COMPARATIVE STUDY OF IMAGE RESTORATION ALGORITHMS AND CONVERGENCE OF ITERATIVE RICHARDSON-LUCY

A THESIS

submitted by

FAIZAN ABBAS ALI

in partial fulfilment of the requirements
for the award of the degrees of

BACHELOR in ELECTRICAL ENGINEERING
&
MASTER OF TECHNOLOGY in COMMUNICATION SYSTEMS

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2015

Acknowledgement

I am grateful to the guidance provided by Dr. Aravind R., my Dual Degree project advisor. Doing a project under his guidance has helped me get exposed to varied topics and aspects of image restoration and their application in the field of image processing. His continued encouragement helped me grow in my support.

I thank Dr. A. N. Rajagopalan and other faculty members and PhD. students from the image processing group of Electrical Engineering IIT Madras for their help in carrying out my project successfully.

I also thank my family for their unquestioned support and trust in me.

And last but not the least I would I like to thank Leslie Lamport and D. Knuth for LaTeXand `http://tex.stackexchange.com/` for their unrestricted support.

Abstract

The purpose of this project is to investigate different image restoration algorithms and their applications and implications. The study is done on the techniques of restoration of shift-invariant blurred images which are corrupted by noise.

The first part of the project deals with various optimization methods used in image processing. The techniques evaluated are tested on different parameters and then used in the algorithms described further. This part also deals with Wiener filtering and an introductory mathematical formulation of Richardson-Lucy algorithm.

The thesis then deals with the aforementioned Richardson-Lucy algorithm. The algorithm was compared against other blind image restoration algorithm. The convergence of the Richardson-Lucy algorithm is dealt with details on gray-scale and colored images alike. The convergence of the algorithm is observed in noisy-blurred images and also in separate cases. Finally the thesis compares the performance of `MATLAB`'s implementation and the `C++` implementation of the algorithm.

Index terms— Image Processing, Richardson-Lucy Algorithm, `C++` implementation, data analysis

Table of Contents

List of Figures

Chapter 1

Introduction

In many imaging applications, the measure image is a degraded version of the *true* (or *original*) image that ideally represents the scene. The degradation may be due to

- Atmospheric distortions (including turbulence and aerosol scattering)
- Optical aberrations (such as diffraction and out-of-focus blur)
- Sensor blur (resulting from spatial averaging of photosites)
- Motion blur (resulting from camera shake or the movements of objects in the scene)
- Noise (such as shot noise and quantization)

Image restoration algorithms aim to recover the true image from degraded measurements. This inverse problem is typically ill-posed, meaning that the solution does not satisfy at least one of the following: *existence*, *uniqueness*, or *stability*. Regularization techniques are often adopted to obtain a solution with desired properties, indicating a knowledge of prior information.

Image restoration is widely used in almost all technical areas involving images; astronomy,remote sensing, microscopy, medical imaging, photography, surveillance, and HDTV systems to name a few. For example, license plate may appear illegible due to motion blur; photographs captured under low-light conditions may suffer from noise; out-of-focus photographs may look blurry; standard TV signals may

not be sufficiently sharp for high-definitions TV sets; archived movies may be corrupted by artifacts and noise; atmospheric distortions may degrade the quality of images in remote sensing. In these examples and in many more scenarios, the importance of image restoration ranges from beneficial to essential.

Deblurring is commonly referred to restoration of images degraded by blur. Although the degradation process is in general non-linear and spatial varying, a large number of problems could be addressed with a linear shift-invariant(LSI) model (Section **??**). Because the output of an LSI system is the convolution of the true image with the impulse response of the system, the point spread function(PSF), image restoration in LSI systems is called image *deconvolution*. When the impulse response of the system is a delta function and there is only noise, the restoration process becomes image *denoising*. If the PSF is unknown, then the problem is referred to as *blind* image restoration.[1]

The past three decades have brought significant progress in the development of efficient methods for classical deconvolution in both single and multi-image scenarios. Most of the earlier work was concentrated on the *Space Invariant* (SIV) blurring.[2]

The subsequent chapters deal in details the LSI model (Section 2.1) of image degradation. Then we will look into different Image restoration algorithm (Section 2.2) with inclination towards SIV blurred images and their restoration. Also the optimization algorithm (Algorithm 1 and Algorithm 2)used will be discussed. We will then shift our focus towards Bayesian Image Restoration (Section 2.3) and analyze the methods when we have *a priori* information.

The main focus of this thesis is Richardson-Lucy algorithm which we discuss in much detail in Chapter 3. We will look through the `MATLAB` implementation of the algorithm, understand the working of it (Section 3.2) and then convert the code into a working `C++` code. We compare the results from `MATLAB` implementation and the `C++` implementation.

Chapter 2

Image Restoration

2.1 Linear Shift-Invariant Degradation Model

Suppose that *f(x,y)* is the true image that we would like to recover from the degraded measurement *g(x,y)*, where *(x,y)* are the spatial coordinates. For a linear shift-invariant system, the imaging process can be formulated as

$$g(x, y) = h(x, y) * f(x, y) + n(x, y), \tag{2.1}$$

where "*" is the convolution operation, *h(x,y)* is the PSF of the imaging system, and *n(x,y)* is the additive noise. The imaging formulation can also be done in matrix-vector form or in frequency domain. Defining **g,f** and **n** as the vectorized versions of *g(x,y)*,*f(x,y)* and *n(x,y)*,respectively,the matrix-vector formulation is

$$\mathbf{g = Hf + n,} \tag{2.2}$$

where **H** is a two-dimensional sparse matrix with elements taken from *h(x,y)* to have the proper mapping from **f** to **g**. The vectorization could be done in a number of ways, including raster-scan, row-major, and column-major order.

On the other hand, the Fourier-domain version of the imaging model is

$$G(u, v) = H(u, v)F(u, v) + N(u, v) \tag{2.3}$$

where *G(u,v), H(u,v), F(u,v),* and *N(u,v)* are the Fourier transforms of *g(x,y), h(x,y), f(x,y),* and *n(x,y)*, respectively.

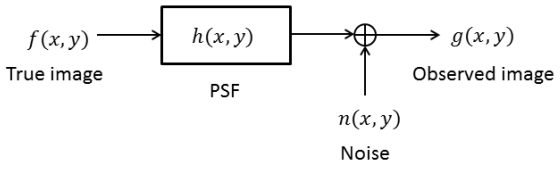

Figure 2.1: Linear shift-invariant image degradation model

Since recorded images are of finite spatial extent, the support of the PSF goes beyond the borders of the degraded image; this boundary value problem may create artifacts affecting the entire restored image[3]. The simplest way of extending an image is zero padding, assuming all pixels outside measured region have zero values. With this extension, the blur matrix **H** has a Block Toeplitz with Toeplitz Block (BTTB) or simply Doubly Block Structure.[4] Another common method is periodic extension,in which case the blur matrix **H** is Block Circulant with Circulant Block (BCCB).Because a BCCB matrix can be diagonalized by discrete Fourier transform, periodic extension is the implicit method in Fourier domain solutions.

The PSF *h(x,y)* is the impulse response of the imaging system; it is the image of a point from the scene and essentially tells the smallest image detail an imaging system can form. It's Fourier transform *H(u,v)* is the frequency response of the imaging system and is called the optical transfer function(OTF). In general, frequency domain analysis of an imaging system provides additional insight about the behavior of the system. Also, when there are multiple subsystems, we can multiply the individual transfer functions to get the overall transfer function. This method is easier than the repeated convolution of the impulse responses and allows for easy visualization and understanding of the performance of the overall system.

