Contents

Chapter 1. Image Processing

1.1 Basic Mechanisms

We see a picture. In it, there are friends, family, famous landmarks, and other significant images. They are what we try to record and remember. In short, we mostly focus on three characteristics: who, what, and where. Although machines can also recognize the defining qualities of an image, their processes of recognition differ significantly from ours. Analyzing the pixels belonging to an object of interest is, essentially, how a machine can decipher image content. From a scientific point of view, machines try to remove noise and detect edges or objects. Finally, the machines store and present these deciphered images. In the core of this paper, I will not only present some image processing algorithms for denoising, but I will also analyze the performance of each algorithm on a test image.

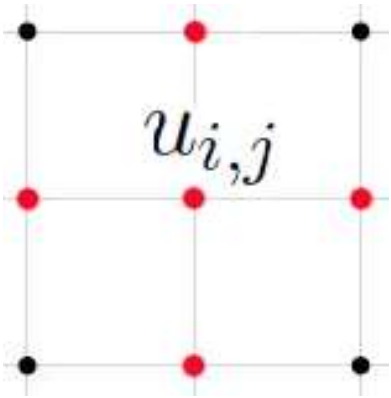
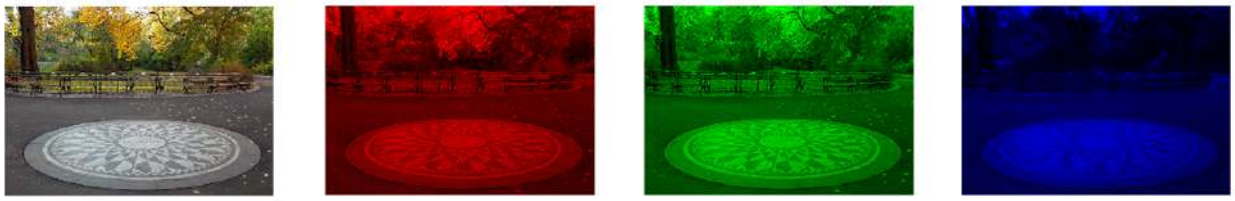


Figure 1.1: Structure of Grayscale image matrix

All digital image processing represents images as a matrix or 2D array. With grayscale image, we model images as function $u : \Omega \to [0, 1]$ where $\Omega = [0, 1]^2$ denote the image domain, and $u(x, y)$ denotes the intensity or brightness of the image where x, y represents locations. On a computer, we can store a discrete version of f, sampled on a uniform two dimensional grid, and process the discrete or digital version of the image. In Matlab, this two dimensional array is organized into a matrix. For color images, we model images as function $u : \Omega \to [0, 1]^3$, where $u(x, y)$ is now a vector in $\mathbb{R}^3$ consisting of the RGB values of the image at location (x, y). This vector can also contain YUV values or values from any other color space. In Matlab, these data are organized into a three dimensional array of size $M \times N \times 3$. I will only perform floating point precision arithmetic on each image processing algorithm.



(a) Original Image (b) Red Channel (c) Green Channel (d) Blue Channel

Figure 1.2: Each Channels of RGB

1.2 Noise

Image noise refers to random variations in brightness or color information within images. I will introduce processes illustrating the denoising of these images in **Chapter 2**. Mathematically, we model noise using random variables. Let $u : \Omega \to [0, 1]$ denote a grayscale image. We consider additive noise to be of the form

$$u_N(x, y) = u(x, y) + \eta(x, y)$$

where $\eta(x, y)$ is a random variable for each $(x, y) \in \Omega$ and u_N denotes the noisy image. We consider additive Gaussian noise, which is obtained by taking $\eta(x, y)$ to be a Gaussian random variable of standard deviation σ for each (x, y). Then this is implemented in Matlab as follows:

```
f = imread(Image); % Convert to matrix
f = im2double(f);  % Cast value type to double
[row, col, channel] = size(f); % Take size of f
Noise = f + 0.1*randn(row, col, channel); % Create a noise image
```

(a) σ=0.00, Original image (b) σ=0.05

(c) σ=0.15 (d) σ=0.25

Figure 1.4: Noise increasing as λ gets bigger

1.3 Blur

A blurred image can result from either motion of camera as it operates or improper focal distancing, among other reasons. For a blurring kernel h, blur is defined as

$$f_B(x,y) = \int_{-\infty}^{\infty} h(x-s, y-t) \cdot f(s,t) ds dt$$

The Matlab command **fspecial** provides various types of blur kernels. With this kernel, and the command **imfilter**[2], the image can be blurred. For example, the code below blurs an image of the Strawberry Field in Central Park, NY. The keyword 'motion' returns a filter that, once combined with the image, approximates the linear motion of a camera by lens pixels, where the camera moves an angle of theta degrees in a counterclockwise direction. The filter is a vector representing horizontal and vertical motion. For the kernel h defined above, it can be formed by matrix where black has a value of 0 as follows:

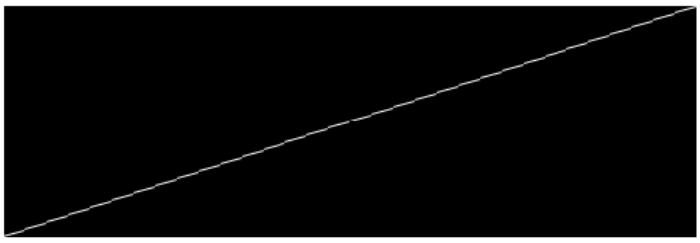


Figure 1.5: Behave of kernel h

We consider in this paper motion blur, implemented in Matlab as follows:

```
1 f = imread(Image);
2 h = fspecial('motion', 15, 15);
3 b = imfilter(f, h, 'replicate');
```

