TUHH
Technische Universität Hamburg

Master Thesis

Restoration and enhancement of reconstructed light field images of projection systems

Author

Jabir Mohamed Abdi

15 October 2021

1. Professor: Prof. Dr.-Ing. Thorster A. Kern
2. Professor: Prof. Dr.-Ing. Alexander Penn
Supervisor: M. Sc. Sravan Shelam

at the

INSTITUTE
FOR MECHATRONICS
IN MECHANICS

Abstract

Image restoration is an important and challenging aspect of image processing. During the image capturing and processing phase the resultant image may be degraded and may lose important information in the image. This thesis attempts to apply filters on 3D images that were captured from test objects and generated from a goniometer setup.

There were two methodologies used in this thesis for applying the filters on the 3D images. The first methodology involves applying noise filters on the 2D images captured by the goniometer setup and 3D filters are further applied on the 3D images generated by the goniometer setup. The second methodology involves applying 2D filters on the slices of the 3D images. The results are observed based on the slices and performance metrics is measured on the resultant image quality.

For the first methodology, the noise filter provides an improvement on the 3D image. The 3D filters applied to the 3D image did not further restore the 3D images. For the second methodology, the 2D filters were not able to restore the 3D images as expected.

In this thesis, two methodologies of filters were tested and the first methodology noise filter has shown improvement in restoring the 3D image. The main limitations of the thesis were the computation power currently available during the experiments and further tests and different methodologies were discussed such as the wavelet-based filters and convolutional neural networks.

Declaration of originality

I declare that I have submitted my master thesis to Prof. Dr.-Ing. Thorster A. Kern with the title

Restoration and enhancement of reconstructed light field images of projection systems

I have produced independently and without the use of any aids other than those indicated and I have marked as such literally or analogously from publications. The work has not been submitted to any examination office in the same or similar form or in extracts. I assure the submitted written version corresponds to the version stored on the enclosed medium.

_____

Date and signature

Contents

Symbol

Symbol	Name	Unit
$f(x,y)$	2D Image	-
$f(x,y,z)$	3D image	-

Acronyms

2D	2 Dimension
3D	3 Dimension
AWGN	Additive White Gaussian Noise
PSNR	Peak Signal Noise Ratio
RMSE	Root Mean Square Error
SSIM	Structural Similarity Index

1 Introduction

Images with noise and degraded is a common issue on imaging systems. The corrupted images may lose important information that can no longer be useful for interpretation. For instance, images captured from medical imaging systems such as Xray, CT-scan (etc.) might be degraded during the image acquisition process to the point that the images may no longer be useful in diagnosing patients.

Image restoration and enhancement techniques are an important image processing technique to recover as much as possible the image by using a prior knowledge of the degradation and applying the inverse process to recover the image.

In the institute of mechatronics and mechanics, has a light field measurement system that is used to measure image quality of devices such as head up displays and virtual reality headsets. The imaging systems captures light fields from an object target, reconstructs the light field and finally displays the reconstructed volume as a 3D image. The issues faced with the light measuring system is the precision of the imaging system, the lighting environment of the goniometer setup where the images are captured as well as the reconstruction algorithm that has deteriorated the resultant reconstructed volume.

This thesis attempts to restore the reconstructed volume by applying filters on the reconstructed volume and the captured images on the light field measurement system to recover the resultant 3D image. The outline of the thesis includes the second chapter that gives a description of the light field measurement system set up, the reconstruction algorithm and a review of the related works of the filters used for image restoration. The third chapter deals with the testing of the chosen filters and measurement of their performances and finally the fourth and fifth chapter discusses the results and conclusion of the thesis.

2 Background

2.1 Image measurement system

In the institute of mechatronics and mechanics, a goniometer is set up to capture images for the reconstruction of 3D images. This will have future applications in heads up displays in the automotive sector. The goniometer has six degrees of freedom and is equipped with a luminance measuring camera to capture the images. A test object is placed on the end of the test stand and a set of images are captured of the object. The images are captured within different positions and orientations of the camera to reconstruct a 3-dimensional image of the object.

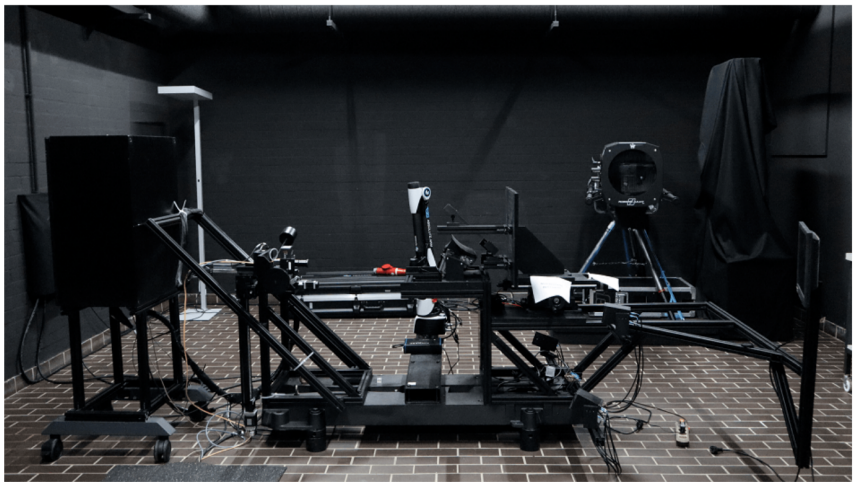


Figure 1: *Goniometer setup with camera on the left side.*

With the captured images, light field vectors are calculated from the pixels of each of the images. This method of generating the light field vectors involves geometric optics and the position and orientation of the camera [1]. The light field vectors are then used to generate the 3D image by using linear algebraic equations that are related to the intersection of lines and planes [2]. There is also another approach to generate the 3D image by using a line weighing approach [3].

The final volume reconstructed has distortions, this may be due to the inaccuracies of the positioning systems in the goniometer(translation 0.05mm and rotation 0.0002degrees) or by the initial estimate of the distance of the camera from the test object which results in affecting the reconstruction of the 3D image.

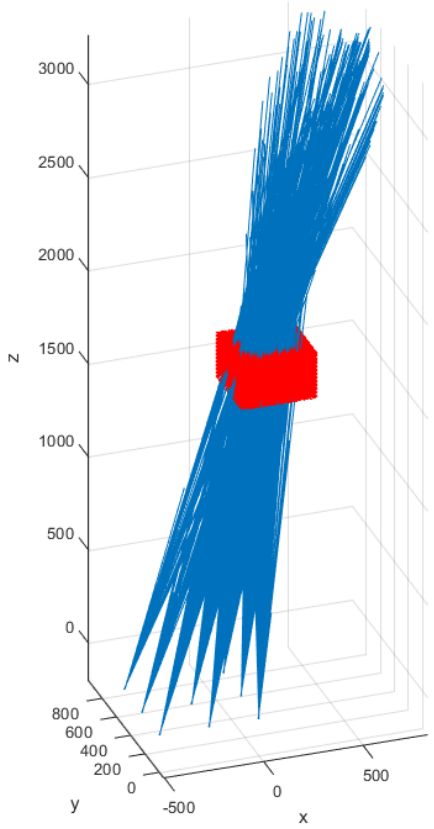


Figure 2: *Light fields generated (blue) and the reconstruction volume (red)*

2.2 Related works

Image restoration techniques attempt to recover a distorted image obtained from an imaging system by applying computations on images known as filters to recover the image. Digital image restoration had its first applications in the 1960s on astronomical imaging. The speed of the spacecraft and as well as the shutter speed of the camera distored the images by a motion blur. Currently, image restoration techniques are used in a variety of fields such as in the medical field on X-ray and CT-scans [4] [5] [6].

The degradation process and the restoration process of an image is illustrated in figure 3. When the image is captured, noise is introduced to the image due to factors such as the environment in where the image is captured. The image then is further degraded during the image acquisiton phase and is represented by the degradation function H. The degradation and noise are usually not fully understood and the main objective is to estimate the noise and degradation in ordet to apply an appropiate filter in an attempt to reduce the distortion in the image.

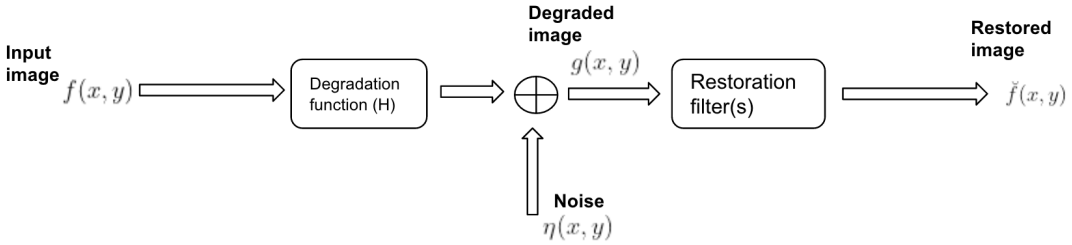


Figure 3: *Image degradation/restoration process*

Some of the classical image restoration techniques are the inverse filter, wiener filter, and iterative approaches. The inverse filter involves identifying the degradation function H and dividing the transform of the degraded image by the degradation transfer function. The filter is simple in implementation but does not take into account the noise in the image and may amplify the noise in the filtered image [7]. The Wiener filter is one of the earliest and best-known approaches to linear image restoration. It seeks to estimate the undistorted image by minimizing the error function. The wiener filter also suppresses noise by estimating the noise to signal power ratio [8]. Iterative approaches have the advantage of not knowing the exact degradation function and noise of the image and uses a recursive algorithm to restore the image. Iterative algorithms that are known are the least squares algorithm and recursive solutions [9].

The current and emerging techniques of image restoration are the use of neural networks and wavelets. Neural networks is a computing system inspired by biological neural networks. They are currently used in applications concerning machine learning [10] [11]. The applications of neural networks on image restoration are suitable as they can be easily adapted to the specific restoration problem. Wavelet-based filtering is a relatively new technique but emerging in image restoration applications. An example of one wavelet filtering is multifrequency channel decompositions of images and wavelet models [12].

3 Methodology

The two ideas workflow that has decided upon for the restoration of the reconstruction volume is illustrated on figures 4 and 5. For the first workflow idea,there is a noise filter section, which will involve applying noise filters to each of the images captured by the camera. With the images captured by the luminance camera now filtered, the images and resultant light field vectors go through the reconstruction algorithm in order to generate the reconstruction volume (3D image). Then finally 3D filters are applied to the reconstruction volume in order to get the final filtered 3D image.

The second workflow idea involves turning the reconstructed volume into slices and applying a 2D filter on each of the slices. The slices are then combined together again for the restored 3D image.

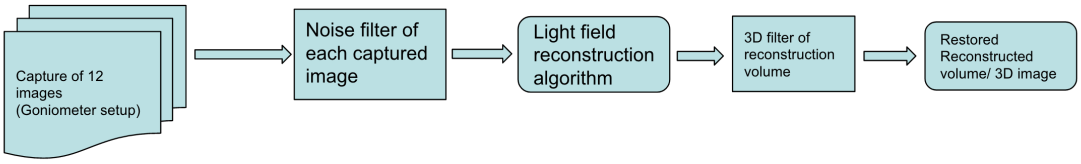


Figure 4: *First work flow for the restoration and enhancement of the reconstruction volume*

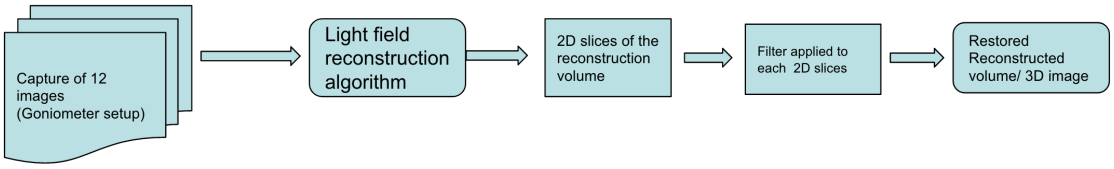


Figure 5: *Second work flow for the restoration and enhancement of the reconstruction volume*

3.1 Spatial and frequency domain filters

The term "filter" used in signal processing and image processing describes a mathematical operation that accepts or rejects certain frequency components of a signal. In the case of image processing, a low pass filter rejects higher frequencies of an image. This results in the blurring of an image.