The major factors contributing to the image degradation are as follows.

- *Diffraction:* Because of the wave nature of the light, an optical system can form a point image. Even when there is no other degradation (such a system is called a diffraction-limited system), an optical system will have a characteristic minimum blur,which is determined by the aperture of the system. For a circular aperture of diameter D, the diffraction PSF has a specific shape known as the Airy Disk. The well-known Rayleigh resolution limit to discern two discrete points, $1.22\frac{\lambda f}{D}$, is approximately the distance between the maximum and the first minimum of the Airy disk.

- *Atmospheric Blur:* Atmospheric distortion are caused mainly by turbulence and aerosol scattering/absorption. A turbulent medium causes wavefront tilt, resulting in local or global image shifts.

The MTF of the atmospheric turbulence typically has an exponential decay, with dependence on various parameters, including medium characteristics, path length,and exposure time. Aerosol effects, on the other hand, cause diffusion and are modeled well with a Gaussian MTF. It is possible to estimate these parameters and restore images.

- *Out-of-focus Blur:* The out-of-focus PSF takes the shape of the camera aperture. For a circular aperture, the PSF is a disk, which is sometimes referred to as the circle of confusion. For a thin-lens model, it can be shown that the diameter of the disk is $D(|d' - d|/d')m$,where D is the diameter of the disk, f is the focal length, d is the distance between the lens and the in-focus plane, d' is the distance between the lens and the out-of-focus plane, and m is the lens magnification.

- *Motion Blur:* Motion blur occurs when the scene is not static and the exposure time is not small enough with respect to the motion in the scene or the camera. In such a case the projected imagery is smeared over the sensor according to the motion.

- *Sensor Blur:* A sensor integrates light over a photosensitive are, which is typically a rectangular area forming the PSF of the sensor. If there is a micro-lens array in front of the sensor, then the sensor PSF takes the shape of the micro-lens.

- *Anti-Aliasing Filter:* Aliasing happens if the spatial sampling rate is less than the Nyquist rate. To prevent aliasing, an anti-Aliasing filter should be applied before the image plane. The anti-aliasing filter should be designed accordng to the photosite pitch and the color filter array.

- *Optical Abberation:* An optical system may introduce additional aberrations, including spherical aberration,chromatic aberration and geometric distortion. blur caused by optical aberrations could be space varying and color channel dependent.

- *Noise:* In addition to the blurring effects, the image could also be corrupted by noise. There are different sources of noise, including dark current, shot noise, read noise and quantization.

2.2 Image Restoration Methods

2.2.1 Least Square Estimation

The inverse problem of estimating $\mathbf{f}$ from the observation $\mathbf{g} = \mathbf{Hf}$ is typically not well-posed because of non-existence, non-uniqueness, or instability of solution. The least squares estimation minimizes the sum of squared differences between the real observation $g(x, y)$ and the predicted observation $h(x, y) * f(x, y)$. The cost function to be minimized can be written as $\sum_{(x,y)} |g(x, y) - h(x, y) * f(x, y)|^2$ in spatial domain, $\sum_{(x,y)} |G(u, v) - H(u, v)F(u, v)|^2$ in Fourier domain, and $||g - Hf||^2$ in matrix-vector notation.[5] The solution in matrix-vector notation is

$$\hat{\mathbf{f}} = \arg\min_{f} ||\mathbf{g} - \mathbf{Hf}||^2 = \mathbf{H}^{+}\mathbf{g}, \tag{2.4}$$

where $\mathbf{H}^{+}$ is known as the pseudo-inverse of $\mathbf{H}$. For an over-determined full-rank system, the pseudo-inverse is $\mathbf{H}^{+}$=$(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, which can be derived by setting the derivative of the cost function to zero: $\frac{\partial}{\partial \mathbf{f}}||\mathbf{g} - \mathbf{Hf}||^2 = 2\mathbf{H}^T\mathbf{Hf} - 2\mathbf{H}^T\mathbf{g} = 0$.

2.2.2 Steepest Descent Algorithm

It is one of the most widely used method for convex optimization and convergence.The prime factor for it's popularity being that it is un-dissembling and very easy to implement. The steepest descent method updates an initial estimate iteratively in the reverse direction of the gradient of the cost function $C(\mathbf{f})$. An iteration of the steepest descent method is

$$\mathbf{f}^{(i+1)} = \mathbf{f}^{(i)} - \alpha \frac{\partial}{\partial \mathbf{f}} C(\mathbf{f}) \tag{2.5}$$

where:

α is the step size

$\mathbf{f}^{(i)}$ is the i^{th} estimate

In our case where cost function is defined as $C(\mathbf{f}) = (1/2)||\mathbf{g} - \mathbf{Hf}||^2$, and also changing matrix-vector notation to indexing notation the iteration step is

$$f^{(i+1)}(x, y) = f^{(i)}(x, y) + \alpha h^T(x, y) * (g(x, y) - h(x, y) * f^{(i)}(x, y)) \tag{2.6}$$

The iteration are repeated until the stopping criterion, which can be either number of maximum iteration possible, or the rate of change in the estimated signal $||\mathbf{f}^{(i+1)} - \mathbf{f}^{(i)}||/||\mathbf{f}^{(i)}||$ or in the cost function. The step size i.e. α should be small enough to ensure converge and large enough so that the rate of convergence is fast.

Algorithm 1 Steepest Descent Algorithm

$\delta \leftarrow$ value near minima of $f(x)$

fix a random start point x_0

choose α

if $f(x) \geqslant \delta$ **then**

calculate $\nabla f(x)|_{x_0}$

search in $-\nabla f(x)$

update $x_{i+1} = x_i + \alpha * d_i$ // d_i is the direction of fall of gradient

end if

2.2.3 Conjugate Gradient Algorithm

This method is a much more efficient than the steepest descent algorithm, and it assumes that the gradient of the cost function is calculable at every step, and that this information can be used to improve the search for the global minimum.This method uses the conjugate gradients for traversing downhill, in place of the gradient of the cost function. As a result, the solution is quick and reached in comparatively fewer iterations.[5]

Algorithm 2 Conjugate Gradient Algorithm

$\delta \leftarrow$ value near minima of $f(x)$

fix a random start point x_0

choose α

if $f(x) \geqslant \delta$ **then**

calculate $\nabla f(x)|_{x_0}$

search in $-\nabla f(x)$

update α such that $f(x + \alpha_{(i)} * d_i) < f(x + \alpha * d_i)$ // d_i is the direction of fall of gradient

update $x_{i+1} = x_i + \alpha * d_i$

end if

2.2.4 Wiener Filtering

The Wiener estimator looks for an estimate in the form $\hat{\mathbf{f}} = \mathbf{W}\mathbf{g} + \mathbf{b}$ that minimizes the mean square error between the true image and the estimated image, $E\{(\mathbf{f}-\hat{\mathbf{f}})^T(\mathbf{f}-\hat{\mathbf{f}})\}$ where $E\{.\}$ is the expectation operator. The optimal $\mathbf{W}$ and $\mathbf{b}$ values can be obtained using the orthogonality principle[6], which specifies two conditions : (1) the expected value of the estimate must be equal to the estimated value of the true image, that is, $E\{\hat{\mathbf{f}}\} = E\{\mathbf{f}\}$; and (2) the restoration error must be orthogonal to the observation about it's mean, that is $E\{(\mathbf{f}-\hat{\mathbf{f}})(\mathbf{g}-E\{\mathbf{g}\})^{\mathrm{T}}\} = 0$.. From the first condition, the bias term can be written as