(a) Original image (b) Blurred image

Figure 1.6: Blurred image

Chapter 2. Partial Differential Equations

Let $f : \Omega \to [0, 1]$ be an image that has been corrupted by noise. The noisy image is restored by seeking to find an image u that minimizes the energy $E[u]$ given by the following equation

$$E\left[u\right] = \min_{u} \frac{1}{2} \int_{\Omega} \phi(|\nabla u|^2)\, dxdy + \frac{\lambda}{2} \int_{\Omega} (u - f)^2 dxdy \tag{2.1}$$

The energy function $E[u]$ has two terms as can be seen from its expression. The first term is free of noise and hence, makes $E[u]$ smooth while the second term is fidelity term and hence, introduces the noise in the image. There are different algorithms for denoising or restoring images and are based on $E[u]$ for different choices of ϕ. Since these three can be interpreted as gradient descent equations for the function E, the schemes will be implemented based on gradient descent equations and $E[u]$.

Let $g(u, u_x, u_y) = \phi(|\nabla u|^2) + \frac{\lambda}{2}(u-f)^2$. Then introduce ∇E that behaves like gradient and $E[u]$ has a extrema where ∇E=0. Furthermore, ∇E points in the direction of maximum positive rate of change. Note that $|\nabla u|^2 = u_x + u_y$. By Euler-Lagrange equation,

$$\begin{aligned}
\nabla E &= \frac{\partial}{\partial u} - \frac{\partial}{\partial x}\frac{\partial g}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial g}{\partial u_y} \\
&= \lambda(u - f) - \frac{\partial}{\partial x}\frac{1}{2}\phi'(u_x^2 + u_y^2)2u_x - \frac{\partial}{\partial y}\frac{1}{2}\phi'(u_x^2 + u_y^2)2u_y \\
&= \lambda(u - f) - \frac{\partial}{\partial x}\phi'(u_x^2 + u_y^2)u_x - \frac{\partial}{\partial y}\phi'(u_x^2 + u_y^2)u_y \\
&= -\lambda(f - u) - \mathrm{div}\Big(\phi'(u_x^2 + u_y^2)(u_x + u_y)\Big) \\
&= -\Big[\lambda(f - u) + \mathrm{div}\Big(\phi'(|\nabla u|^2)(\nabla u)\Big)\Big]
\end{aligned}$$

To find $\min_{u} E[u]$, let Δt be a small step size in the direction of $-\nabla E$ and then

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t(-\nabla E) \tag{2.2}$$

This equation can be modified as

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \mathrm{div}\Big(\phi'(|\nabla u|^2)\nabla u\Big) + \lambda(f - u) \\
&= -\nabla E
\end{aligned} \tag{2.3}$$

The last equation represents the gradient descent. The gradient descent for heat equation, Perona-Malik, and total variation are determined by different choices of ϕ, which is as follows:

$$\begin{cases}
\phi(s) = s & \text{Heat Equation} \\
\phi'(s) = c(s) & \text{Perona-Malik. c is flux function, detailed in } \textbf{Chapter 2.2} \\
\phi(s) = \sqrt{s} & \text{total variation}
\end{cases}$$

2.1 Heat Equation

As described above, heat equation takes $\phi(s) = s$. By Taylor's expansion, we can expand $u(x+h)$ and $u(x-h)$ as follows:

$$\begin{aligned} u(x+h) &= u(x) + hu'(x) + \frac{h^2}{2}u''(x) + O(h^3) \\ u(x-h) &= u(x) - hu'(x) + \frac{h^2}{2}u''(x) + O(h^3) \end{aligned} \tag{2.4}$$

Adding two equations in (2.3) follows that

$$u(x+h) + u(x-h) = 2u(x) + h^2u''(x) + O(h^3)$$

Solving for u", the following equation is obtained

$$u''(x) = \frac{1}{h^2}\Big[u(x+h) - 2u(x) + u(x-h)\Big] + O(h) \tag{2.5}$$

Based on (2.4), because u was defined as multivariable function in x and y, approximation of Δu is

$$\begin{aligned} \Delta u &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \\ &\approx \frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) \end{aligned} \tag{2.6}$$

This is the standard 5-point stencil for the Laplacian. The gradient descent for heat equation is given by

$$\frac{\partial u}{\partial t} = \Delta u + \lambda(f - u)$$

So the equation (2.2) can be modified as

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} + \lambda(f - u)\big)$$

The scheme is stable when $\Delta t \leq 0.25$. Note that it is well-known that the heat equation excessively blurs edges and removes important image details.