The basic mechanics behind spatial filtering is that it has a pre-defined operation/equation and a mask. The mask is a rectangle that encompasses the neighbouring pixels

that is used to compute the result of the centre pixel in the mask. After the value of the centre pixel is computed, the mask then is shifted to other pixels in the image until all the pixels of the image have been computed. The figure 6 illustrates the mechanics of the spatial filter.

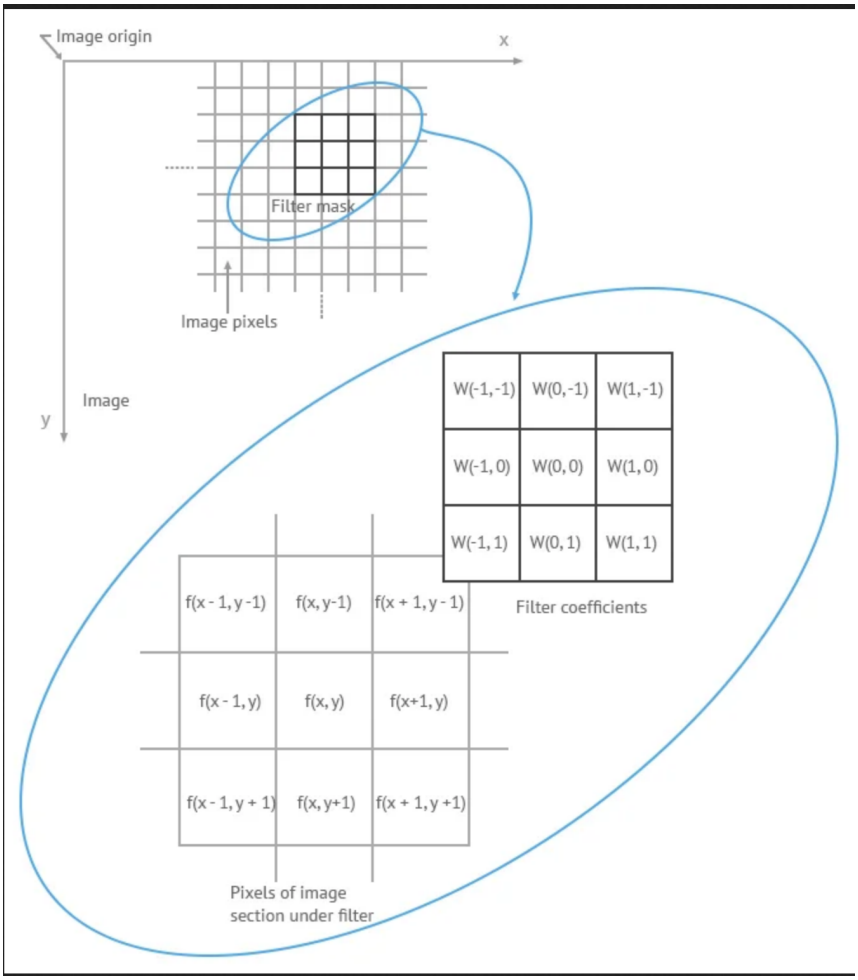


Figure 6: *Spatial filter mechanism the fitler mask operates on a neighborhood of pixels in order to compute the center pixel in the mask [7].*

For the frequency domain, the main difference with it is that spatial filtering computes the pixel values as it is while the frequency domain converts the image to its Fourier transform equivalent. The filter is then multiplied by the Fourier transform of the image and the result is then converted back by using the inverse Fourier transform.

$$g(x,y) = \mathscr{F}^{-1}\left[H(u,v),F(u,v)\right] \tag{1}$$

where $\mathscr{F}^{-1}$ represents the inverse discrete fourier transform, $g(x,y)$ represents the

filtered image , $H(u,v)$ and $F(u,v)$ represent the filter and the original image respectively. The frequency-domain filters can also be implemented in the spatial domain but the main advantage of using the frequency domain filters is that it is computationally less expensive and may also be more efficient.

3.2 First methodology

3.2.1 Noise filter

The main motivation behind applying noise filters on the captured is due to the initial anticipation for any noise that is captured by the luminance camera. The noise can originate from the environment in which the images are captured. For instance, when the images were captured for the basketball object, it required extra lighting to illuminate the object. This noise could pose an issue when the light field vectors are generated from the images. When the light field vectors are generated the luminance values from the pixels of the images may not represent the actual values due to the noise. This then results in a distorted 3D image after the light field vectors are then fed through the reconstruction algorithm.

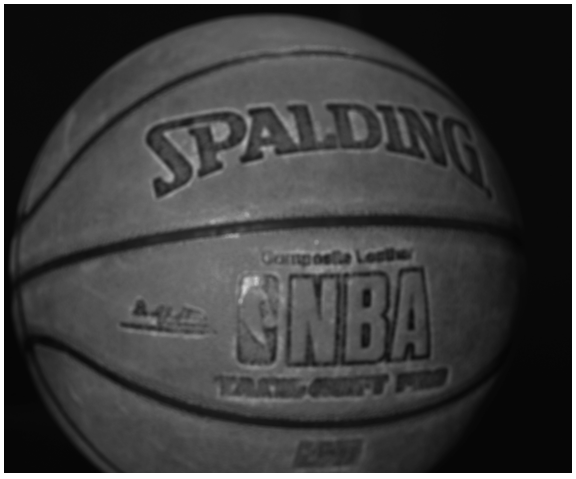


Figure 7: *Basketball captured by the luminance camera.*
The setup was illuminated by an external light source in order to capture the image

To attempt to filter out the noise, the noise distribution should be estimated to identify what noise models are present and to apply appropriately. In order to do this, a histogram is made of different sections of the image that have a uniform colour range. For instance, in the image illustrated on 8 the histogram shows the distribution of the

image pixel values and may give a glimpse into what kind of noise models are present in the images.

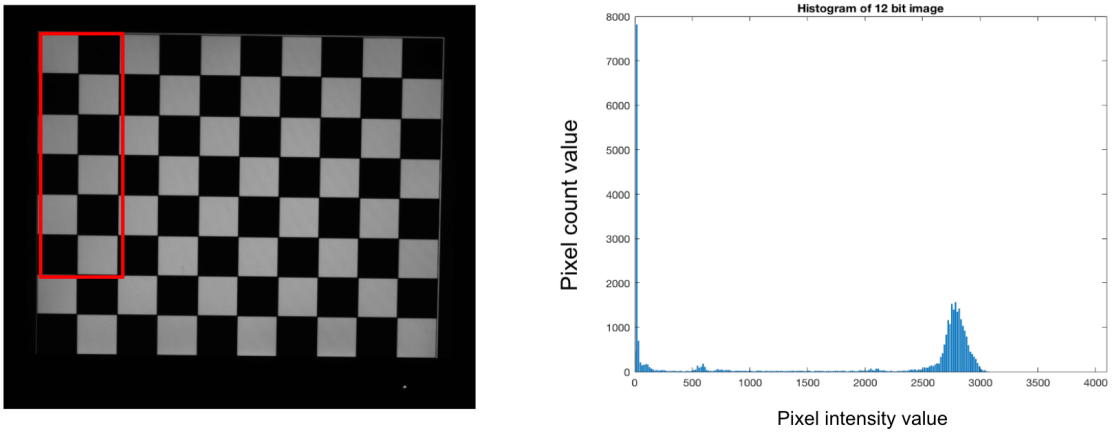


Figure 8: *Checkerboard image captured by the luminance camera(left).*
Pixel distribution based on the section of the checkerboard image highlighted by red (right)

From the images illustrated on 8, it is observed that the noise distribution varies based on the region of interest we are observing. It would have been expected that there are two sharp peaks representing the black and white pixels but the pixels distribution is slightly different, showing a gaussian noise model on the white pixels (pixel range of 2500-3000). With how the image characteristics change throughout the image, an adaptive filter is used. The choice made for the adaptive filter used in this is the adaptive local noise reduction

The adaptive local noise reduction filter [13] operates on a region or neighbourhood S_{xy} of an image. The result of the filter is based on the value of the noisy image $g(x,y)$, the variance of the noise σ_η^2, $z_{\bar{S}_{xy}}$ the local average intensity of the pixels S_{xy} and $\sigma_{S_{xy}}^2$, local variance of intensities of pixels in S_{xy}. The equation for the adaptive local noise reduction filter is expressed as:

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_{S_{xy}}^2}\left[g(x,y) - \bar{z}_{S_{xy}}\right] \tag{2}$$

In this equation,the main assumption is that the ratio of two variances does not exceed 1, which implies $\sigma_\eta^2 > \sigma_{S_{xy}}^2$. The equation is implemented and the code can be found in the appendix section of this report.

3.2.2 3D filters

After the captured images, filtered by the noise filters explained in the section 3.2.1 Noise filter, it is then used to generate light field vectors that are then used in the reconstruction algorithm to generate the 3D image (reconstruction volume). The 3D image is then filtered with a 3D filter in an attempt to further remove any distortions in the 3D image.

For the implementation of the filters, the knowledge of how the images are degraded and to estimate the degradation function are important in order to eliminate the distortion in the 3D image. However, the true degradation function is usually not completely known and a compromise has to be made in estimating the degradation function. In the current case of the reconstruction volume, an estimate of the gaussian blur will be implemented on the filters. The filters that are investigated in this project are blind deconvolution, iterative algorithm, wiener adaptive filter, non-local means filter and Wiener filter.

3.2.3 3D Wiener filter

The Wiener filter is a robust filter for image restoration. The Wiener filter seeks to estimate the restored image, which minimizes the statistical error function.

$$e^2 = E\left\{(f - \check{f})^2\right\} \tag{3}$$

where $E\{\bullet\}$ is the expected value of the argument. In the frequency domain, the 3D Wiener filter is expressed as:

$$\hat{F}(u,v,w) = \left[\frac{1}{H(u,v,w)}\frac{|H(u,v,w)|^2}{|H(u,v,w)|^2 + S_\eta(u,v,w)/S_f(u,v,w)}\right]G(u,v,w) \tag{4}$$

The meaning of the terms are as follows:

1. $\hat{F}(u,v,w)$ Fourier transform of restored image,
2. $G(u,v,w)$ Fourier transform of the degraded image,
3. $H(u,v,w)$ degradation transfer function,
4. $|H(u,v,w)|^2 = H^*(u,v,w)H(u,v,w)$,
5. $S_\eta(u,v,w)$ power spectrum of the noise,
6. $S_f(u,v,w)$ power spectrum of the undegraded image

The additional feature of the Wiener filter is that it can suppress noise by estimating the noise and inserting it to the terms $S_\eta(u,v,w)$ and $S_f(u,v,w)$. Without these two terms the Wiener filter turns into an inverse filter and may result into noise amplification and might not restore the image.

3.2.4 Non-local means filter

The non-local means filter is based on calculating a non-local average of all pixels in an image [14]. The filter gives a greater weight to pixels that are of similar structure and intensity when calculating the weighted average of the output pixels.

The figures 9 and 10 illustrates how the algorithm estimates the weight distribution based on the similarity of the pixels. The weight value ranges from 1(white) to 0 (black).