$$b = E\{\mathbf{f}\} - \mathbf{W}E\{\mathbf{g}\}, \tag{2.7}$$

and by substituting the bias term and $\hat{\mathbf{f}} = \mathbf{W}\mathbf{g} + \mathbf{b}$ into the second condition, the restoration matrix is found to be

$$\mathbf{W} = \mathbf{Q}_{\mathbf{fg}}\mathbf{Q}_{\mathbf{g}}^{-1}, \tag{2.8}$$

where $\mathbf{Q}_{\mathbf{fg}} = E\{(\mathbf{f}-E\{\mathbf{f}\})(\mathbf{g}-E\{\mathbf{g}\})^T\}$ is the cross-covariance matrix between the true image and the observation, and $\mathbf{Q}_{\mathbf{g}} = E\{(\mathbf{g}-E\{\mathbf{g}\})(\mathbf{g}-E\{\mathbf{g}\})^T\}$ is the covariance matrix of the observation. Simplifying further for the case of **LSI** system and using the assumption that true image and noise are uncorrelated, the Wiener filter(3.8) is

$$\mathbf{W} = \mathbf{Q}_{\mathbf{f}}\mathbf{H}^T(\mathbf{H}\mathbf{Q}_{\mathbf{f}}\mathbf{H}^T + \mathbf{Q}_{\mathbf{n}})^{-1}, \tag{2.9}$$

where $\mathbf{Q}_{\mathbf{f}}$ and $\mathbf{Q}_{\mathbf{n}}$ are the covariance matrices of the true image and noise. The bias term is zero only when expected values of true and observed image are zero. But we can avoid the bias term by mean normalizing the observed and estimated images before applying the filter (3.8), and finally adding the true image mean to obtain the solution.

The Wiener filter in Fourier domain provide further insight to the solution. It is given as,

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_n(u,v)}{S_f(u,v)}}, \tag{2.10}$$

where $S_n(u,v)$ and $S_f(u,v)$ are power spectral densities of noise and true image respectively. Thus, estimated image is given as $\hat{F}(u,v) = W(u,v)G(u,v)$. In absence of any blur i.e. $H = 1$, the filter takes form

$$W(u,v) = \frac{S_f(u,v)}{S_n(u,v) + S_f(u,v)} \tag{2.11}$$

which at low frequencies, take form $W(u,v) = 1$ i.e. allow all the information to pass, and at high frequencies becomes ***SNR***. In absence of noise the filter goes to the other extreme and behave as an inverse filter $W(u,v) = \frac{1}{H(u,v)}$.

Algorithm 3 Wiener Filtering (**G**,**f**)

for $k \rightarrow 0$ **to** 2 **do**

calculate $\hat{\mathbf{H}} = \frac{1}{1+k}$

calculate estimate $\hat{\mathbf{F}} = \mathbf{HG}$

find $\hat{\mathbf{F}}$ such that $\sqrt{|f - \hat{f}|^2}$ is minimum

output $\hat{\mathbf{F}}$

end for

Experiments and Results

The original images in Figure 2.2(left), Figure 2.4(left), and Figure2.6(left) were degraded by white Gaussian blur and white Gaussian noise with $\sigma_{b1} = 1.2$ and $\sigma_{n1} = 15$, $\sigma_{b2} = 1.0$ and $\sigma_{n2} = 10$, and $\sigma_{b3} = 0.5$ and $\sigma_{n3} = 15$ respectively. The resulting images in Figure2.2(right), Figure2.4(right), and Figure2.6(right) thus obtained were passed through Wiener filter implemented in `MATLAB` using algorithm 3. The results Figure2.3(left), Figure2.5(left), and Figure2.7(left) were compared to the results of in-built `MATLAB`$'s$ function *wiener2* Figure2.3(right), Figure2.5(right), and Figure2.7(right) respectively.

Figure 2.2: Gray-scale Lena image used for the experiment

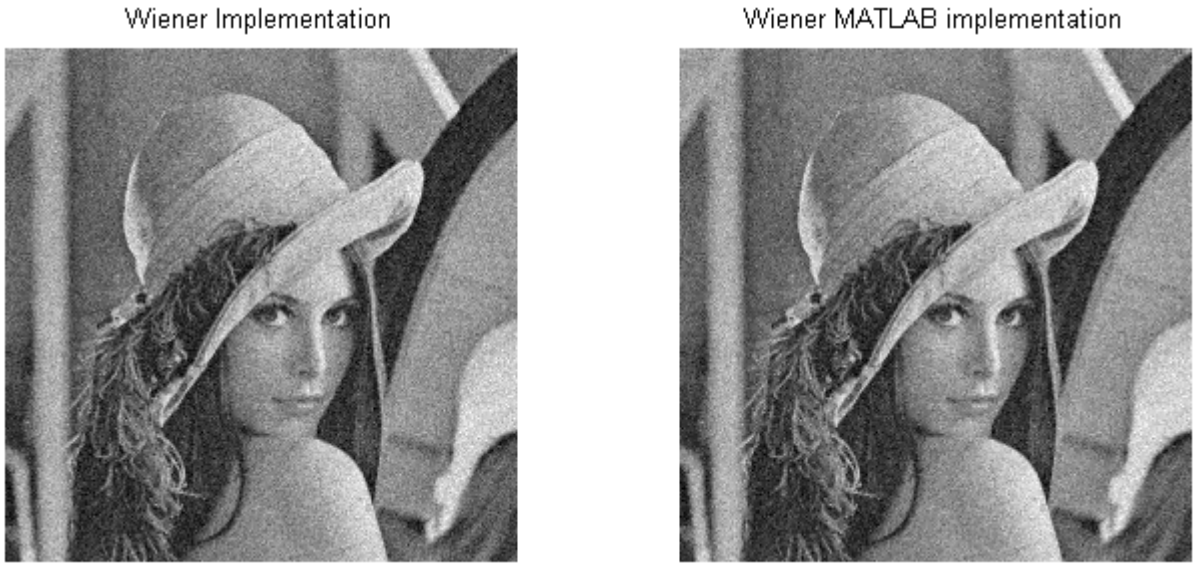

Figure 2.3: Result of my implementation of Wiener filter and `MATLAB` in-built option