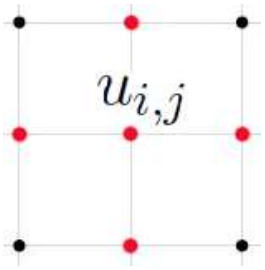


Figure 2.1: Five stencils for heat equation. Red represents used stencils for heat equation

2.2 Perona-Malik Equation

Perona-Malik propose to perform edge detection and noise removal via anisotropic diffusion [4]. Based on the equation (2.2), gradient descent is determined by $\lambda = 0$ and $\phi'(s) = c(s)$ where c is the flux function which is chosen locally as a function of a magnitude of the gradient of the image and defined as follows:

$$c(|\nabla u|) = \begin{cases} \exp(-|\nabla u|^2/k^2) & \text{exponential} \\ \big(1 + |\nabla u|^2/k^2\big)^{-1} & \text{fractional} \end{cases}$$

where k is a constant. The simplified divergence operator is given by

$$\begin{aligned} \frac{\partial u}{\partial t} &= \operatorname{div}\Big(c\big(|\nabla u|^2\big)\nabla u\Big) \\ &= \frac{\partial}{\partial x}\Big(c(x,y,t)u_x\Big) + \frac{\partial}{\partial y}\Big(c(x,y,t)u_y\Big) \\ &= \underbrace{c_x u_x + c_y u_y}_{(A)} + \underbrace{c(u_{xx} + u_{yy})}_{(B)} \end{aligned} \tag{2.7}$$

(A) is discretized using centered differences.

$$\begin{aligned} c_x u_x + c_y u_y &= \frac{c_x}{2}(u_{i+1,j} - u_{i-1,j}) + \frac{c_y}{2}(u_{i,j+1} - u_{i,j-1}) \\ &= \frac{c_x}{2}(u_{i+1,j} - u_{i,j} + u_{i,j} - u_{i-1,j}) + \frac{c_y}{2}(u_{i,j+1} - u_{i,j} + u_{i,j} - u_{i,j-1}) \end{aligned} \tag{2.8}$$

where $u_x \approx \frac{1}{2}(u_{i+1,j} - u_{i-1,j})$ and $u_y \approx \frac{1}{2}(u_{i,j+1} - u_{i,j-1})$

For (B), we use the five point stencil for the Laplacian is used.

$$\begin{aligned} c(u_{xx} + u_{yy}) &= c_{i,j}(u_{i+1,j} + u_{i-1,j} - 2u_{i,j} + u_{i,j+1} + u_{i,j-1} - 2u_{i,j}) \\ &= c_{i,j}(u_{i+1,j} - u_{i,j} + u_{i-1,j} - u_{i,j} + u_{i,j+1} - u_{i,j} + u_{i,j-1} - u_{i,j}) \end{aligned} \tag{2.9}$$

Defining the vectors

$$\begin{aligned} \overrightarrow{N} &= u_{i-1,j} - u_{i,j} \\ \overrightarrow{S} &= u_{i+1,j} - u_{i,j} \\ \overrightarrow{E} &= u_{i,j+1} - u_{i,j} \\ \overrightarrow{W} &= u_{i,j-1} - u_{i,j} \end{aligned}$$

and applying the vectors to the flux equation,

$$c_v = \begin{cases} \exp(-v^2/k^2) & \text{exponential} \\ \big(1 + v^2/k^2\big)^{-1} & \text{fractional} \end{cases}$$

where $v \in \{\overrightarrow{N}, \overrightarrow{S}, \overrightarrow{E}, \overrightarrow{W}\}$ and $N, W, E,$ and S represent pixel above, left, right, below when it comes to an image matrix space. Then using equations (2.8) and (2.9), the gradient descent is

now modified as

$$c_x u_x + c_y u_y + c(u_{xx} + u_{yy}) = (\overbrace{c_{i,j} + \frac{c_x}{2}}^{c_S})(\overbrace{u_{i+1,j} - u_{i,j}}^{\overrightarrow{S}}) + (\overbrace{c_{i,j} - \frac{c_x}{2}}^{c_N})(\overbrace{u_{i-1,j} - u_{i,j}}^{\overrightarrow{N}}) + (\underbrace{c_{i,j} + \frac{c_y}{2}}_{c_E})(\underbrace{u_{i,j+1} - I_{i,j}}_{\overrightarrow{E}}) + (\underbrace{c_{i,j} - \frac{c_y}{2}}_{c_W})(\underbrace{u_{i,j-1} - u_{i,j}}_{\overrightarrow{W}})$$

A 4-nearest neighbors discretization of the Laplacian operator which is suggested by Perona-Malik is

$$u_{i,j}^{t+1} = u_{i,j}^{t} + \mu\Big[c_N \overrightarrow{N} + c_S \overrightarrow{S} + c_E \overrightarrow{E} + c_W \overrightarrow{W}\Big]_{i,j}^{t}$$

where $0 \leq \mu \leq 0.25$ for stability of numerical schemes [4].

2.3 Total variation

Rudin, Osher, and Fetami proposed to restore the noisy image f by using $\phi(s) = \sqrt{s}$. Then gradient (2.3) for total variation are modified as follows:

$$\frac{\partial u}{\partial t} = \operatorname{div}\Big(\frac{\nabla u}{|\nabla u|}\Big) + \lambda(f - u)$$

where λ be a positive parameter [1, 5]. Now take $u(0, x) = u_0(x)$ be the initial image on $\Omega \times \{t = 0\}$. Let $u_x = \frac{\partial u}{\partial x}$ and $u_y = \frac{\partial u}{\partial y}$. Then expanding the divergence,

$$\operatorname{div}\Big(\frac{\nabla u}{|\nabla u|}\Big) = \frac{\partial}{\partial x}\Big(\frac{u_x}{|\nabla u|}\Big) + \frac{\partial}{\partial y}\Big(\frac{u_y}{|\nabla u|}\Big)$$