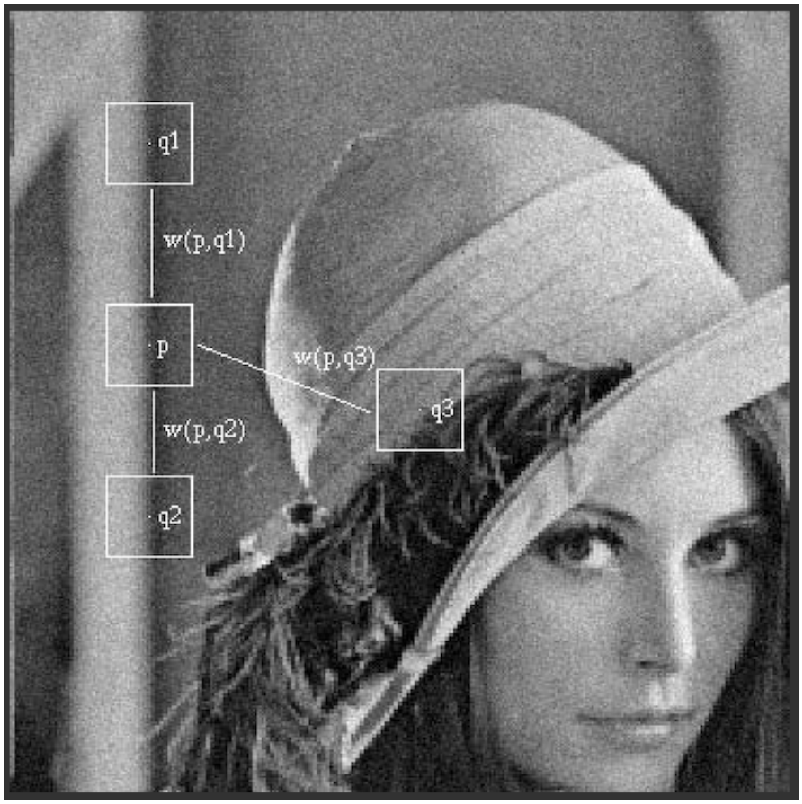


Figure 9: *NL means scheme.Similar pixels w(p,q1) and w(p,q2) have weight values close to 1 while w(p,q3) has weight value close to 0(A.Budaes et al)*

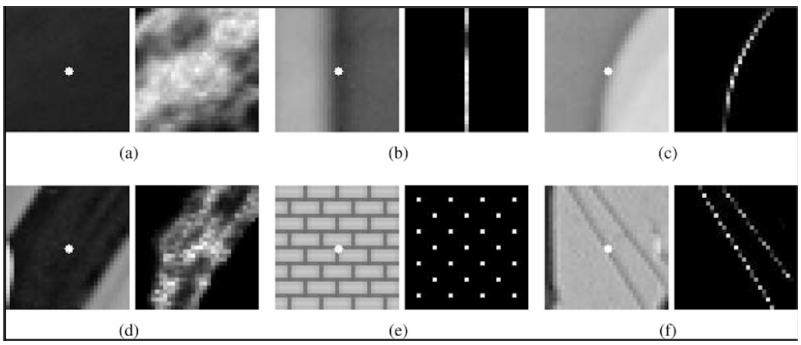


Figure 10: *NL means weight distribution, the pixels that are similar to the center pixel are represented by the white pixels with more weight.The dark parts represent the pixels with less resemblance with the center pixel(A.Budaes et al)*

The non-local means filter for 3D images[15] is illustrated by the equation below.

$$NL(u)(x_i) = \sum_{x_j \in V_i} w(x_i, x_j)\, u(x_j) \tag{5}$$

In which $u(x_j)$ is the gray level intensity of voxel j and the weight function $w(x_i, x_j)$ is the weight given of voxel i to a voxel j. The weight function is expressed as follows:

$$w(x_i, x_j) = \frac{1}{Z(i)} e^{-\frac{|u(N_i)-u(N_j)|_2^2,\sigma}{h^2}} \tag{6}$$

The term h is the degree of filtering, $u(N_i)$ and $u(N_j)$ are the volume neighbourhoods of voxel i and j, $|u(N_i) - u(N_j)|_2^2\sigma$ is the euclidean distance by weighted gaussian kernel standard deviation σ. The term $Z(i)$ is the normalizing constant which is expressed as:

$$Z(i) = \sum_j e^{\frac{|u(N_i)-u(N_j)|_2^2,\sigma}{h^2}} \tag{7}$$

The terms are the same as the ones expressed in equation (5).

3.2.5 Iterative algorithm

The iterative algorithm involves a repetitive application of the filter to the noisy image until the image converges to a restored image. The iterative algorithm used in this project is the Lucy-Richardson algorithm[16][17].

The Lucy-Richardson algorithm is based on a maximum likelihood estimation. Maximizing the likelihood function of the model results in an equation that is satisfied when the following iterative equation converges.

$$\check{f}_{k+1}(x,y) = \check{f}_k(x,y)\left[h(-x,-y) * \frac{g(x,y)}{h(x,y) * \check{f}_k(x,y)}\right] \tag{8}$$

The terms means are as follows: $\check{f}_{k+1}(x,y)$ is the next iteration estimate of the image, $\check{f}_k(x,y)$ is the current estimate image, $h(-x,-y)$ and $h(x,y)$ are the degradation function and $g(x,y)$ is the noisy image. The term $*$ represents convolution in the spatial domain. The main drawback of the algorithm is that the algorithm suffers from noise amplification if the number of iterations is increased to a certain extent.

3.2.6 Blind deconvolution

This approach of image restoration takes an initial estimate of the degradation function. The algorithm is based on the maximum likelihood estimation formulation the same as the Lucy-Richardson algorithm but the main difference is that this algorithm does not require a good estimate of the degradation function of the noisy image[18][19].

3.2.7 Adaptive wiener filter

The filter is a 3D noise adaptive filter for images that contain additive white gaussian noise (AWGN) [20]. Additive white gaussian noise is a basic noise model used in image processing to mimic the effect of random processes that occur in nature.

The filter estimates noise variances of each sub-band(slices along the z-axis) of the 3D image. The filter is an extension of a 2D adaptive wiener filter for 3D images. The output intensities of the voxels are calculated as follows:

$$O(k,l,m) = s + \frac{v^2 - \hat{\sigma}^2}{v^2}(J(k,l,m) - s), \tag{9}$$

where $J(k,l,m)$ represents the voxel intensity at specific co-ordinate, s is the local mean of intensities with a 3D mask achieved by:

$$s = \frac{1}{KLM}\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{m=1}^{M} J(k,l,m). \tag{10}$$

v is the local variance of intensities of the 3D mask obtained by:

$$v^2 = \frac{1}{KLM} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{m=1}^{M} J^2(k,l,m) - s^2 \tag{11}$$

$\hat{\sigma}^2(m)$ is the noise variance for a fixed slice on the z-axis. This is obtained by the average noise variance of the previous, current and following slice by:

$$\hat{\sigma}^2(m) = (\hat{\sigma}'^2(m-1) + \hat{\sigma}'^2(m) + \hat{\sigma}'^2(m+1))/3 \tag{12}$$

and finally $\hat{\sigma}'^2$ is noise variance estimated from the patches of an image slice represented as P and estimated by the minimal eigenvalue :

$$\hat{\sigma}^2 = \lambda_{\min}(\Sigma_J) \tag{13}$$

$$\sum_J = \frac{1}{P} \sum_{i=1}^{P} J_{ij} J_{ij}^T \tag{14}$$

J_{ij} is the image slice along the z-axis. Selection of the 3D mask volume is represnted in the equations by the values K,L,M represented the x,y and z size of the 3D mask volume.

3.3 Second methodology

For this methodology, the main idea of this method is to divide the 3D image after the reconstruction algorithm into slices along the z-axis. These Z-slices are then treated as 2D images in which each slice is applied a 2D filter. After applying the filter on the slices the slices are then combined to form the restored 3D image. For this method, the degradation function that is assumed to distort the 2D slices is the out of focus blur model using the method referred to in the article by Gajjar [21].

3.3.1 Out of focus blur

The out of focus blur model simulates a blurred image captured by a camera with a circular aperture. The out of focus blur model is expressed in the spatial domain as:

$$h(x,y) = \begin{cases} \frac{1}{\pi R^2}, x^2 + y^2 \leq R^2 \\ 0, \qquad otherwise \end{cases} \tag{15}$$

where R is the radius of the circle of confusion.

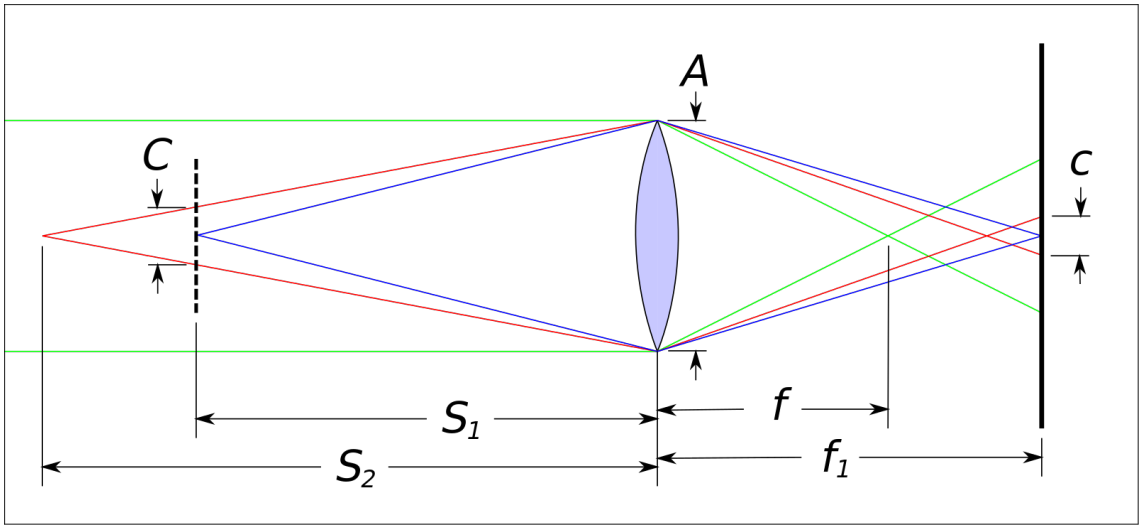


Figure 11: *Lens and ray diagram for calculating the circle of confusion diameter c for an out-of-focus subject at distance S2 when the camera is focused at S1.*

The fourier transform (Optical Transfer Function) is expressed by:

$$H(u,\, v) = \frac{J_1 \pi R r}{\pi R r} \tag{16}$$

where J_1 is the Bessel function of the first order, R is the radius of the blur and $r = \sqrt{u^2 + v^2}$ is the radius of the periodic circle in amplitude spectrumn $H(u,v)$.

The parameter R (blur raidus), is estimated by finding the radius of the innermost circle of the Fourier transform of the blurred image. The relation of R and the radius of the innermost circle of the Fourier transform of the blurred image is expressed as:

$$R = \frac{383 L_0}{2\pi r_0} \tag{17}$$

where $L_0 x L_0$ is the size of the fourier transform of the blurred image. To obtain the radius of the innermost circle, the circle hough transform is used on the Fourier transform of the blurred image. The circle Hough transform is an edge detection algorithm that detects the circular edges on an image and removes the other features in the image leaving only the circles of the image. After the hough transform is applied, using the

inbuilt MATLAB function (imfindcircles) to detect the centre circle and its radius. With the radius now known and with the previous equation (refer equation of r and R) based on the article by Gajjar [21] a polynomial expression is derived from relating the R with the radius of the innermost circle. It is expressed as:

$$R_{blur} = \begin{cases} -0.16427 * {R_{spec}}^5 + 5.96525 * {R_{spec}}^4 - \\ 84.90475 * {R_{spec}}^3 + 593.7951 * {R_{spec}}^2 - \\ 2047.95 * R_{spec} + 2844.03, \cdot \ if \ R_{spec} < 12 \\ \\ 2.39010e-06 * {R_{spec}}^4 - 0.000628 * {R_{spec}}^3 \\ +0.05973 * {R_{spec}}^2 - 2.54973 * R_{spec} + \\ 47.02154; \qquad otherwise \end{cases} \tag{18}$$

where R_{blur} is the radius of the blur, R_{spec} is the radius of the innermost circle of the fourier transform of the blurred image. With the R variable calculated, the value is included in the out of focus blur model in equation (15) and finally $H(u,v)$ is substituted into the 2D Wiener filter expressed as:

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} \right] G(u,v) \tag{19}$$

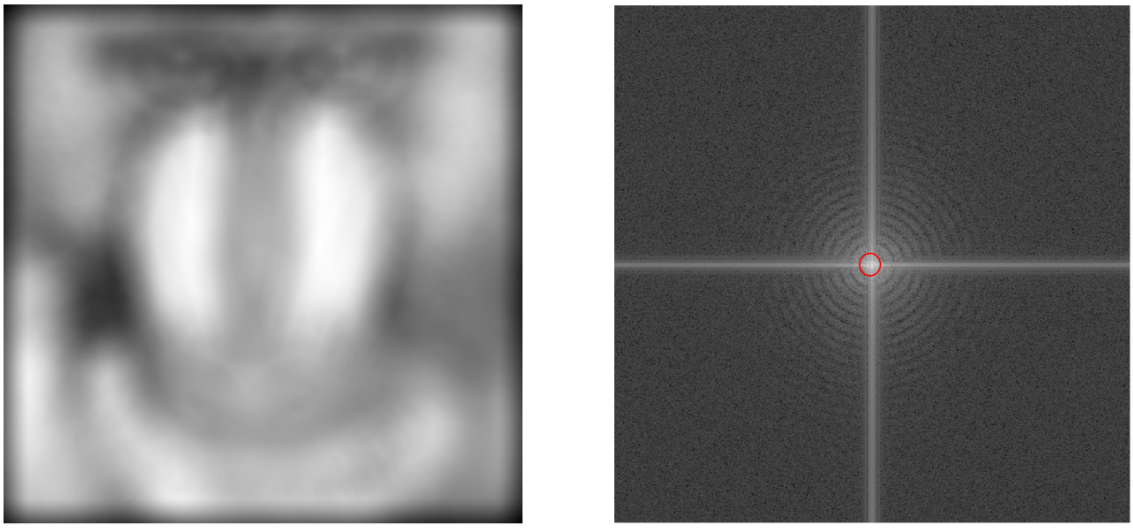


Figure 12: *Blurred image (left) and log of amplitude spectrum of blurred image (Right) and the innermorst circle for getting the radius R_{spec}*

4 Results and evaluation

In this section, the noise filter and 3D filters are applied to the test images and the reconstruction volume. For this experiment, the noise filters are applied to the images of the test objects before the light field vectors are generated and eventually have the reconstruction volume. The results are then observed from the slices along the z-axis of the reconstruction volume.