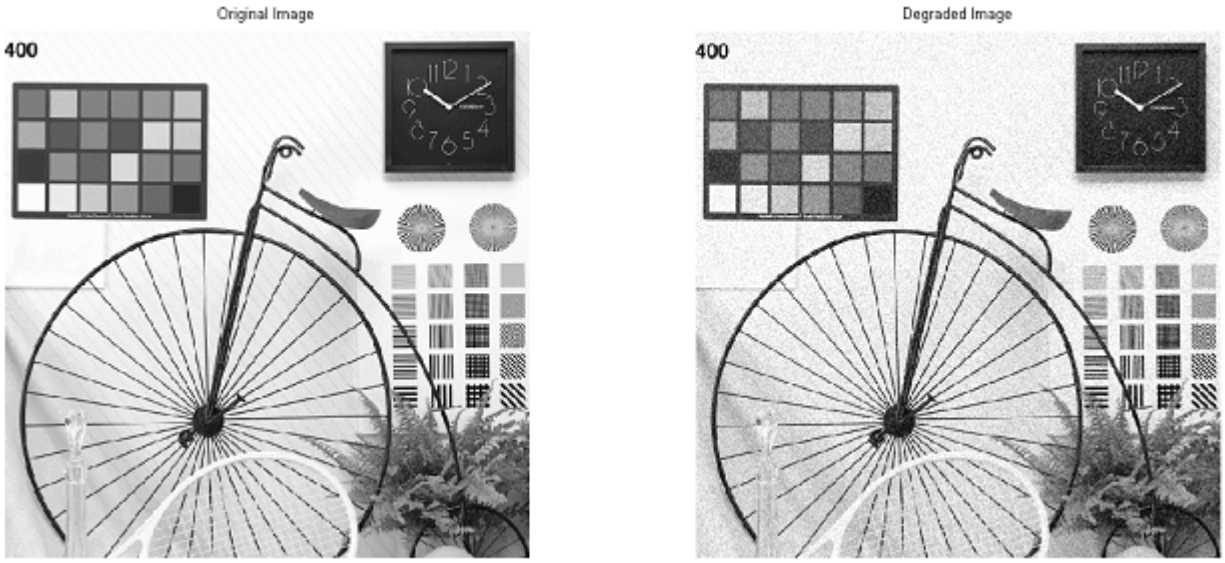

Figure 2.4: Grayscale image used for the experiment

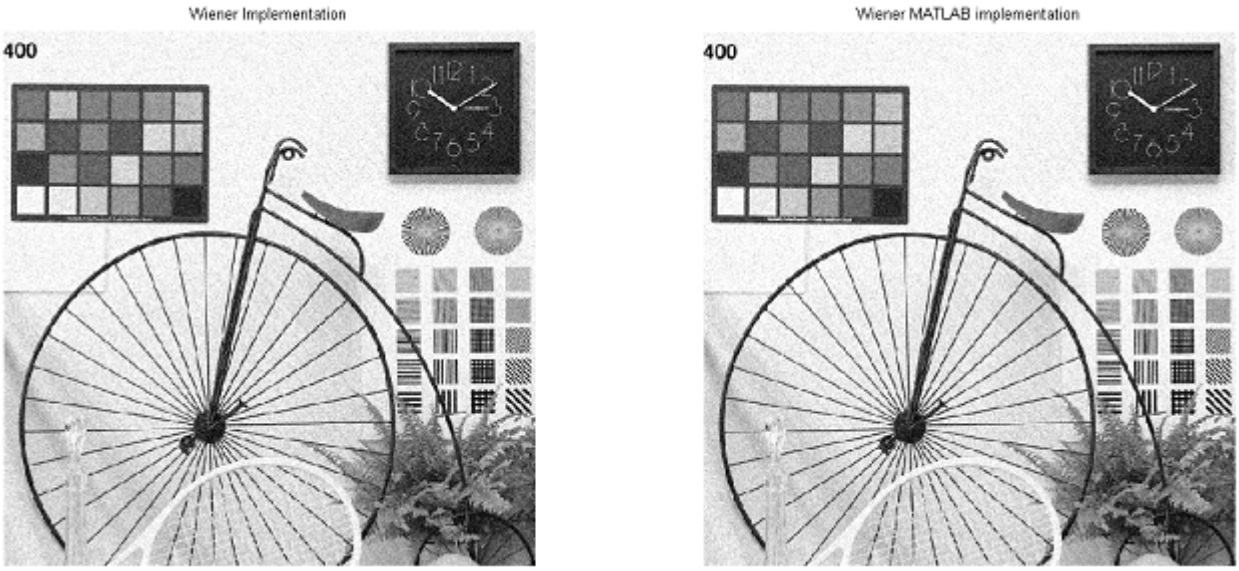

Figure 2.5: Result of my implementation of Wiener filter and `MATLAB` in-built option

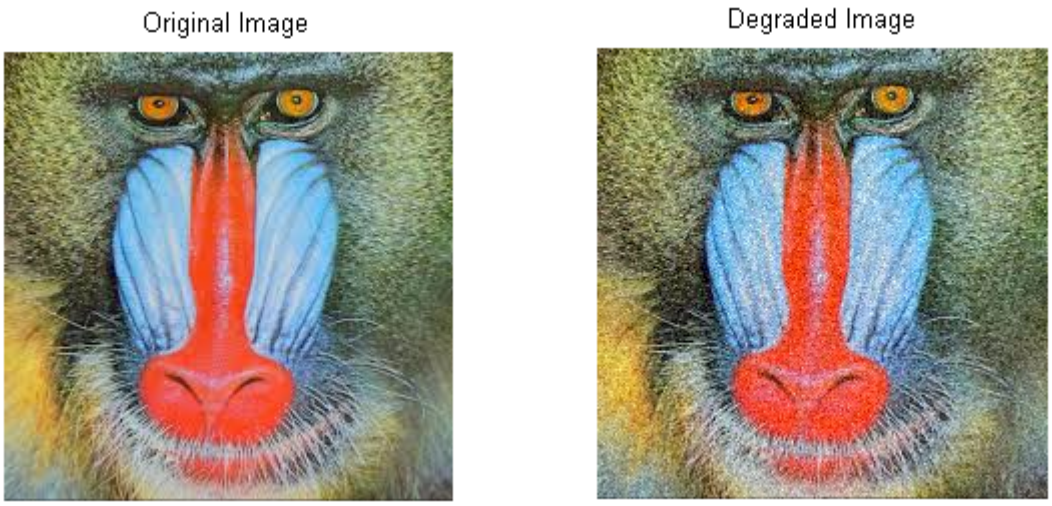

Figure 2.6: RGB mandarin image used for the experiment

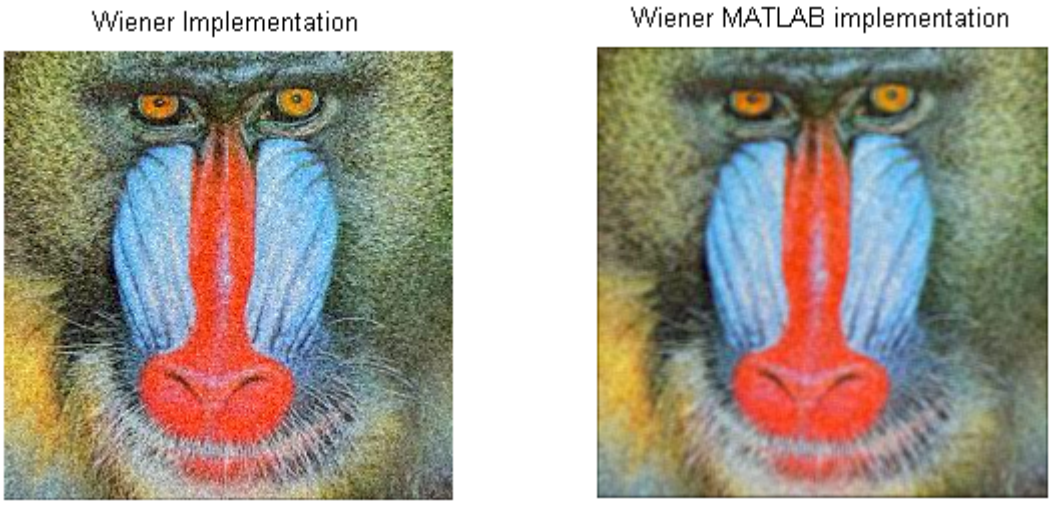

Figure 2.7: Result of my implementation of Wiener filter and `MATLAB` in-built option

2.3 Bayesian Image Restoration

Bayess law states that the posterior probability is proportional to the product of the likelihood and the prior probability i.e. $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, where for proposition A and evidence B
$P(A)$ is the *prior*,
$P(A|B)$ is the *posteriori*,
$P(B|A)P(B)$ is the support of A,
The likelihood encompasses the information contained in the new data. The prior expresses the degree of certainty concerning the situation before the data are taken. Bayesian estimation provides an elegant statistical perspective to the image restoration problem. The unknown image, noise, and PSF(in case of blind deconvolution) are all treated as random variables.