Now let $u_{i,j}$ denote point on a grid at ih and jh where h be the step size. Note that ∇_x^+ and ∇_y^+ are the forward differences in x and y direction and ∇_x^- and ∇_y^- are the backward differences in x and y direction, respectively. Define $m(a, b)$ as

$$m(a, b) = \Big[\frac{sign(a) + sign(b)}{2}\Big] \cdot \min(|a|, |b|)$$

A discrete numerical solution which is suggested by Getreuer [1] can be derived as

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t\Big[\nabla_x^-(M_1) + \nabla_y^-(M_2) + \lambda(f_{i,j} - u_{i,j}^{n})\Big]$$

where $u_{0,j}^n = u_{1,j}^n$, $u_{i,0}^n = u_{i,1}^n$, $u_{i,N}^n = u_{i,N-1}^n$, $i, j = 0, \ldots, N$, and

$$M_1 = \frac{\nabla_x^+ u_{i,j}^n}{\sqrt{(\nabla_x^+ u_{i,j}^n)^2 + (m(\nabla_y^+ u_{i,j}^n, \nabla_y^- u_{i,j}^n))^2}}$$

$$M_2 = \frac{\nabla_y^+ u_{i,j}^n}{\sqrt{(\nabla_y^+ u_{i,j}^n)^2 + (m(\nabla_x^+ u_{i,j}^n, \nabla_x^- u_{i,j}^n))^2}} \quad i, j = 1, \ldots, N-1$$

Chapter 3. Experimental Results

In the previous chapter, some PDE-based schemes for the image restoration test were presented. Now, the results of applying these schemes to real images will be shown and then their performance will be analysed. What is the highest quality image, and how it can be determined? Let u represent the restored image and I represent the original image, both having a size of an $M \times N$ image matrix. The Mean Squared Error (MSE) is defined by

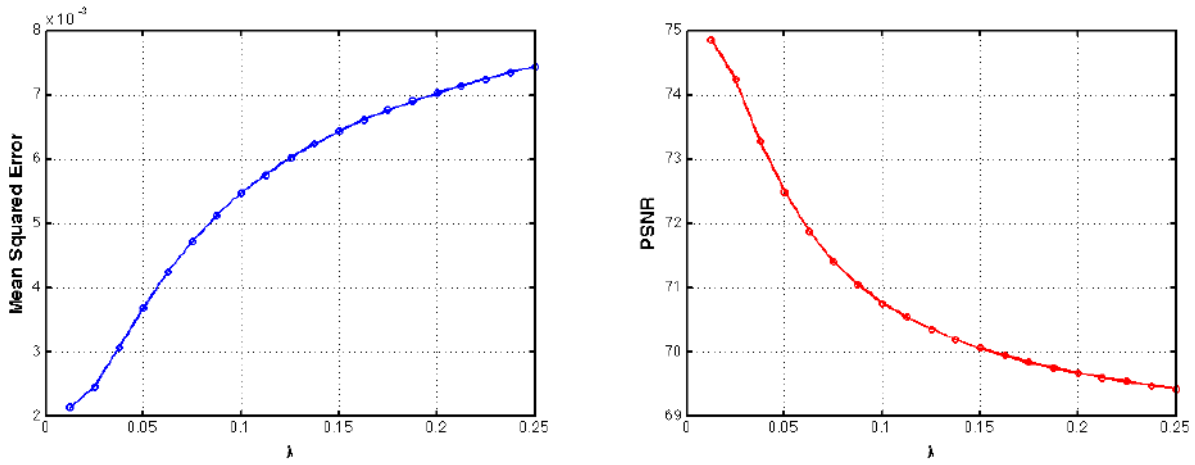
$$\text{MSE} = \sum_{m=1}^{M} \sum_{n=1}^{N} \big(I - u\big)^2$$

Peak Signal to Noise Ratio (PSNR) can then be defined as

$$\begin{aligned} PSNR &= 10 \log \Big(\frac{MAX_I^2}{MSE}\Big) \\ &= 20 \log \big(MAX_I\big) - 10 \log \big(MSE\big) \end{aligned} \tag{3.1}$$