Figure 13: *Test object captured by the goniometer*

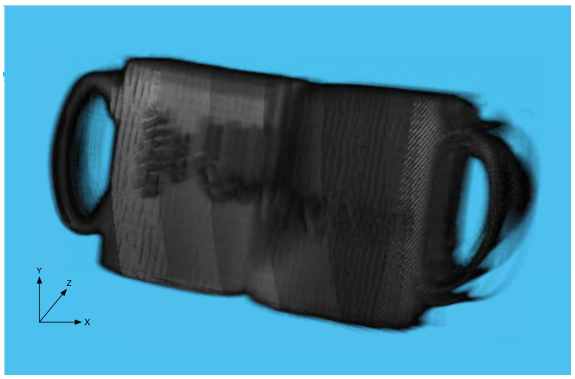


Figure 14: *3D reconstruction of the test object.The results of the reconstruction volume are observed as slices along the z-axis.*

For the reconstruction volume, there were changes made to the resolution of the reconstruction volume. Normally, the reconstruction volume should have a resolution of $1 \times 1 \times 1$ meaning that a 1mm unit will represent a voxel unit in the x-axis, y-axis and z-axis respectively. In the experiment conducted in this thesis, a compromise was made in reducing the resolution along the z-axis by 5. The main reason behind this is

to reduce the computation time since the reconstruction algorithm takes a significant amount of time. The result resolution of the reconstruction volume being $1 \times 1 \times 5$. That is 1mm unit per voxel on the x and y-axis and 5mm unit per voxel on the z-direction. The reason that the x and y-axis resolution were not reduced is that the slices along the z-axis can represent clear image slices with a sufficient resolution in the x and y-axis.

For the 3D filters, the reconstruction volume is prepared with the noise filter already implemented on the test images. The results of the filtered are compared by observing the resultant reconstruction volume, the slices of the resultant reconstruction volume and by using performance metrics. The performance metrics used in these experiments are namely the PSNR(Power Signal to Noise Ratio), SSIM (Structural Similarity Index), and MSE(Mean Squared Error) [22].

The power signal to noise ratio is between the maximum power of a signal compared to the maximum power of the noise signal (include reference). Higher values of PSNR is interpreted as a good value. The unit of the metric is in decibels and is expressed by :

$$PSNR = 20log_{10}\frac{L^2MN}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left[f(x,y)-g(x,y)\right]^2} \tag{20}$$

where, M and N are the dimensions of the image, L is the dynamic range of the image pixels, $f(x,y)$ is reference image and $g(x,y)$ is the restored image.

The Structural Similarity Index is a feature-based metric that compares the similarity between two images. The results range between 0 and 1, 1 indicating that the two images compared are the same and 0 indicating that the compared image is not at all structurally similar. The structural similarity index is calculated by:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{21}$$

where, μ_x and μ_y are the average of the reference and resotred respecitvely, σ_x^2 and σ_y^2 are the variances of reference and restored image, σ_{xy} is the covariance. Constraints $c1 = (k1L)2$ and $c2 = (k2L)2$. L = dynamic range of pixel values.

The Mean squared error is the error between the pixel values of the reference image to the restored image. This is used for interpreting similarity and lower values of the MSE indicate a good value that the restored image has greater similarity to the reference image. The MSE is expressed by:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left[f(x,y)-g(x,y)\right]^2 \tag{22}$$

where, M and N are the dimensions of the image, $f(x,y)$ is the reference image and $g(x,y)$ is the restored image.

4.1 First methodology results

4.1.1 Noise filter results

To test if the noise filter discussed in the section 3.2.1 on noise filter, the test object decided for it is a checkerboard pattern. The main idea of using the checkerboard is to have pixel values that are uniform(255= white and 0= black) to observe how the noise distribution is reduced when applying the filter.

The size of the filter was initially done with a small size of about 13x13 and it was gradually increased and the histogram is compared. The figure 15 and 16 illustrates how the distribution of the pixels has changed by the filter.

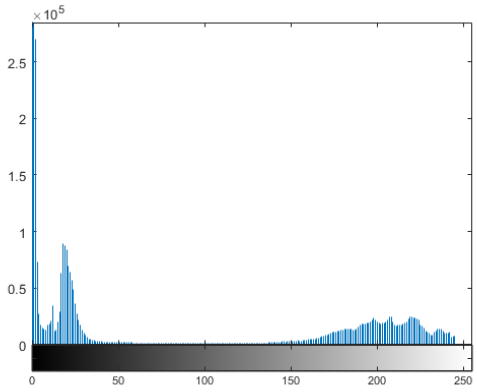


Figure 15: *Histogram of the test checkerboard image*

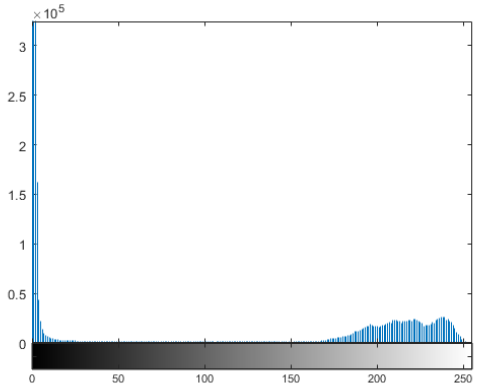
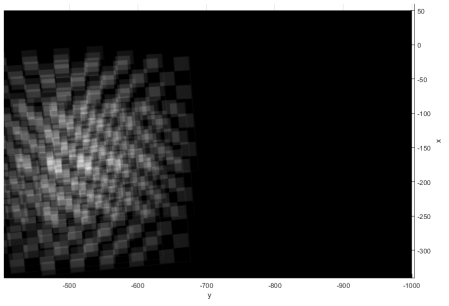
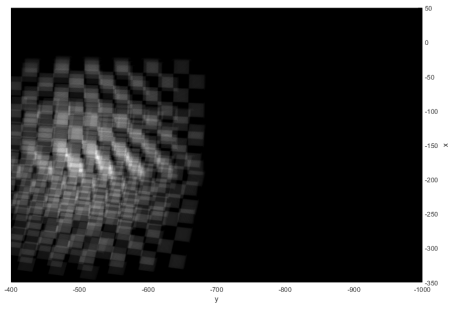


Figure 16: *Histogram of the applied noise filter on the test checkerboard image*

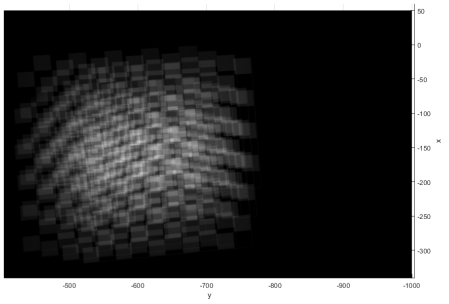
The filter now has reduced the noise distribution and the distribution of the pixel are now more centred towards the 255 value representing white and 0 representing the black pixels. With the images being filtered, the images are now reconstructed with the reconstruction algorithm and the reconstructed slices are observed on figure 17.



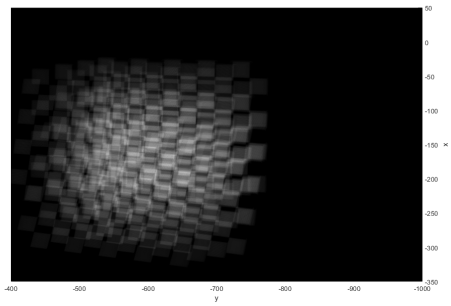
(a) *Noisy image 1500mm*



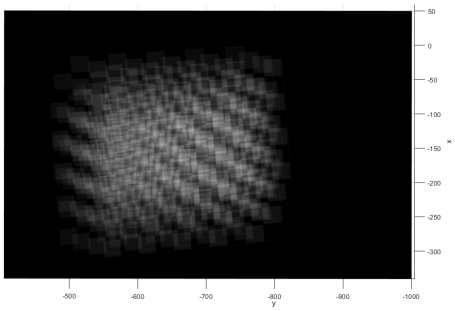
(b) *Filtered image 1500mm*



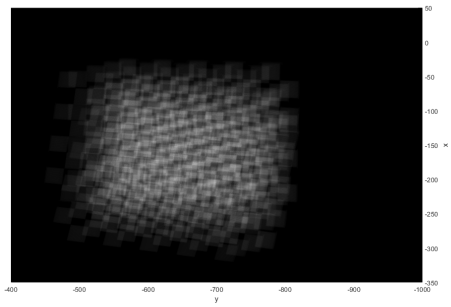
(c) *Noisy image 1700mm*



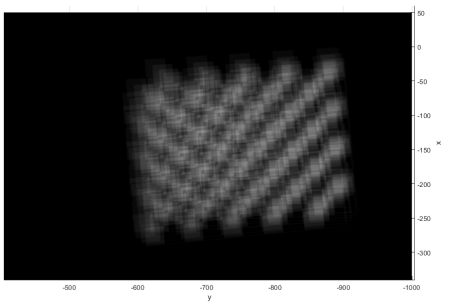
(d) *Filtered image 1700mm*



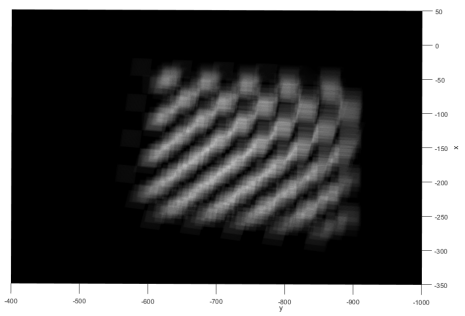
(e) *Noisy image 1800mm*



(f) *Filtered image 1800mm*



(g) *Noisy image 2000mm*



(h) *Filtered image 2000mm*

Figure 17: *Noisy image slices (left) and filtered image slices (right) along the z-axis*

For the results displayed on figure 17, the distortions are reduced as the image of the checkerboard pattern can be slightly seen. For instance, the doubling effect seems to be reduced at images (e),(f),(g) and (h). The adaptive local noise reduction filter is an adaptive filter that is capable of filtering out the noise based on neigbhorhood of the surrounding pixels.

4.1.2 3D filters results

For the 3D filters, each of the 3D filters explained in the previous chapter will be implemented on the reconstruction volume which has the noise filter already implemented. The 3D reconstruction volume when filtered is compared along with the slices of the reconstruction volume.

The performance metrics is also documented for each filter. A comparison is also made with the two different reconstruction algorithms namely line weighing and linear algebra approach explained in the background section of this thesis.

4.1.3 Line weighing reconstruction algorithm results

For the line weighing algorithm, two test objects were used in this experiment. The test objects were a pair of cups and a basketball. 24 images were captured in the goniometer setup and the noise filter applied to each of the images before the light filed vectors were generated. The light field vectors are then used to reconstruct the 3D image by using the line weighing algorithm. The 3D filters applied and tested in this section are the Wiener filter, non-local means filter, iterative algorithm, adaptive wiener filter and blind deconvolution filters.

The 3D wiener filter requires a degradation function or point spread function in order to correct the distortion. For this experiment the decision made on the distortion was that the 3D images are effected by blurring. The gaussian blur, which is used as a general model to simulate blur was used as a degradation function on the wiener filter. The kernel size of the blur with the best result was 9x9 (see appendix on wiener filter). The results for the 3D wiener filter on both the cups and basketball are displayed on figures 20 and 21.