In most image restoration problems, image noise is modeled to be zero-mean independent identically distributed (iid) Gaussian random variable: $p(n(x,y)) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{1}{2\sigma_n^2}(n(x,y))^2\right)$. Thus conditional probability of the observed image is

$$\begin{aligned} p(\mathbf{g}|\mathbf{f}) &= \prod_{x,y} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{1}{2\sigma_n^2}(g(x,y) - h(x,y) * f(x,y))^2\right) \\ &= \frac{1}{\left(\sqrt{2\pi}\sigma_n\right)^M} \exp\left(-\frac{1}{2\sigma_n^2}||\mathbf{g} - \mathbf{H}\mathbf{f}||_2^2\right), \end{aligned} \tag{2.12}$$

where σ_n is noise standard deviation and M is the total number of pixels in the observed image. But in many cases where the noise is modeled as Poisson random process then the conditional probability of the observed image is

$$p(\mathbf{g}|\mathbf{f}) = \prod_{x,y} \frac{(h(x,y) * f(x,y))^{g(x,y)} \exp\left(-(h(x,y) * f(x,y))\right)}{g(x,y)!} \tag{2.13}$$

The Equation 2.13 when using ML estimator approaches the well-known iterative Richardson-Lucy algorithm[7][8],

$$f^{(i+1)}(x,y) = \left[\frac{g(x,y)}{h(x,y) * f^{(i)}(x,y)} * h^T(x,y)\right] * f^{(i)}(x,y) \tag{2.14}$$

Richardson-Lucy or RL algorithm can be used both for *non-blind*,in which we assume the blurring operator to be known, and *blind*,in which we assume the blurring operator is unknown, deconvolution.

The problem in *non-blind* restoration is the PSF of the image is rarely known with certainty and if otherwise it's not accurate. This gives rise to PSF mismatch, which in-turn leads to poor deblurring results. The presence of noise further intensifies the problem.[17]

Chapter 3

Richardson-Lucy Algorithm

In the section 2.3, R-L algorithm was introduced briefly. We will now discuss the same in details. The iterative R-L algorithm is based on Bayesian restoration and uses the fact that we have *apriori* knowledge of the PSF kernel. Also we will assume that the size of the estimated image, degraded image and the original image is same, which makes the iterations simpler.

3.1 Pseudo-code

Algorithm 4 Iterative R-L Algorithm

degraded image g and kernel k

initial estimate $f_0 = g * k$

for iteration 1 **to** 10 **do**

Subtract from blurred image $b = f - f_i * k$

Add the error corrected image $a = b + f_i$

Set image for next iteration $f_i = a$

end for

3.2 MATLAB code explanation

Given an image I, which is *blurred* and *noisy*, an initial estimate of kernel h.Find the best estimate of the *de-convolved* image J.[9, 10]

3.2.1 Definitions

DECONVLUCY J = *deconvlucy*(I, PSF) deconvolve image I using $Lucy - Richardson$ algorithm, returning *deblurred* image J. The assumption is that the image I was created by *convolving* a true image with a point-spread function PSF h and possibly by adding *noise*.

- NUMIT *(optional)* number of iterations the algorithm runs through. Default value is 10.
- READOUT *(optional)* Additive *noise*.Default value is 0.
- SUBSMPL *(optional)* is used when the PSF finer than the image. Default is 1.

3.2.2 Code Interpretation

Preparing PSF

`sizeOTF(numNSdim) = SUBSMPL *sizeI(numNSdim);` // If sampling rate of PSF is not same as that of image. `H = psf2otf(PSF,sizeOTF);` // Changing from time domain to frequency domain. *i.e.* H = $DFT(PSF)$

Preparing Parameter for Iterations

`wI = max(WEIGHT.*(READOUT + J1),0);` // Generate wI matrix such that the positivity constraints are satisfied. i.e. $wI = 0$ or $wI = noisy\ pixel\ value > 0$

`scale = real(ifftn(conj(H).*fftn(WEIGHT(idx:)))) + sqrt(eps);` // Scaling factor based on the weight matrix and how the flat-field of the image is defined.

3.2.3 L-R Interations

`for k = 1:NUMIT`// Loop over the number of iterations

Image predictions for the next iteration `Y = max(J2 + lambda*(J2 − J3),0);`// difference between the last two iterations of the algorithm. Basically this generates the error correction matrix. Also enforces positivity constraints.[10]

Core for the L_R interations `CC = corelucy(Y,,H,DAMPAR22,wI,READOUT,SUBSMPL,idx,vec,num);`//calling the in-built function which performs the basic **Lucy** algorithm.

`J3 = J2;` $//previous = next$

`J2 = max(Y.*real(ifftn(conj(H).*CC))./scale,0);` $//J\{2\} = y \otimes \left(h^T \otimes CC\right)$ in time domain.

`J4 = [J2(:)−Y(:) J4(:,1)];` // Implementation related iterations. This is related to optimization technique of `MATLAB` internally.

Corelucy

Resampling and Reshaping `ReBlurred = real(ifftn(H.*fftn(Y)));` // Reblurring the image get an initial estimate *i.e.* $y \otimes h$.

```
if SUBSMPL ~= 1,
ReBlurred = reshape(ReBlurred,vec);
for k = num,
vec(k) = [];
ReBlurred = reshape(mean(ReBlurred,k),vec);
end
end;
```

Explanation: This code snippet reshapes **ReBlurred** matrix on number of $non-singleton\ dimensions$

Estimation for next Iteration

`ReBlurred = ReBlurred + READOUT;` // Adding *noise* to image if present.

`ReBlurred(ReBlurred == 0) = eps;` // Positivity constraint.

`AnEstim = wI./ReBlurred + eps;` // solving the equation $Ax = B$ for each element in the **ReBlurred** to get an estimate of the output image. *eps* is used to maintain the positivity constraint.

```
if DAMPAR22 == 0,% No Damping
ImRatio = AnEstim(idx{:});
else % Damping of the image relative to DAMPAR22 = (N*sigma)^2
gm = 10;
g = (wI.*log(AnEstim)+ ReBlurred - wI)./DAMPAR22;
g = min(g,1);
G = (g.^(gm-1)).*(gm-(gm-1)*g);
ImRatio = 1 + G(idx{:}).*(AnEstim(idx{:}) - 1);
end;
```

`f = fftn(ImRatio);` // return the image generated by the function

3.3 Convergence of RL Algorithm

R-L algorithm starts usually from an initial model of constant density distribution that apparently has maximum entropy as well as being perfectly smooth, then it modifies the estimates step by step by collecting information from the observational data until a reasonable fit is reached. This approach is opposite to that of any regularization methods, which searches for best fitted models subject to constraints like smoothness or flatness. Since the *likelihood* of model fitting is improved after each step while an infinite number of iterations is neither possible nor useful, we need to figure out when to stop the iterations. This become increasingly more important especially when *a priori* knowledge about the signal is not available or the signal-to-noise ratio is poor.[14]

3.3.1 Experimental Observations

The algorithm was tested under different conditions to check the convergence and find an optimal number of iterations. The test images in each of the experiments are gray-scale.