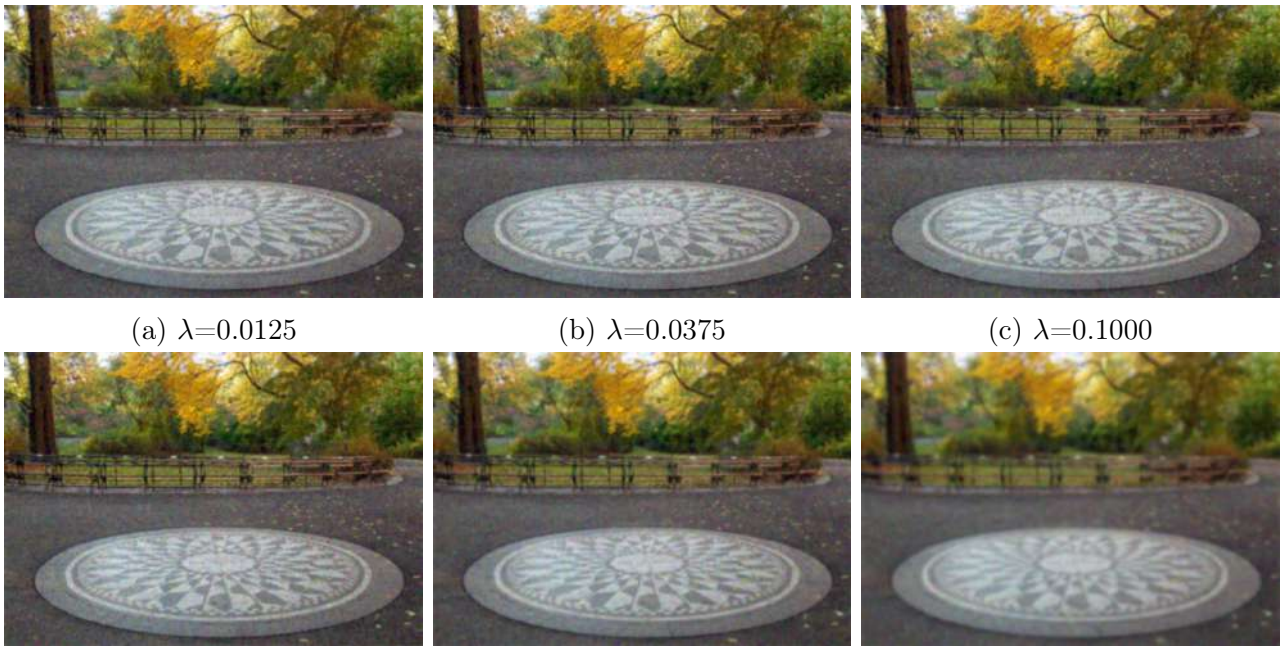
where MAX_I is the maximum pixel value of image I and PSNR is measured in decibels. The Mean Squared Error represents the average of the squares of the errors between the original and restored images. This error is the amount by which the values of the original image differ from the degraded image [3], which implies that higher PSNRs construct higher quality images.

3.1 Heat Equation



MSE-λ PSNR-λ

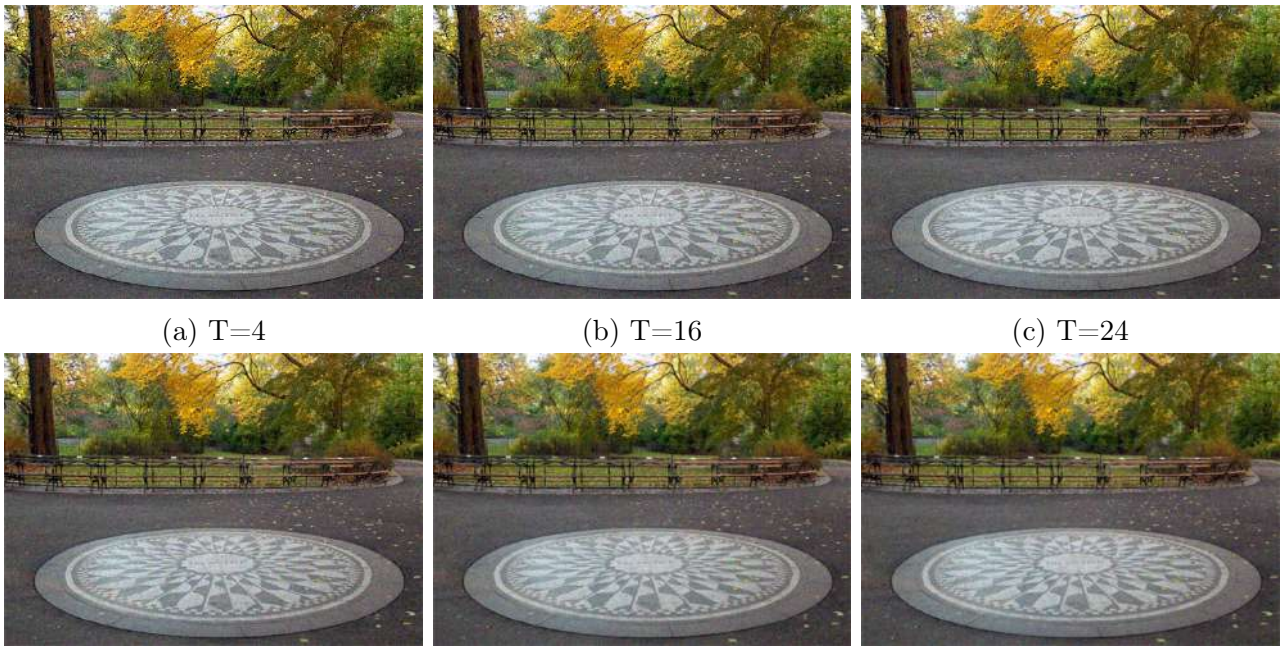
Figure 3.1: MSE and PSNR for Heat Equation



(a) λ=0.0125 (b) λ=0.0375 (c) λ=0.1000

(d) λ=0.1375 (e) λ=0.2000 (f) λ=0.2500

Figure 3.2: Heat Equation with 6 different λ



(a) T=4 (b) T=16 (c) T=24

(d) T=40 (e) T=56 (f) T=64

Figure 3.3: Heat Equation with 6 different time and fixed value λ=.00125

For this project, steady state heat equation is used. This means that the PDE runs until the stopping condition is satisfied. For small number ϵ, the following stopping condition is used:

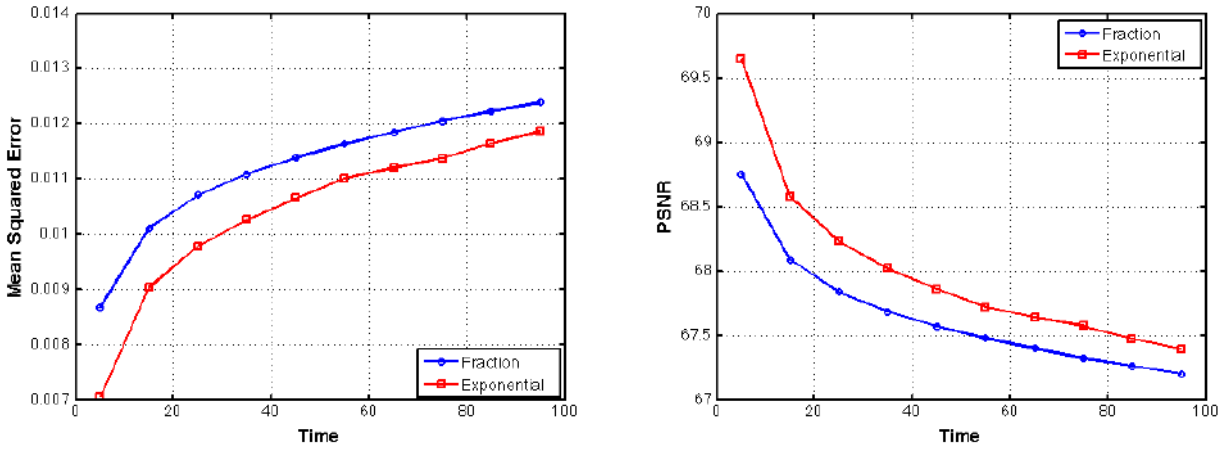
$$\epsilon < \min(\max(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}))$$

PSNR is upper bounded by approximately 75 and lower bounded by approximately 69. A Figure 3.2 (f) looks smoother than the other images of Figure 3.2. However, because the gap between

upper and lower bound of PSNR for heat equation is approximately 6, there's no remarkable image restoration between Figure 3.2 (a) and (f). Figure 3.3 shows image restoration under same λ but different time.

3.2 Denoising with the Perona-Malik model

Recall that the conduction coefficients of the Perona-Malik equation can be determined in two ways: exponential and fractional. In this section, only denoising will be focused on.



(a) MSE−Time (b) PSNR−Time

Figure 3.4: MSE and PSNR for Perona-Malik model

The MSE graph above strictly increases, while PSNR graph strictly decreases. In MSE−time graph, the first exponential step is almost zero value. MSE value sits between 0.0007 and 0.0013. Time starts from 5 to 95 with step size 10.

3.2.1 Denoising with Fractional Coefficient

There's a little difference between original noisy image and restored image which is executed during five time units. When time=15, it is distinguishable with noisy image (a). Figure (d) and (e) become more smoother than (b) and (c). Last, figure (f) results too smoother image so that some details that should be on the circle are wiped out.

(a) noisy image (b) time=5 (c) time=15

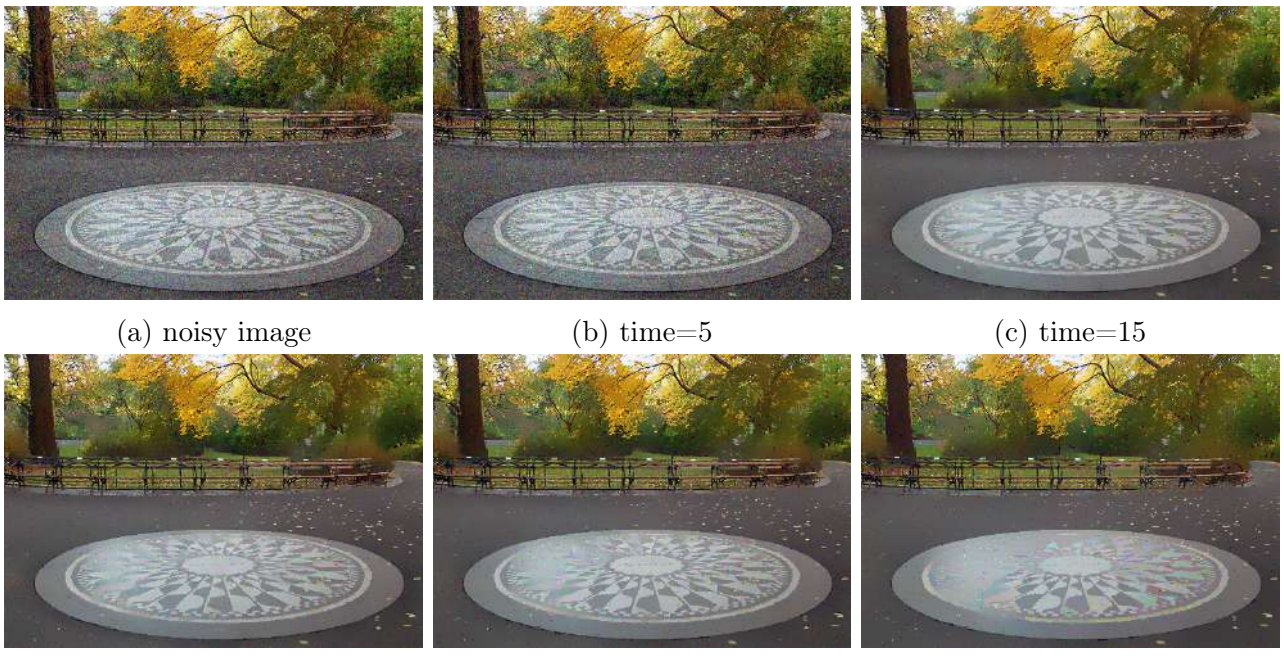
Figure 3.5: Images for Fractional Coefficients with time=5,15, and noisy image

