.

Gaussian Filtering based Image Denoising:

The Gaussian filtering is an important space among the weighted mean filters. It is based on the Gaussian function shape for selection of the right value of linear smoothing filter. It uses the Gaussian function of discrete 2-D by zero-mean to be the smoothing filter.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

METRICS USED FOR COMPARISON:

Mean Squared Error(MSE):

The mean squared error (MSE) of two images measures the average of the squares of the errors (difference in pixels) — that is, the average squared difference between the denoised values and original values.

Peak Signal to Noise Ratio(PSNR):

The higher the PSNR, the better denoised image matches the original image and the better the denoising algorithm. This occurs because we want to minimize the MSE between images with respect the maximum signal value of the image.

SOFTWARE IMPLEMENTATION:

Language Used – Python
Tool – Jupyter Notebook

Anisotropic Diffusion Denoising:

```
def anisodiff(im, steps, b, l = 0.25):
    temp = np.zeros(im.shape, dtype=im.dtype)
    for t in range(steps):
        dn = im[:-2,1:-1] - im[1:-1,1:-1]
        ds = im[2:,1:-1] - im[1:-1,1:-1]
        de = im[1:-1,2:] - im[1:-1,1:-1]
        dw = im[1:-1,:-2] - im[1:-1,1:-1]
        temp[1:-1,1:-1] = im[1:-1,1:-1] +\
                          l * (f1(dn,b)*dn + f1(ds,b)*ds +
                                  f1(de,b)*de + f1(dw,b)*dw)
        im = temp
    return im
```

FFT based Image Denoising:

```
def denoise_fft(im, keep_fraction):
    F = np.fft.fft2(im)
    ff = F.copy()
    r,c = ff.shape
    ff[int(r*keep_fraction):int(r*(1-keep_fraction))] = 0
    ff[:,int(c*keep_fraction):int(c*(1-keep_fraction))] = 0
    im_new = np.fft.ifft2(ff).real
    return im_new
```

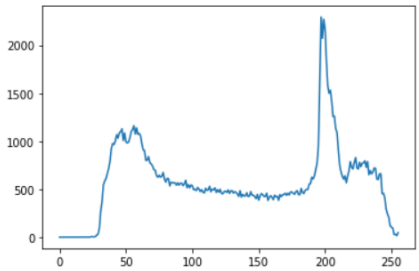
Gaussian Filtering based Image Denoising:

```
def gblur(a):
    kernel = np.array([[1.0,2.0,1.0], [2.0,4.0,2.0], [1.0,2.0,1.0]])
    kernel = kernel / np.sum(kernel)
    arraylist = []
    for y in range(3):
        temparray = np.copy(a)
        temparray = np.roll(temparray, y - 1, axis=0)
        for x in range(3):
            temparray_X = np.copy(temparray)
            temparray_X = np.roll(temparray_X, x - 1, axis=1)*kernel[y,x]
            arraylist.append(temparray_X)

    arraylist = np.array(arraylist)
    arraylist_sum = np.sum(arraylist, axis=0)
    return arraylist_sum
```

EXPERIMENTAL RESULTS

Image 1 –



Different noises and Anisotropic Diffusion:

1) **Poisson Noise**

2) **Gaussian Noise**

3) Salt and Pepper Noise

4) Speckle Noise

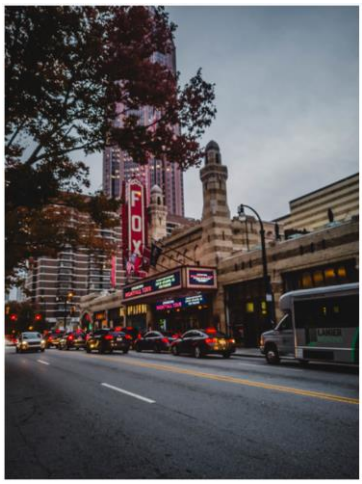
Different noises and FFT based Denoising:

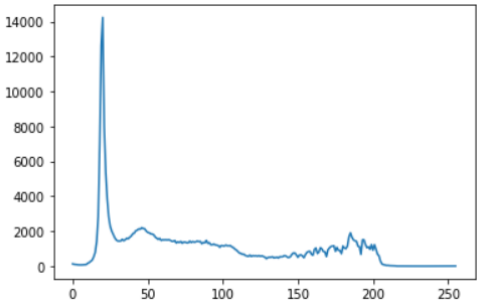
Different Noises and Gaussian Filter based Denoising:

Comparison:

Noise/Denoising	**Anisotropic Diffusion**	**Anisotropic Diffusion**	**FFT**	**FFT**	**Gaussian**	**Gaussian**
	MSE	*PSNR*	*MSE*	*PSNR*	*MSE*	*PSNR*
Poisson	0.0037	72.44	0.0063	70.13	0.0039	72.24
Gaussian	0.0064	70.08	0.0076	69.35	0.0050	71.14
Salt & Pepper	0.0072	69.60	0.0062	70.22	0.0038	72.35
Speckle	0.287	53.61	0.052	60.38	0.060	61.04

Image 2 –





Different noises and Anisotropic Diffusion:

1) Poisson Noise

2) Gaussian Noise

3) Salt and Pepper Noise

4) Speckle Noise

Different noises and FFT based Denoising:

Different Noises and Gaussian Filter based Denoising:

Comparison:

Noise/Denoising	Anisotropic Diffusion	Anisotropic Diffusion	FFT	FFT	Gaussian	Gaussian
	MSE	*PSNR*	*MSE*	*PSNR*	*MSE*	*PSNR*
Poisson	0.0016	76.05	0.0027	73.76	0.0015	76.39
Gaussian	0.0033	72.99	0.0041	71.98	0.0027	73.77
Salt & Pepper	0.0044	71.69	0.0027	73.74	0.0015	76.33
Speckle	0.100	58.12	0.025	64.12	0.021	64.79

CONCLUSION

Based on the testing on multiple types of images with different histograms(contrast), Gaussian Denoising was consistently better than the other two. Sometimes the results varied based on the type of noise.

For Poisson and Gaussian Noise, the anisotropic diffusion based denoising gave out very good results, comparable to that of the Gaussian denoised output. It was better than the FFT implementation.

For Salt and Pepper Noise, the anisotropic denoising was worse than both of the other denoising, but very small PSNR difference between the three.

For Speckle Noise, the anisotropic diffusion yielded far worse results in the metrics over FFT and Gaussian denoising, though on normal observation, the denoised output better than the other two.

FUTURE WORK

Look at other implementations of PDE based image denoising and work on bigger image sets on different contrasts to end with a stronger conclusion. We should explore other image comparison metrics for more in depth nuances.

Creating a comparison between different PDE based approaches of denoising to find the better and more efficient one and compare that to other approaches.

REFERENCES

1) https://medium.com/image-vision/noise-in-digital-image-processing-55357c9fab71

2) https://www.researchgate.net/publication/285457935_Wavelet_and_FFT_Based_Image_Denoising_Using_Non-linear_Filters

3) https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf

4) http://www.cs.man.ac.uk/~fumie/tmp/introductory-finite-difference-methods-for-pdes.pdf

5) https://en.wikipedia.org/wiki/Finite_difference_method

6) http://ijarcst.com/doc/vol3issue2/ver2/ankita.pdf - *Image Denoising with Various Filtering Techniques*

7) https://www.ni.com/en-us/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html

8) https://arxiv.org/abs/1412.6291v1 - *Perona-Malik equation and its numerical properties*

9) https://en.wikipedia.org/wiki/Mean_squared_error