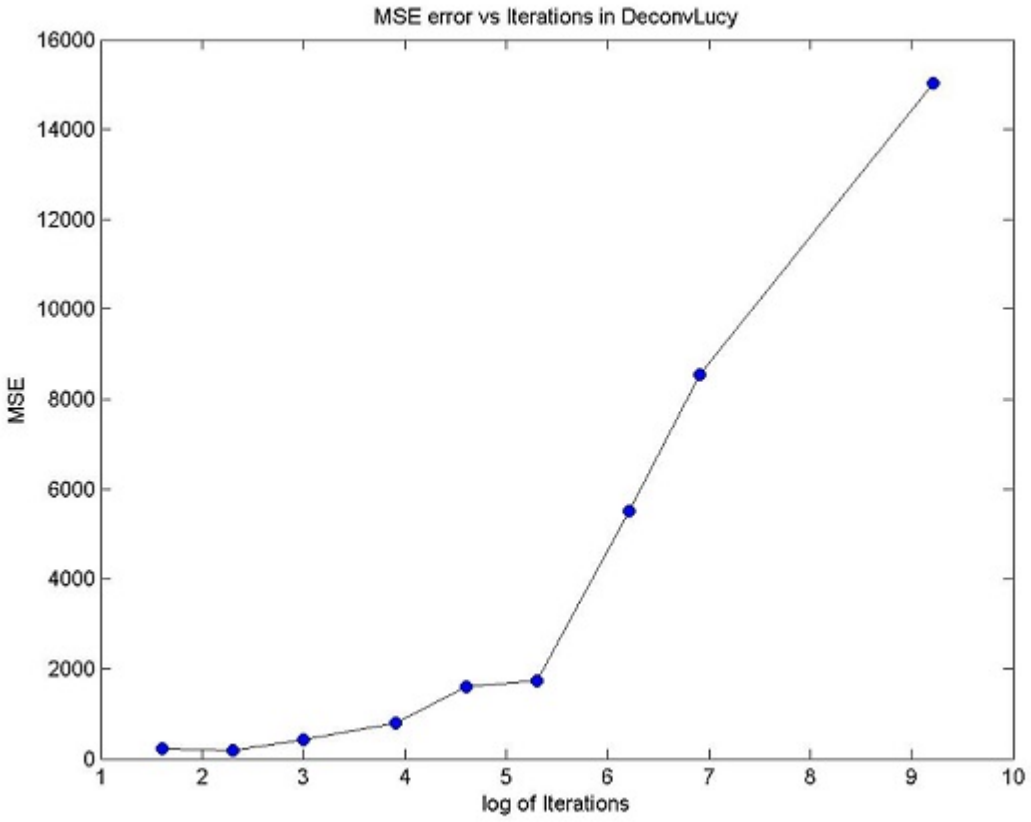

Figure 3.1: MSE error of an estimated image and original image vs log of iteration of R-L algorithm performed

We started off by performing large number of iterations for the R-L algorithm on images that have been degraded by Gaussian blur kernel and white Gaussian noise. The algorithm was thought to be asymptotically convergent, but the results were not in agreement with the theory. The behavior of my implementation of the algorithm and `MATLAB`$'s$ were identical. From Figure 3.1 it is evident the algorithm diverges as the number of iterations increases. A decrease in MSE was observed between `5` to `100` iterations region. Most of the experiments were then performed only for the aforementioned iterations region only.

In the subsequent experiments we tried to figure out the cause for the divergence of the MATLAB's implementation of the algorithm. SIV blurring with white Gaussian noise was applied to gray-scale images and RL algorithm iterations were performed.

In Graph3.2 and Graph3.3, the square root of MSE was recorded and compared between my implementation of the algorithm and that of the MATLAB. It is can noted that till 10 iterations the error is reasonable. Also it can be seen that my implementation closely follows the MATLAB's implementation.

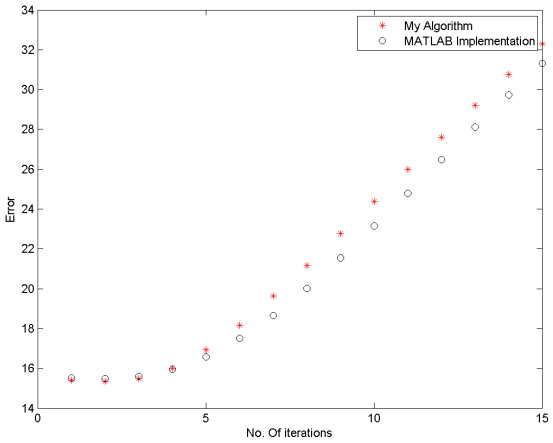

Figure 3.2: Comparison of performance of my implementation and MATLAB's implementation for 15 iterations

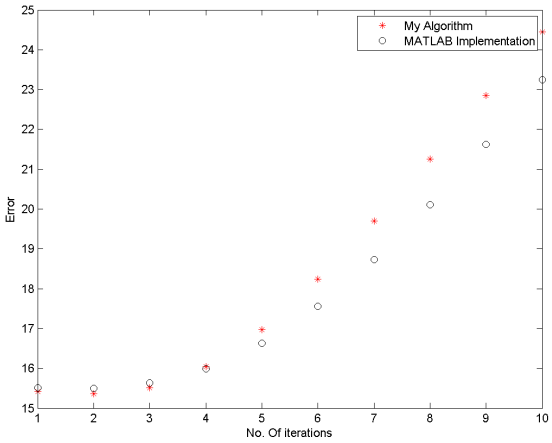

Figure 3.3: Comparison of performance of my implementation and MATLAB's implementation for 10 iterations

Then the experiments were performed to find out the convergence of *blind* and *non-blind* RL algorithm (MATLAB) respectively. The initially the number of iteration were kept a little a high to find out that when does the algorithm starts to diverge.

The experiment was performed on gray-scale images which were only *blurred* by Gaussian blur kernel. No noise was added to the test images. The results showed that the algorithm first starts to converge and then diverge. Graph3.4 shows the results as recorded.

When only noise (WGN) was applied to the test images the results were entirely different. The MSE error just kept on increasing with each iteration. Graph3.5 shows the recorded results.

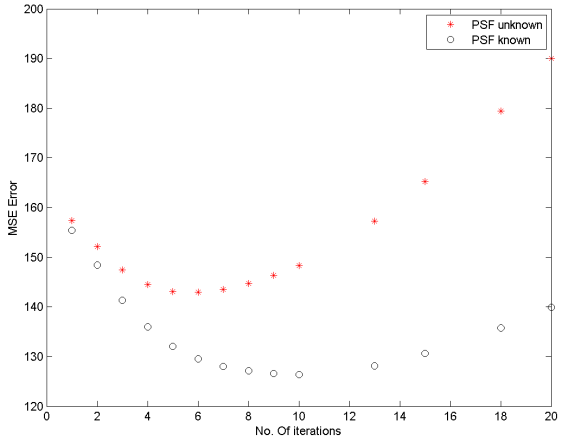

Figure 3.4: Convergence of *deconvlucy blind* and *nonblind* for only blurred image

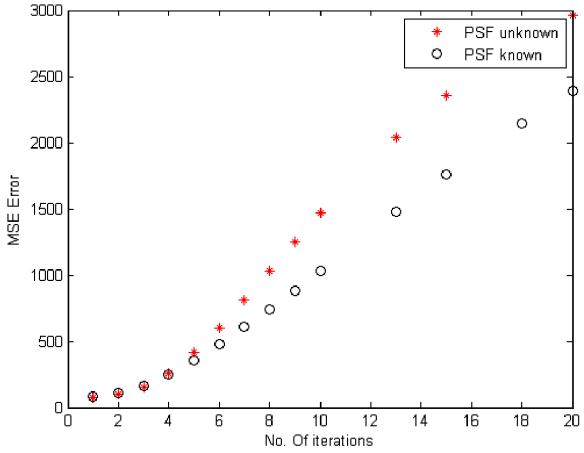

Figure 3.5: Convergence of *deconvlucy blind* and *nonblind* for only noisy image

The experiment was then repeated with going on till only `10` iterations. From Graph3.6 we can infer that it is reasonable to run the algorithm for upto `10` iterations to gain significant results when the image is only SIV blurred.