(d) time=35 (e) time=55 (f) time=95

Figure 3.6: Images for Fractional Coefficients with time=35, 55, and 95

3.2.2 Denoising with Exponential Coefficient

Comparing (a) with (b) below, there's no remarkable difference between them. In MSE graph, it can be seen that the exponential line starts at almost zero. This is related with the fact that when two given images are identical, the MSE value will be zero. When time=15, the image is distinguished from noisy image. Figure 3.4 (b) shows that the PSNR values of exponential are larger in the entire time domain. From images (d), (e), and (f), it can be seen that as the time gets longer, smoothness is improved. The half of the details on the circle are wiped out when time=95.



(a) noisy image (b) time=5 (c) time=15

(d) time=35 (e) time=55 (f) time=95

Figure 3.7: Images for Exponential Coefficients with five different execution time

3.3 Total variation

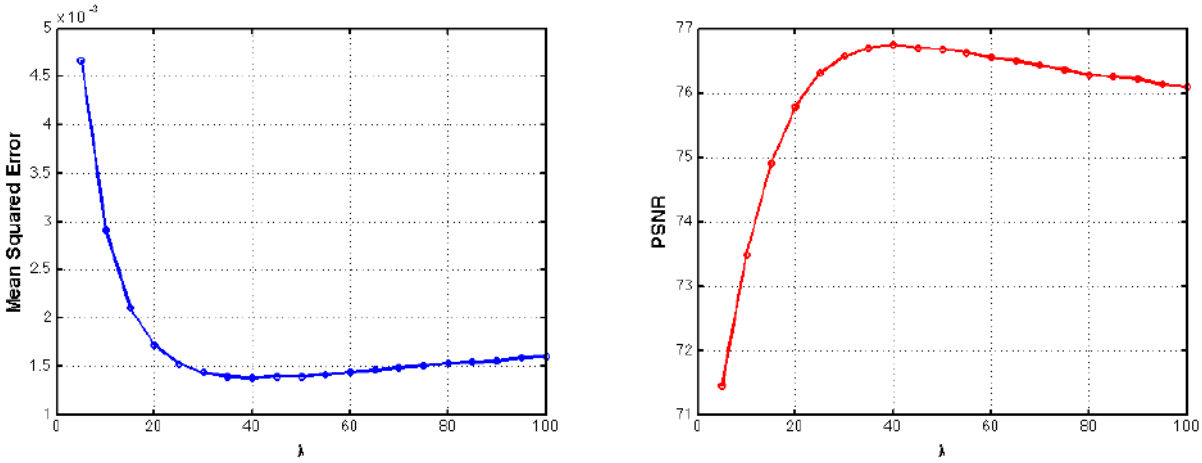
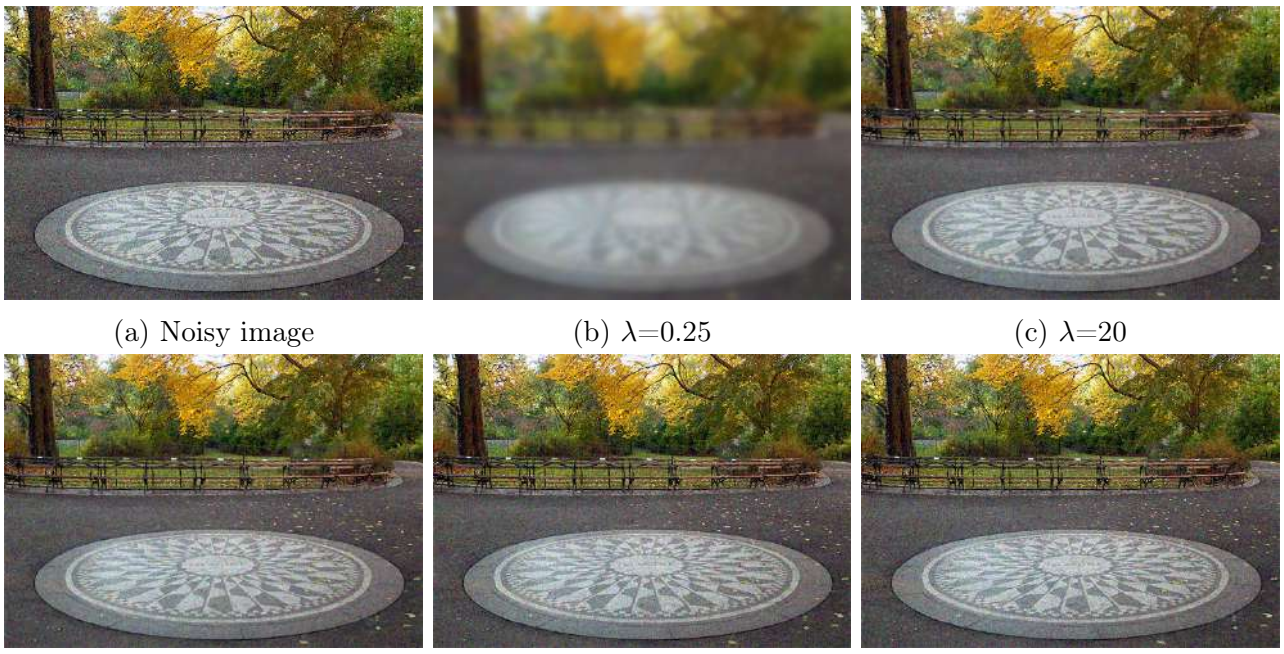