For the adaptive wiener the results of the filtered are displayed on figures 22 and 23 with both the cups and basketball images. The adaptive Weiener had a filter mask size of 20x20x20.

The non-local means filter is applied with the degree of filtering set as 2 and the filter size is 5. The results of the filter is in figures 24 and 25.

The blind deconvolution was implemented with the inbuilt MATLAB function 'blind'. The filter size and number of iterations were gradually increase until the filtered image no longer improves the image quality. The results shown on figures 26 and 27 are for the gaussian blur used as the degradation function on the blind deconvolution algorithm with the kernel size being 15 and the number of iterations 10.

The iterative algorithm is implemented with the inbuilt MATLAB function 'lucy'. Similar to the process of the blind deconvolution algorithm, the filter size is increased as well as the number of iteratiions until the image quality has no further improvements. The kernel size is 9 and the number of iterations reached 30.The results are shown on figures 28 and 29.

Figure 18: *Test object cups*

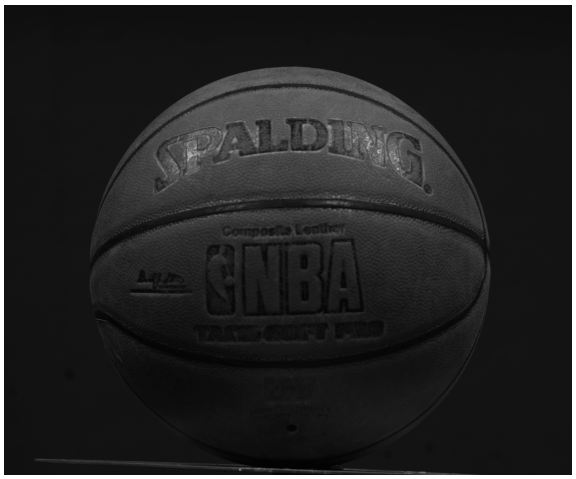
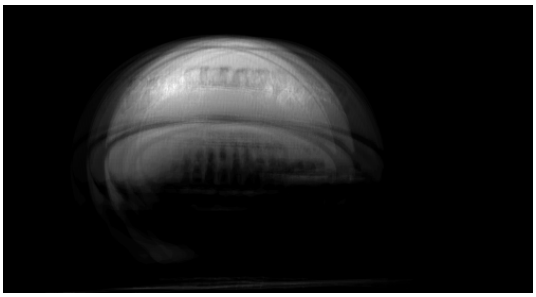
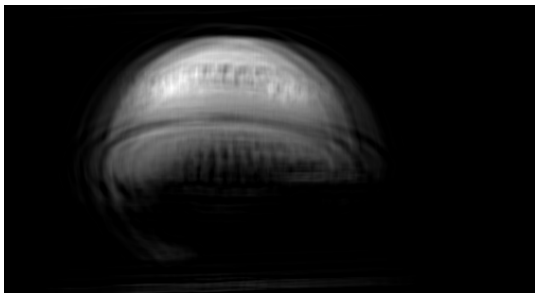


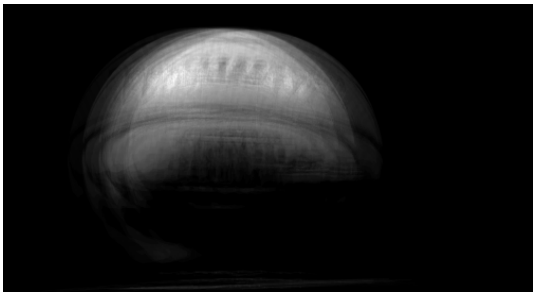
Figure 19: *Test object basketball*



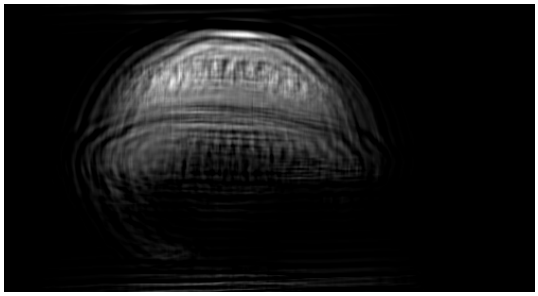
(a) *Noisy 3D image slice 2000mm*



(b) *Filtered image slice 2000mm*

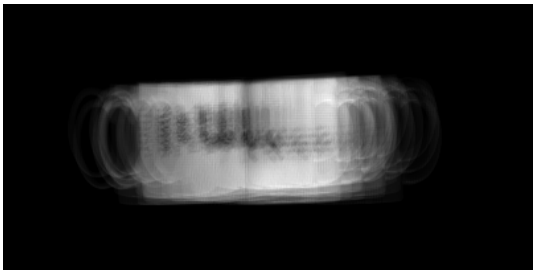


(c) *Noisy 3d image slice 2050mm*

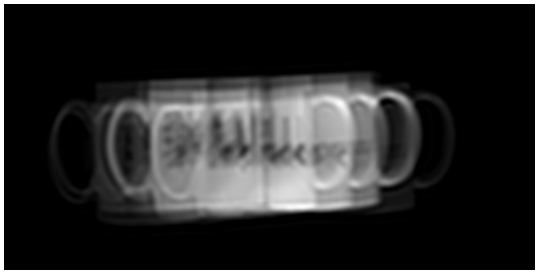


(d) *Filtered image slice 2050mm*

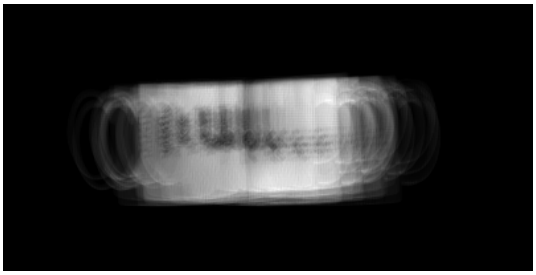
Figure 20: *Noisy image slices (left) and filtered image slices (right) along the z-axis for 3D Wiener filter*



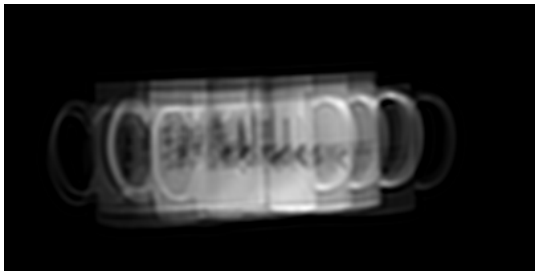
(a) *Noisy 3D image slice 1900mm*



(b) *Filtered image slice 1900mm*

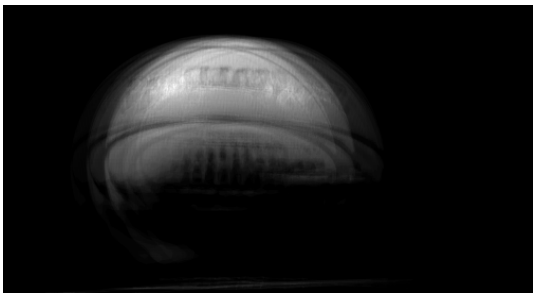


(c) *Noisy 3d image slice 2100mm*

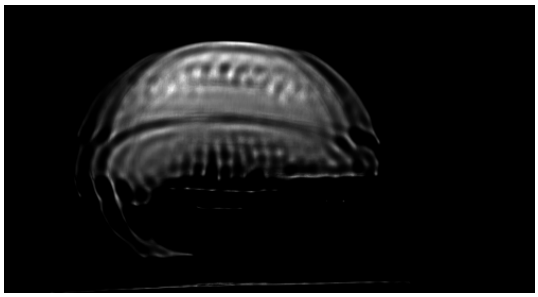


(d) *Filtered image slice 2100mm*

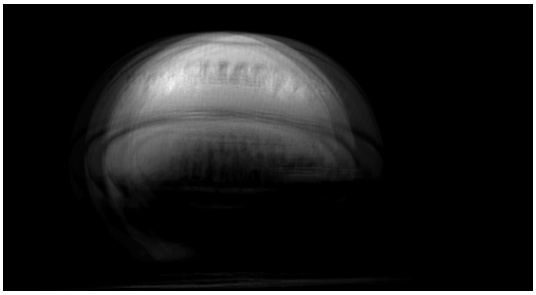
Figure 21: *Noisy image slices (left) and filtered image slices (right) along the z-axis for 3D wiener filter*



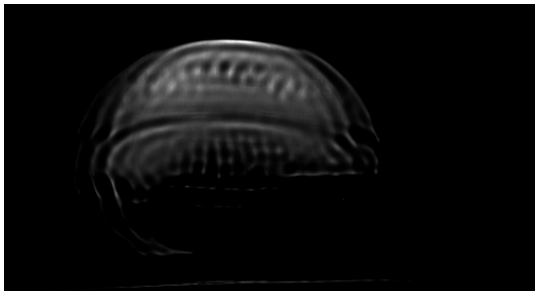
(a) *Noisy 3D image slice 2000mm*



(b) *Filtered image slice 2000mm*

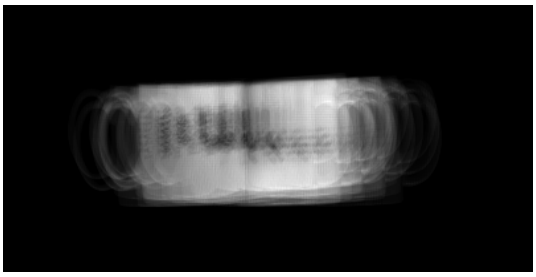


(c) *Noisy 3d image slice 2050mm*

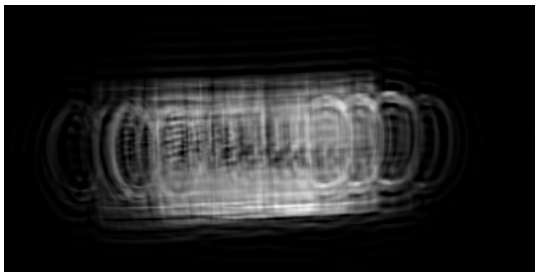


(d) *Filtered image slice 2050mm*

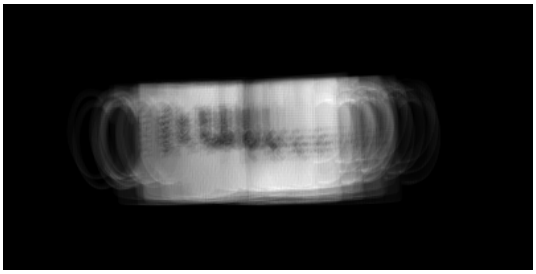
Figure 22: *Noisy image slices (left) and filtered image slices (right) along the z-axis for Adaptive wiener filter*



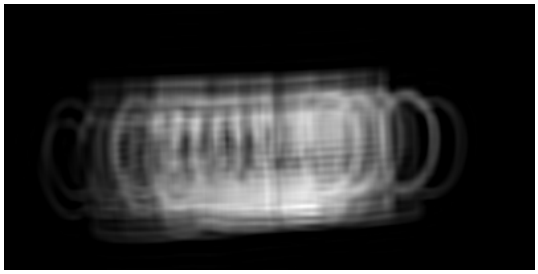
(a) *Noisy 3D image slice 1900mm*



(b) *Filtered image slice 1900mm*

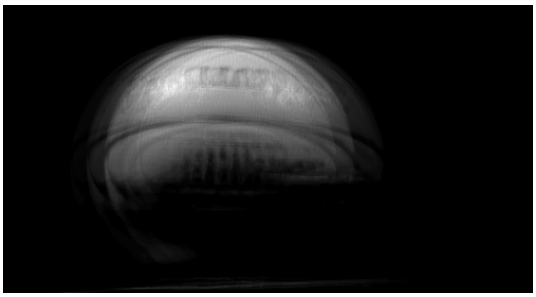


(c) *Noisy 3d image slice 2100mm*

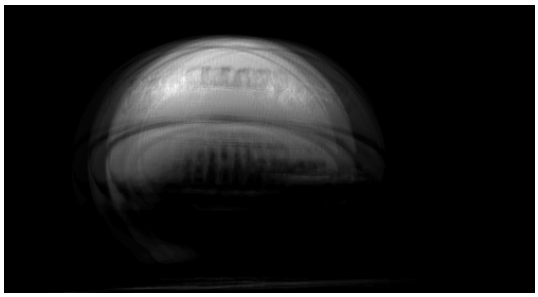


(d) *Filtered image slice 2100mm*

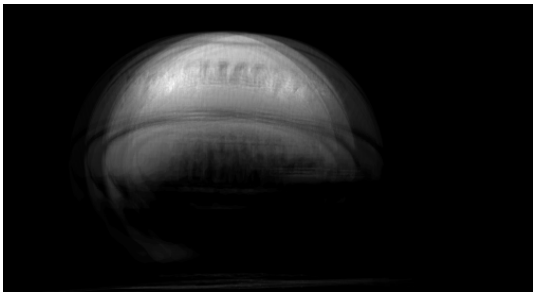
Figure 23: *Noisy image slices (left) and filtered image slices (right) along the z-axis for Adaptive wiener filter*