Whereas on the other hand from Graph3.7, we can infer that if the image only affected by `WGN`, the error keeps on increasing even for small number of iterations.

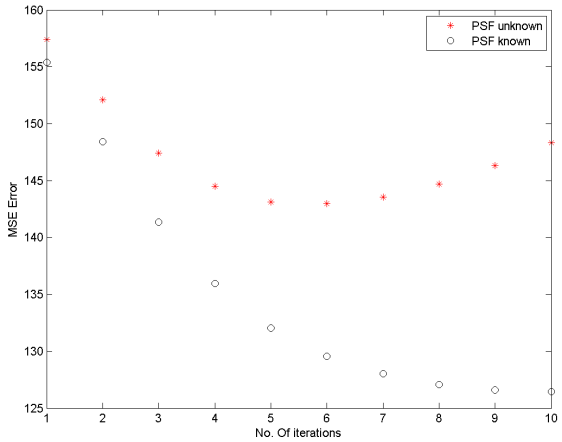

Figure 3.6: Convergence of *deconvlucy blind* and *nonblind* for only blurred image

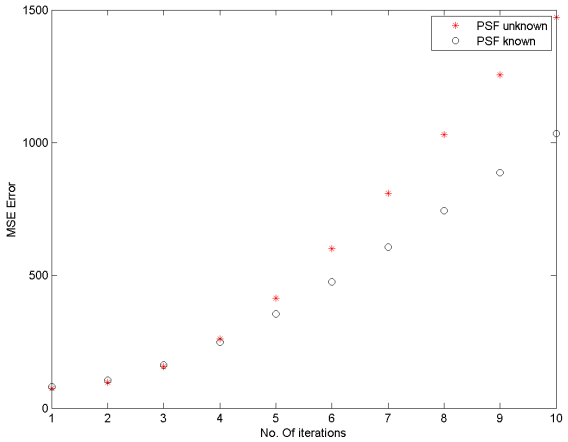

Figure 3.7: Convergence of *deconvlucy blind* and *nonblind* for only noisy image

The experiments were then repeated to benchmark my implementation of the algorithm against the MATLAB's implementation. The images were only subjected to SIV Gaussian blur and MSE was recorded for the 20 and 10 iterations respectively. Graph3.8 and Graph3.9 show the recorded results respectively.

From the result it is evident that my implementation closely follow the MATLAB's implementation. Also the deviation is with the range of error. We can thus safely assume to run the algorithm for 10 iterations. The RL algorithm performs better on images which are not corrupted by noise and are only blurred.

The experiments performed gave an in-sight to the working of the RL algorithm. They showed the performance of algorithm in different conditions and iterations. The main objective of the experiments was to study the convergence of the RL algorithm. Since the Algorithm4 is a difference algorithm the reason for divergence can be the addition of the MSE error to the estimated image over and over again after the best match is found. Thus it is recommended to run the algorithm for upto 10 iterations, as that will give the best results.

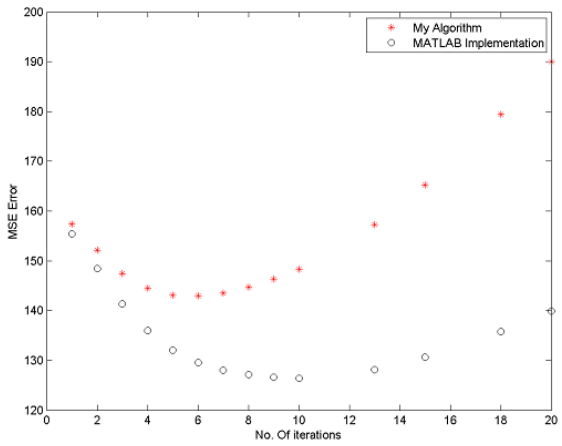

Figure 3.8: Comparison of convergence of *deconvlucy blind* for only blurred image

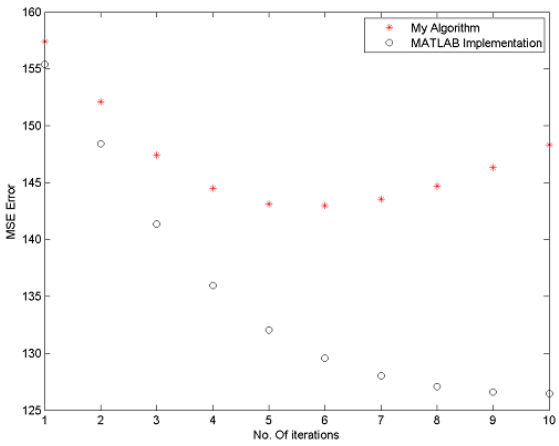

Figure 3.9: Comparison of convergence of *deconvlucy blind* for only blurred image

3.4 Effect of kernel size on Image and PSF estimation

After performing experiments to see the convergence of the RL algorithm (Algorithm 4), effect of other parameters on the restoration of images were studied. One such parameter was kernel estimation and size.[2]

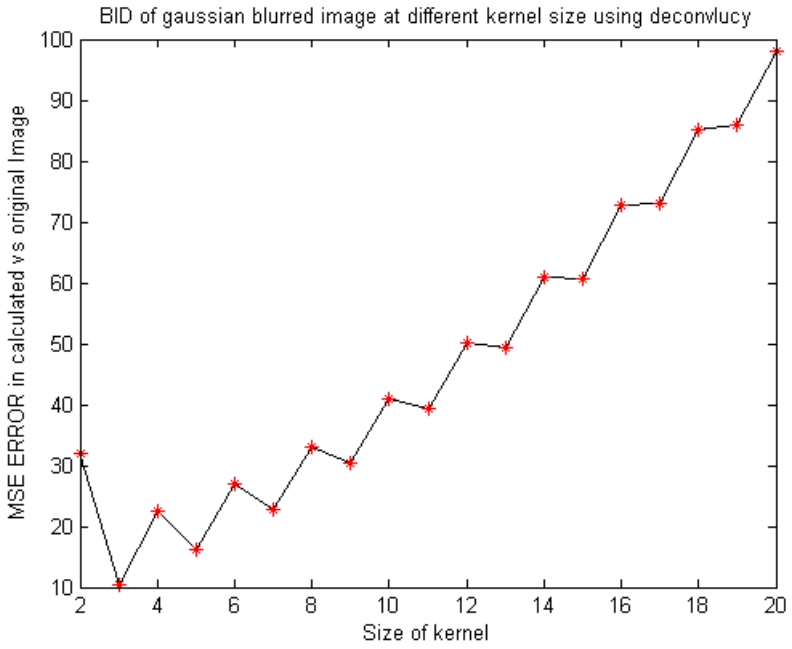

Figure 3.10: MSE in *Blind* image restoration using RL algorithm at different kernel sizes in image estimation