Figure 3.8: MSE and PSNR for the total variation

MSE hits the minimum at between λ=35 and 40. It implies that PSNR hits the maximum values in that interval. After reaching the maximum PSNR value, the graph starts to decrease and approaches to value of 76. This shows that once PSNR approaches the maximum value, the restored images are likely to become noisy (a). Figure 3.10 (d) is the restored image when PSNR has its maxima. λ less than 5 is also applied and is shown below in Figure 3.10 (b). When λ=0.25, the image is too smooth for objects in the picture to be distinguished.



(a) Noisy image (b) λ=0.25 (c) λ=20

(d) λ=40 (e) λ=80 (f) λ=100

Figure 3.9: Denoise depends on λ

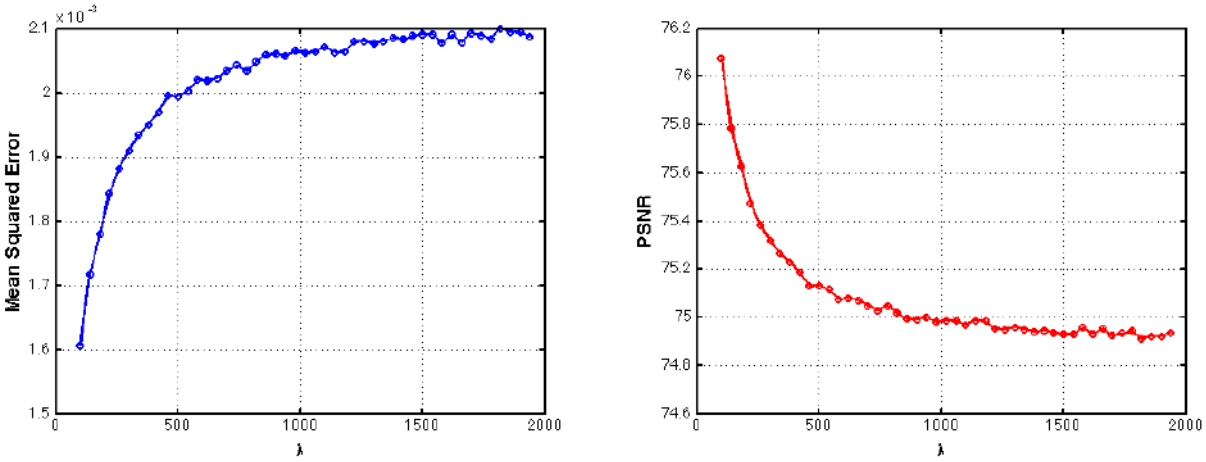


Figure 3.10: MSE and PSNR for the large λ

Figure 3.11 shows how PSNR behaves when λ becomes larger than 100. PSNR is decreasing but not monotonically decreasing and it finally converges to approximately 74.5.

Chapter 4. Conclusion

In this work, we studied how the machine recognizes digital image and the image can be processed. Because humans have a sensitive eye, not only performing image processing but also finding image quality are important. PSNR and MSE are introduced as measurements of image quality. PSNR is defined as follows:

$$PSNR = 20 \log \left(MAX_I\right) - 10 \log \left(MSE\right)$$

The higher PSNR implies better image quality. Hence, the PSNR is directly related with the quality of the image. In **Chapter 3**, we have shown the MSE and PSNR values for each algorithms.

	MSE	PSNR
Heat equation	71-77	0.001-0.045
Perona-Malik	67-70	0.007-0.012
Total Variation	69-75	0.002-0.006

Following two images give the figure number as well here (b) and (c) are restored by Perona-Malik anisotropic diffusion equation with time=20. These two are best results during the entire image processing work, because the noises are clearly removed and the image give the figure number as well here (b) has sharp edges. Among the restored images in **Chapter 3**, even PSNR and MSE show that the shown images are restored from noise. There are still a few yellowish or reddish dots, especially around the circle, which occur because the color conversion between RGB and YUV has not been applied.

(a) Noisy image (b) μ=0.0125 (c) μ=0.025

Figure 4.1: Two best results by Perona-Malik anisotropic diffusion equation

Among the restored images in **Chapter 3**, even PSNR and MSE tell that the shown images are restored from noise, there's still a few yellowish or reddish dots around the circle especially. It occurred because color conversion between RGB and YUV has not been applied.

References

[1] Pascal Getreuer. Rudin-Osher-Fatemi total variation denoising using split bregman. *Image Processing On Line*, 10, 2012.

[2] Mathworks. http://www.mathworks.com/help/images/ref/imfilter.html.

[3] Mathworks. http://www.mathworks.com/help/vision/ref/psnr.html.

[4] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[5] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.