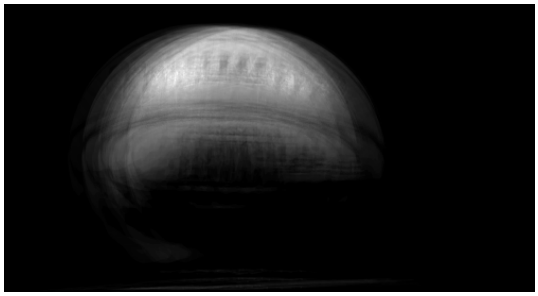
(a) *Noisy 3D image slice 2000mm*



(b) *Filtered image slice 2000mm*

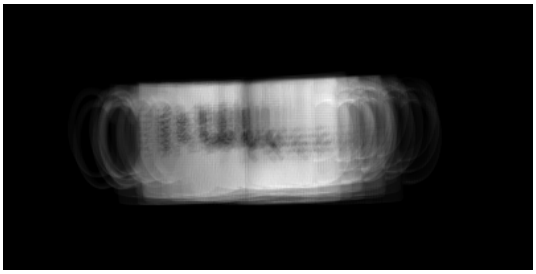


(c) *Noisy 3d image slice 2050mm*

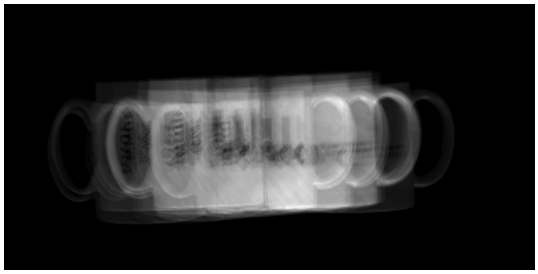


(d) *Filtered image slice 2050mm*

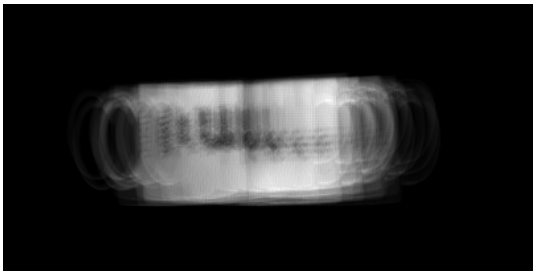
Figure 24: *Noisy image slices (left) and filtered image slices (right) along the z-axis for non- local means filter*



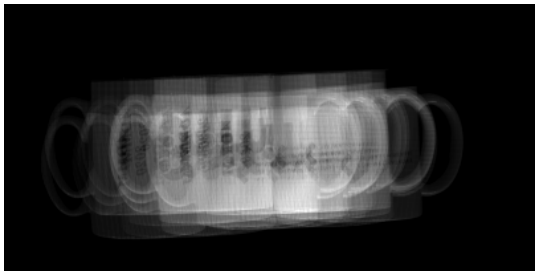
(a) *Noisy 3D image slice 1900mm*



(b) *Filtered image slice 1900mm*

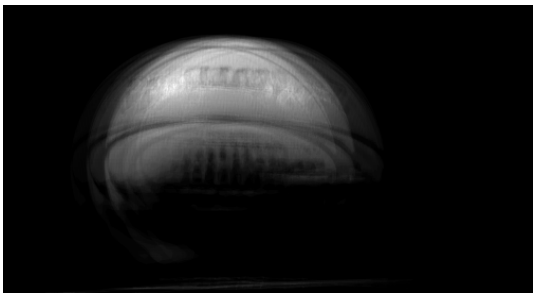


(c) *Noisy 3d image slice 2100mm*

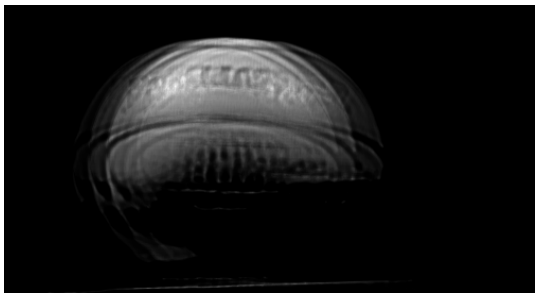


(d) *Filtered image slice 2100mm*

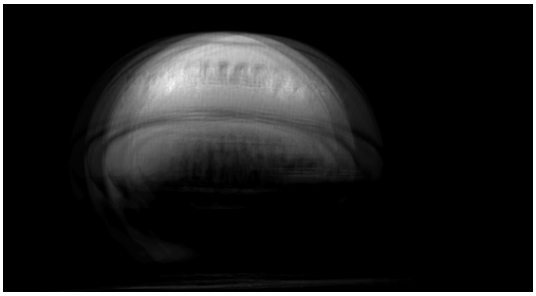
Figure 25: *Noisy image slices (left) and filtered image slices (right) along the z-axis for non-local means filter*



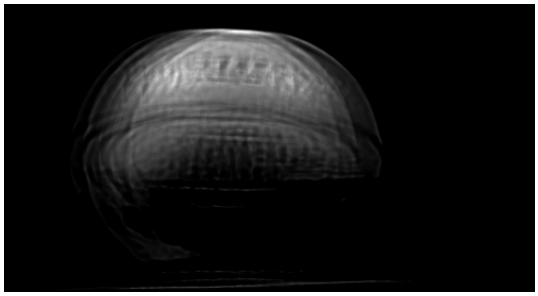
(a) *Noisy 3D image slice 2000mm*



(b) *Filtered image slice 2000mm*

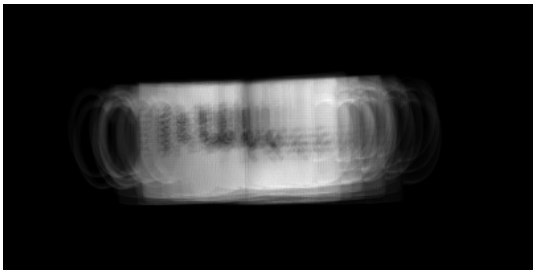


(c) *Noisy 3d image slice 2050mm*

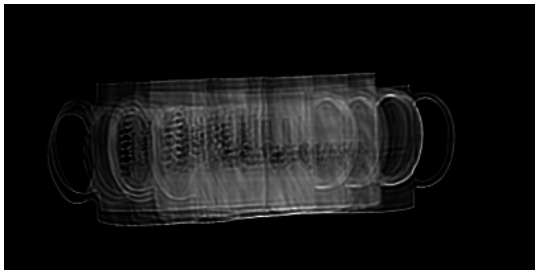


(d) *Filtered image slice 2050mm*

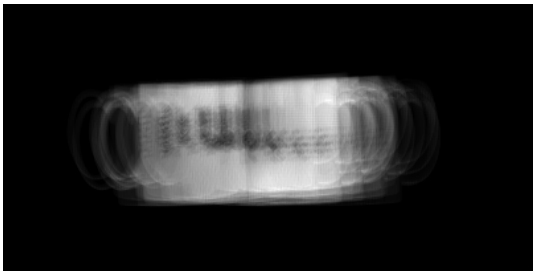
Figure 26: *Noisy image slices (left) and filtered image slices (right) along the z-axis for blind deconvolution algorithm*



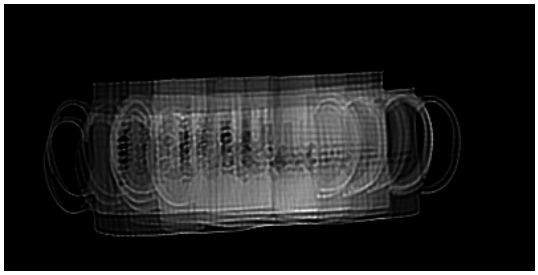
(a) *Noisy 3D image slice 1900mm*



(b) *Filtered image slice 1900mm*

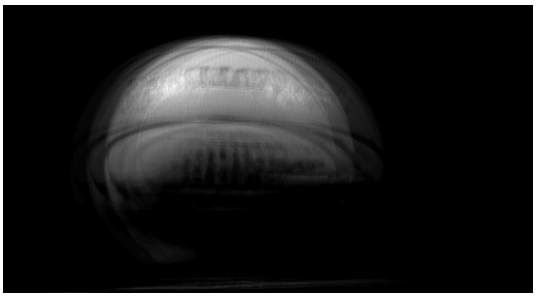


(c) *Noisy 3d image slice 2100mm*

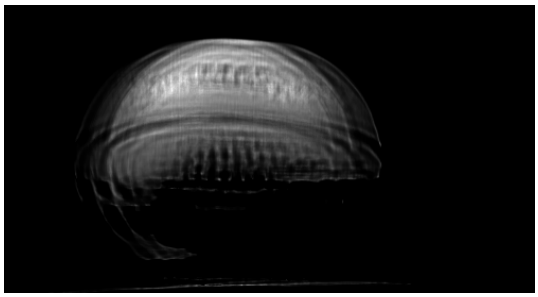


(d) *Filtered image slice 2100mm*

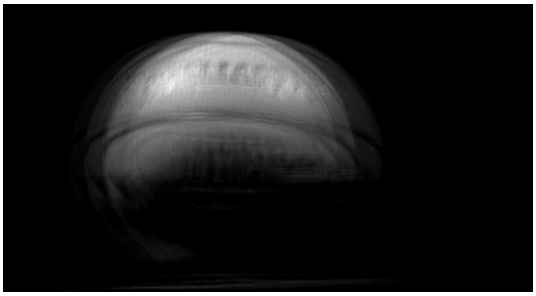
Figure 27: *Noisy image slices (left) and filtered image slices (right) along the z-axis for blind deconvolution algorithm*



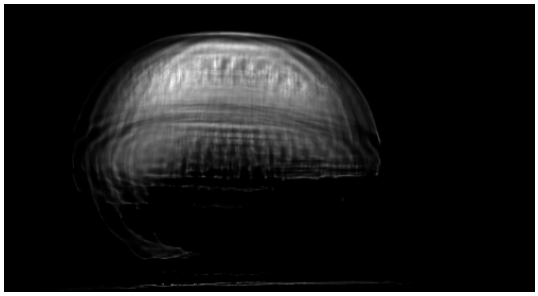
(a) *Noisy 3D image slice 2000mm*



(b) *Filtered image slice 2000mm*

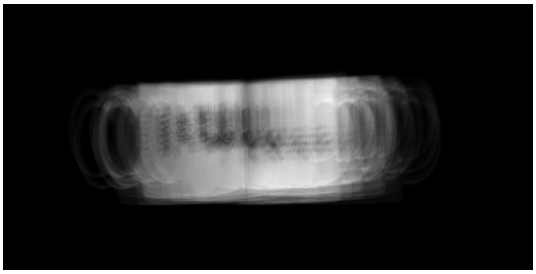


(c) *Noisy 3d image slice 2050mm*

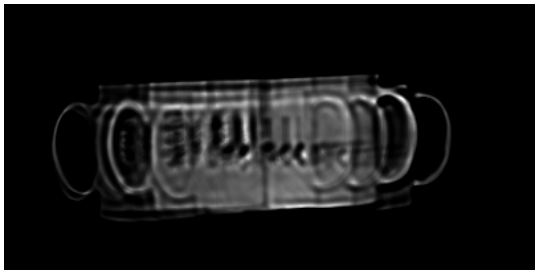


(d) *Filtered image slice 2050mm*

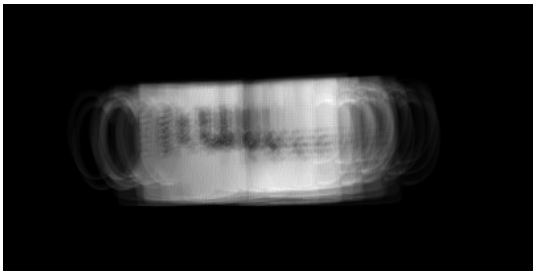
Figure 28: *Noisy image slices (left) and filtered image slices (right) along the z-axis for iterative algorithm*



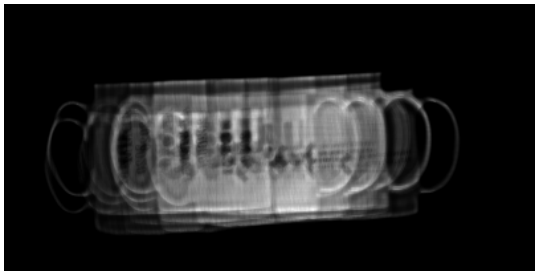
(a) *Noisy 3D image slice 1900mm*



(b) *Filtered image slice 1900mm*



(c) *Noisy 3d image slice 2100mm*



(d) *Filtered image slice 2100mm*

Figure 29: *Noisy image slices (left) and filtered image slices (right) along the z-axis for iterative algorithm*

A summary of the performance metrics for the 3D filters is provided on tables 3 and 4. The performance metrics shows that compared to the other filters the iterative algoirthim performed the best in terms of noise reduction (PSNR), NL means performed well in terms of the SSIM score and adaptive wiener filter performed well in the RMSE score.