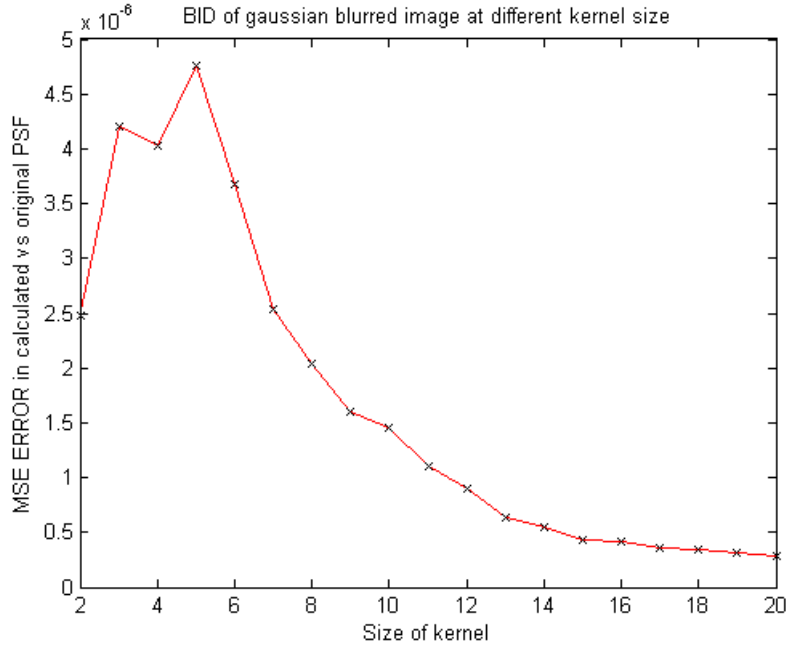

Figure 3.11: MSE in *Blind* image restoration using RL algorithm at different kernel sizes in kernel estimation

As the size of kernel increases the estimated image deviates from the original image. Also the kernel estimation for the RL algorithm becomes increasingly difficult. Graph 3.10 provides us an insight as to how the image estimation is affected by the kernel size. The MSE decreases for odd value of kernel size for smaller values. This is a direct implication of the fact that the Gaussian kernel size is chosen to be $\lfloor 6 * \sigma \rfloor + 1$, which indeed be always be odd. This further is an implication of the fact of the 99% *confidence interval*.

Graph 3.11 shows that the kernel estimate has minimum effect due to the change in size of the kernel. As the kernel size increases, that is the energy of the kernel spreads, the estimation gets better.

3.5 C++ Implementation

For converting the MATLAB implementation of the RL algorithm initially MATLAB's *C coder* was used. But since the use of some MATLAB's *image processing toolbox* function, which do not have a direct conversion to C++, the output needed was not obtained. We then used *opencv* image processing modules to implement the Algorithm 4. The implementation was straight forward with the *opencv* functions available. The implementation uses legacy image storing matrices *IplImage* though as they have better documentation compared to newer available *cv::Mat*. Also the code initially ran only on *Linux* platforms but after the inclusion of windows.h, the code was made to run on Windows platforms as well.

In Figure 3.12 (right) , the original image (Figure 3.12 (left)) was blurred using a Gaussian blur kernel. The output of the C++ implementation (Figure 3.14) and *deconvlucy* MATLAB (Figure 3.13) are shown. The results are comparable and the difference in the output not so distinguishable to naked eye.

The image in Figure 3.15 (left) was blurred with a motion blurring kernel(Figure 3.15 (right)) . The output of the C++ implementation (Figure 3.14) and *deconvlucy* MATLAB (Figure 3.13) are shown. The output of the *deconvlucy* MATLAB is better than the C++ output because of the implementation of edgetaper which reduces the ringing around the edges[11].

The last experiment was performed on RGB images. As earlier the original RGB image was converted to YCbCr image and then the blur kernel was applied on the Y component (luminance). The output of

the *deconvlucy* `MATLAB` and `C++` implementation are visibly similar. The difference in the color rendering is not different either.

Figure 3.12: Original image(left) and degraded image(right)

Figure 3.13: Output of *deconvlucy* `MATLAB`

Figure 3.14: Output of `C++` implementation of RL

Figure 3.15: Original image(left) and degraded image(right)

Figure 3.16: Ouput of *deconvlucy* `MATLAB`

Figure 3.17: Output of `C++` implementation

Figure 3.18: RGB text image(left) and degraded image(right)

Figure 3.19: Output of *deconvlucy* `MATLAB`

Figure 3.20: Output of `C++` implementation

Bibliography

[1] B. K. Gunturk, "Fundamentals of image restoration," in *Image Restoration:Fundamentals and Advances* (B. K. Gunturk and X. Li, eds.), pp. 25–61, Boca Raton: CRC Press, 2013.

[2] M. Šorel and F. Šroubek, "Restoration in presence of unknown spatially varying blur," in *Image Restoration:Fundamentals and Advances* (B. K. Gunturk and X. Li, eds.), pp. 63–87, Boca Raton: CRC Press, 2013.

[3] J. W. Woods, J. Biemond, and A. M. Tekalp, "Boundary value problem in image restoration," in *Proceedings of the IEEE International Conference on Acoustics,Speech, and Signal Processing*, vol. 10, pp. 692–695, April 1985.

[4] M. K. Ng, R. H. Chan, and W. cheung Tang, "A fast algorithm for deblurring models with neumann boundary conditions," *SIAM Journal Of Scientific Computing*, vol. 21, pp. 851–866, December 1999.

[5] R. Fletcher, *Practical Methods of Optimization*. New York: Wiley, 2000.

[6] A. Papoulis, *Probability Random Variables and Stochastic Processes*. New York: McGraw-Hill, 1991.

[7] W. H. Richardson, "Bayesian-based iterative method of image restoration," *Journal of the Optical Society of America*, vol. 62, pp. 55–59, January 1972.

[8] L. B. Lucy, "An iteration technique for the rectification of the obscured distribution," *The Astronomical Journal*, vol. 79, pp. 745–754, June 1974.

[9] D. S. C. Biggs and M. Andrews, "Acceleration of iterative image restoration algorithms," *Applied Optics*, vol. 36, no. 8, pp. 1766–1775, 1997.

[10] R. Hanisch, R. White, and R. Gilliland, "Deconvolutions of hubble space telescope images and spectra," in *Deconvolution of Images and Spectra* (P. Jansson, ed.), CA: Academic Press, 2 ed., 1997.

[11] M. Bertero and P. Boccacci, "A simple method for the reduction of boundary effects in the ricardson-lucy approach to image deconvolution," *Astronomy & Astrophysics*, pp. 369–374, 2005.

[12] K. M. Hanson, "Introduction to bayesian image analysis," in *Medical Imaging 1993*, pp. 716–731, International Society for Optics and Photonics, 1993.

[13] D. Fish, A. Brinicombe, E. Pike, and J. Walker, "Blind deconvolution by means of the richardson–lucy algorithm," *JOSA A*, vol. 12, no. 1, pp. 58–65, 1995.

[14] H. Bi and G. Boerner, "When does the Richardson-Lucy deconvolution converge?," *Astronomy and Astrophysics Suppl.*, vol. 108, pp. 409–415, Dec. 1994.

[15] L. Lamport, *I1 (\ LaTeX)—A Document*, vol. 410. pub-AW, 1985.

[16] MATLAB, *version 8.0.0.783 (R2012b)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[17] M. S. Almeida and L. B. Almeida, "Blind and semi-blind deblurring of natural images," *Image Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 36–52, 2010.