Filter	PSNR	SSIM	RMSE
Adaptive wiener filter	29.34 dB	0.009	800.43
Blind	62.69 dB	0.004	1200.43
Iterative	65.71 dB	0.007	900.43
NL means	24.50 dB	0.02	834.62
Wiener fliter	27.43 dB	0.003	820.58

Table 3: *Table of performance metrics of basketball image*

Filter	PSNR	SSIM	RMSE
Adaptive wiener filter	27.54 dB	0.008	420.33
Blind	24.54 dB	0.007	388.35
Iterative	33.71 dB	0.0092	373.43
NL means	29.50 dB	0.06	444.62
Wiener fliter	22.43 dB	0.007	375.58

Table 4: *Table of performance metrics of cups image*

4.1.4 Linear algebra reconstruction algorithm approach

The 3D filters are tested with the reconstruction volume made from the linear algebra approach. The test object that was captured by the goniometer and used to generate the reconstruction volume is a basketball. The 3D filters tested are namely the the wiener filter, non-local means filter, iterative algorithm , adaptive wiener filter and blind deconvolution filters.

The 3D Wiener filter usees the gaussian blur as the main assumption behind the distortion is blurring. The kernel size with the best performance was of size 13x13. The results of the 3D wiener filter are displayed on figure 31.

The adaptive wiener fitler is implemented with the filter mask size of 20x20x20. The results of the adaptive wiener filter is presented in figure 32.

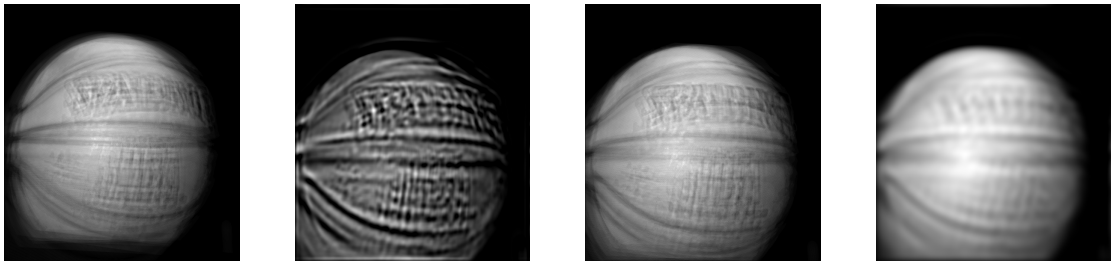
The non-local means filter is applied with the degree of filtering paramter as 2 and the filter size being 5. The results of the non-local means filter are displayed on figure 33.

The blind deconvolution filter is implemented with the kernel size of 15 and number of iterations 50 giving the best results. The results are displayed on figure 34.

The iterative algorithm result is displayed on the figure 35. The kenrel size with the best result was 9 with the number of iterations being 10.

The results are also summarised by the performance metrics on table 5. The performances of the filters are slightly different compared to the perormance metrics shown on tables 3 and 4. The adaptive wiener fitler has the best score on the PSNR, on the SSIM scoring the Wiener filter perfomed the best and finally on the RMSE score again the wiener filter perfomed the best compared to the other fitlers.

Figure 30: *Test object basketball*



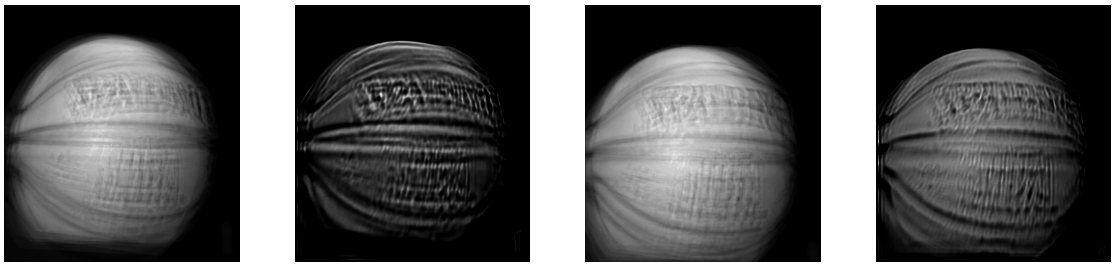
(a) *Noisy 3D image slice 1500mm*

(b) *Filtered image slice 1500mm*

(c) *Noisy 3d image slice 1600mm*

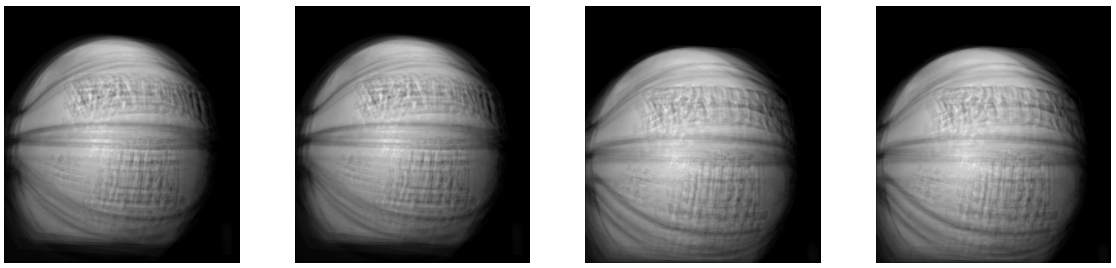
(d) *Filtered image slice 1600mm*

Figure 31: *Noisy image slices (left) and filtered image slices (right) along the z-axis for 3D wiener filter*



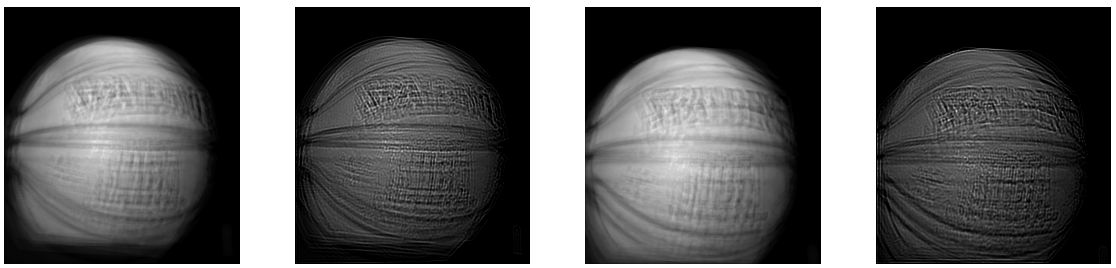
(a) *Noisy 3D image slice 1500mm* **(b)** *Filtered image slice 1500mm* **(c)** *Noisy 3d image slice 1600mm* **(d)** *Filtered image slice 1600mm*

Figure 32: *Noisy image slices (left) and filtered image slices (right) along the z-axis for adaptive wiener filter*



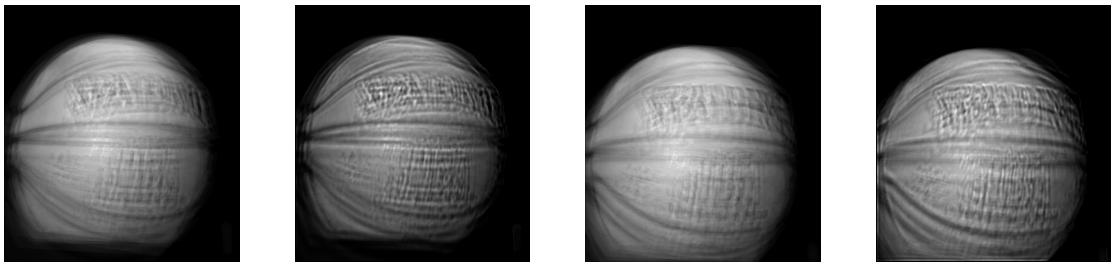
(a) *Noisy 3D image slice 1500mm* **(b)** *Filtered image slice 1500mm* **(c)** *Noisy 3d image slice 1600mm* **(d)** *Filtered image slice 1600mm*

Figure 33: *Noisy image slices (left) and filtered image slices (right) along the z-axis for non-local means filter*



(a) *Noisy 3D image slice 1500mm* **(b)** *Filtered image slice 1500mm* **(c)** *Noisy 3d image slice 1600mm* **(d)** *Filtered image slice 1600mm*

Figure 34: *Noisy image slices (left) and filtered image slices (right) along the z-axis for blind deconvolution algorithm*



(a) *Noisy 3D image slice 1500mm* **(b)** *Filtered image slice 1500mm* **(c)** *Noisy 3d image slice 1600mm* **(d)** *Filtered image slice 1600mm*

Figure 35: *Noisy image slices (left) and filtered image slices (right) along the z-axis for iterative algorithm*

Filter	PSNR	SSIM	RMSE
Adaptive wiener filter	35.78 dB	0.001	779.25
Blind	34.38 dB	0.003	800.43
Iterative	32.78 dB	0.004	668.43
NL means	37.43 dB	0.007	670.62
Wiener fliter	29.58 dB	0.008	665.29

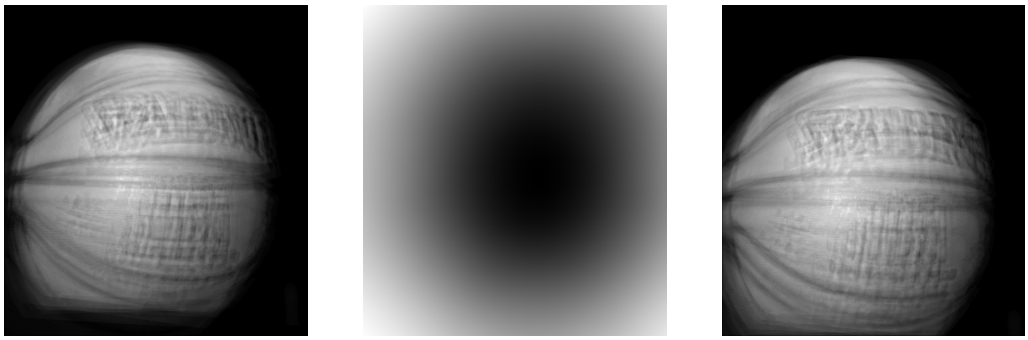
Table 5: *Table of performance metrics of basketball image*

4.2 Second methodology results

4.2.1 Out of focus blur results

The test object used in this experiment is the same one used in section 4.1.4 Linear algebra reconstruction algorithm approach. The test was carried out by obtaining the reconstruction volume as it has been described in section 3.3.1 out of focus blur, the reconstruction volume is divided into slices along the z-axis based on the resolution of the reconstruction volume and the Wiener filter is applied on each of the slices.

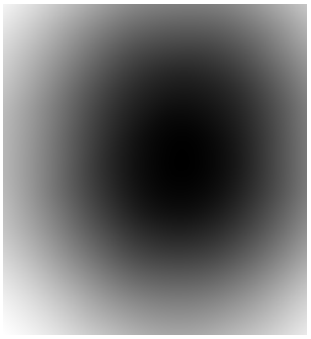
The filtered slices are then combined back again for the filtered reconstruction volume. The results are displayed on figure 36 with the slices of the filtered reconstruction volume. The performance metrics is also summarised on table 6



(a) *Noisy 3D image slice 1500mm*

(b) *Filtered image slice 1500mm*

(c) *Noisy 3d image slice 1600mm*



(d) *Filtered image slice 1600mm*

Figure 36: *Noisy image slices (left) and filtered image slices (right) along the z-axis for the out of focus blur method*

Filter	PSNR	SSIM	RMSE
Adaptive wiener filter	29.20 dB	0.001	832.36

Table 6: *Table of performance metrics of basketball image*

5 Conclusion and future works

In this thesis, different filters were tested to restore 3D images. The filters were tested with different methodologies and the results were measured with performance metrics.

The noise filter was shown to have a slight improvement in terms of reducing the noise in the images captured by the goniometer. The noise filter selected, which is the adaptive local noise reduction filter, is an adaptive filter that reduces the noise based on how the image characteristics change throughout the image. The choice of the noise filter seems adequate based on how the images are captured by the goniometer. The goniometer set-up could have different lighting conditions that could change how the images are captured e.g light bouncing off the images and causing some shinning effect on the objects which may affect the images.

The 3D filters tested in this thesis have overall poor performance in restoring the 3D image. Figures 20-35 show the restored reconstruction volume slices showing that the filters have no overall improvement on the image slices.

The out of focus blur method tested in this thesis, with the method described in the paper by Gajjar [23] has overall poor performance. The filtered reconstruction volume has further deteriorated. Figure 34 illustrate the reconstructed slice images.

The initial hypothesis that the reconstruction volume was distorted by a blurring effect in the reconstruction algorithm seems not to be the case. The degradation function of the 3D filters and the out of focus blur method presented in this thesis were implemented based on the assumption that the reconstruction volume.

The reduction of the resolution of the reconstruction volume is also a possible reason why the filters did not perform as expected. The reconstruction volumes resolution was reduced from $1x1x1$ to $1x1x5$ due to the limitation of computation power to generate the reconstruction volume. The resolution of the volume not being uniform could pose a problem in how the filters work on the reconstruction volume.

Some possible extension to the thesis work is attempting to implement the filters with a better resolution for instance the reconstruction volume being 1x1x1 or lower. With the improvement of the resolution of the reconstruction volume, the field of view could also be increased by reducing the distance of the test object from the luminance camera in the goniometer setup. The other possible filters that can be explored are the wavelet-based filtering methods for image restoration and convolutional neural networks in image restoration.

References

[1] Sravan Shelam and Thorsten Alexander Kern. “Lichtfeldmessung virtueller Bilder in einer Auflösung oberhalb der Wahrnehmungsgrenze”. In: *24. Europäischer Lichtkongress (LICHT 2021)*. 2021, pp. 506–515. ISBN: 978-3-927787-98-8. URL: http://hdl.handle.net/11420/10267 (cit. on p. 2).

[2] Abhinav Eluri. “Volume reconstruction using light field data”. In: (2020), pp. 1–51 (cit. on p. 2).

[3] Ahmad Maladan. “Light field volume reconstruction based on line weighting algorithm”. In: (2021), pp. 1–51 (cit. on p. 2).

[4] Rajesh Kochher, Anshu Oberoi, and Pallavi Goel. “Image restoration on mammography images”. In: *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016* (2017), pp. 1170–1173. DOI: 10.1109/CCAA.2016.7813894 (cit. on p. 3).

[5] Radovan Jiřík and Torfinn Taxt. “Two-dimensional blind Bayesian deconvolution of medical ultrasound images”. In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 55.10 (2008), pp. 2140–2153. ISSN: 08853010. DOI: 10.1109/TUFFC.914 (cit. on p. 3).

[6] Carlos Ortiz-de-Solorzano Oncology TXabier Artaechevarria Arrate Mu~noz-Barrutia. “RESTORATION OF BIOMEDICAL IMAGES USING LOCALLY ADAPTIVE B-SPLINE SMOOTHING Xabier Artaechevarria , Arrate Mu ~ Oncology Division , Centre for Applied Medical Research (CIMA), University of Navarra , Pamplona , Spain Dep . of Electrical , Electronic and ”. In: 1 (2007), pp. 425–428 (cit. on p. 3).

[7] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X (cit. on pp. 4, 6).

[8] Xin Li Bahadir Kursat Gunturk. *Image Restoration Fundamentals and Advances*. CRC Press, 2017. ISBN: 9781138071773 (cit. on p. 4).

[9] Reginald L. Lagendi Jan Bemond and Russel M. Mersereau. “Iterative methods for image deblurring”. In: (1990) (cit. on p. 4).

[10] S. Ghennam and K. Benmahammed. “Adaptive image restoration using Hopfield neural network”. In: (2001), pp. 569–578. DOI: 10.1109/NNSP.2001.943161 (cit. on p. 4).

[11] Dianhui Wang, T. Dillon, and E. Chang. “Pattern learning based image restoration using neural networks”. In: 2 (2002), 1481–1486 vol.2. DOI: 10.1109/IJCNN.2002.1007736 (cit. on p. 4).

[12] S.G. Mallat. “Multifrequency channel decompositions of images and wavelet models”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.12 (1989), pp. 2091–2110. DOI: 10.1109/29.45554 (cit. on p. 4).

[13] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X (cit. on p. 8).

[14] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising”. In: 2 (2005), 60–65 vol. 2. DOI: 10.1109/CVPR.2005.38 (cit. on p. 10).

[15] Song Hu and WenGuang Hou. “Denosing 3D Ultrasound Images by Non-local Means Accelerated by GPU”. In: (2011), pp. 43–45. DOI: 10.1109/ICBMI.2011.53 (cit. on p. 11).

[16] William Hadley Richardson. “Bayesian-Based Iterative Method of Image Restoration*”. In: *J. Opt. Soc. Am.* 62.1 (1972), pp. 55–59. DOI: 10.1364/JOSA.62.000055. URL: http://www.osapublishing.org/abstract.cfm?URI=josa-62-1-55 (cit. on p. 11).

[17] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. USA: Prentice-Hall, Inc., 2003, pp. 176–177. ISBN: 0130085197 (cit. on p. 11).

[18] Timothy J. Holmes. “Blind deconvolution of quantum-limited incoherent imagery: maximum-likelihood approach”. In: *J. Opt. Soc. Am. A* 9.7 (1992), pp. 1052–1061. DOI: 10.1364/JOSAA.9.001052. URL: http://josaa.osa.org/abstract.cfm?URI=josaa-9-7-1052 (cit. on p. 12).

[19] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. USA: Prentice-Hall, Inc., 2003, p. 180. ISBN: 0130085197 (cit. on p. 12).

[20] Ivo Rumenov Draganov and Rumen Parvanov Mironov. *3D Noise Adaptive Wiener Filtering of Images*. 2020. DOI: 10.1109/TELECOM50385.2020.9299543 (cit. on p. 12).

[21] Ruchi Gajjar and Tanish Zaveri. *Defocus blur parameter estimation using polynomial expression and signature based methods*. 2017. DOI: 10.1109/SPIN.2017.8049918 (cit. on pp. 13, 15).

[22] Samajdar T.and Quraishi M.I. *Analysis and Evaluation of Image Quality Metrics*. 2015. DOI: https://doi.org/10.1007/978-81-322-2247-7_38 (cit. on p. 17).

[23] Ruchi Gajjar and Tanish Zaveri. “Defocus blur parameter estimation using polynomial expression and signature based methods”. In: *2017 4th International Conference on Signal Processing and Integrated Networks, SPIN 2017* (2017), pp. 71–75. DOI: 10.1109/SPIN.2017.8049918 (cit. on p. 33).

A Appendix

A.1 MATLAB code

Listing 1: *NL means 3D filter*

```
%% NL means filter 3d
% Input function is I- 3dIMg to be filtered, V-search volume ,T- similary
% window, h- degree of filtering
function g= nlmeans(I,V,T,h)
% adjust the 3d images to gray scale 3d images
img= im2gray(I);
%Get the size of 3d image as m,n and k respectively
[m,n,k] = size(Img);

%Creating the search volume kernel
ks= 2*T+1;

ker= zeros(ks,ks,ks);

%Initialize search volume kernal
su=1;           %standard deviation of gaussian kernel
sm=0;           % sum of all kernel elements (for normalization)
for x=1:ks
    for  y=1:ks
        for z=1:ks
            a= x-T-1; % x distance of pixel from the center pixel evaluate
            b= y-T-1; % y distance of pixel from the center pixel to be evaulated
            c= z-T-1; % z distance of pixel from the cetner pixel to be evaluated
            ker(x,y,z) = exp(((a*a)+(b*b)+(c*c))/(-2*(su*su)));  %maybe multiply by 100?
            sm = sm + ker(x,y,z);
        end
    end
end
kernel = ker ./ f;
kernel = kernel / sm;   % normalization

%initialize output of 3D image
g=zeros(m,n,k);
```

```
%prepare noise array for filter processing by
%padding the 3D array
img2 = padarray(img,[T,T],'both');

%% Calculation output of single voxel

% initial for loop that will go through each individual voxel ↩
    I(r,s,t)
for r=1:m
    for s=1:n
        for t=1:k
            rm=r+T;
            sn=s+T;
            tn=t+T;

            % Neighborhood of evaluated voxel
            Wi=img(rm-T:rm+T,sn-T:sn+T,tn-T:tn+T);

            %Boundaries of search Volume
            vxmin= max(rm-V, T+1);
            vxmax= min(rm+V, m+T);

            vymin= max(sn-V, T+1);
            vymax= min(sn+V, n+T);

            vzmin= max(tn-V, T+1);
            vzmax= min(tn+V, k+T);

            % initial and calculated weight

            NL=0;
            Z=0;    %Normalizing constant

            % loop around search volume
            for x=vxmin:vxmax
                for y=vymin:vymax
                    for z=vzmin:vzmax

                      %Neighborhood of voxel 'j' being compared to↩
                          with voxel i
                       Wj= img(x-T:x+T, y-T:y+T, z-T:z+T);

                       %Calculate eucleidian distance
                       d2 = sum(sum(kernel.*(Wi-Wj).*(Wi-Wj).*(Wi-↩
                          Wj)));

                       % weight of similary : w(i,j)
                       wij= exp(-d2/(h*h));
```

```
                        % update Z and NL
                        Z= Z + wij;
                        NL= NL + (wij*img(x,y,z));

                         end
                    end
                end
                % Outuput of the Voxel calulated and the loop ↩
                    starts again for
                %new voxel
                g(m,n,k)=NL/Z;
            end
        end
    end

     % Convert to uint16
     g=uint16(g);

    end
```

Listing 2: *Adaptive 3d Wiener filter*

```
    %%function adaptive 3d Wiener filter

    function Z = wieneradap(G,R,M)
    % input for the function is G (3d image) R(patch size) and  M↩
        (3D mask)

    %MAke padarray if the input image is not a square matrix on x ↩
        and y

    %Note input matrix(3D matrix) on x and y should be square ↩
        matrix if not then padding
    %done to make it square matrix

    %Also test if R is multiple of 3d Matrix x and y

    % Create output same size as the input image
    Z = zeros(size(G));

    %Create for image planes
    J= zeros(size(G));

    %initialize covariance matrix
    cov= zeros(R,R);

    %Initialize result of loop for eigenvalue
    res=zeros(size(cov,1),size(cov,2),size(G,3));
```

```
%initialize minimal eigenvalue
var=zeros(1,size(G,3));

%Slices of Zaxis
for z=1:size(G,3)
   J(:,:,z)= G(:,:,z);
end

% % Create patches of every slice (Z axis) of image of size ↩
   RxR
% for z=1:size(G,3)
% J(:,:,z) = mat2cell(G(:,:,z), R*ones(1,size(G,1)/R), R*ones↩
   (1,size(G,1)/R));
% end

% create pathces of every slice (Z axis) of image RxR


% loop to make covariance matrix
for z=1:size(G,3)
    mt= mat2cell(J(:,:,z),R*ones(1,size(G,1)/R), R*ones(1,size↩
       (G,1)/R));
for i=1:size(G,1)/R
for p=1:size(G,1)/R    %?
cov= cov + mt{i,p}*transpose(mt{i,p});

end
end
% save result of every slice
res(:,:,z)= cov/(size(G,1)/R);

% Save variance of every slice as minimal eigenvalue of result
var(z)= min(eig(res(:,:,z)));
end

%% local mean intensistiers 2
% 3d mask M , should be odd integer mls


 mls=zeros(size(G));

G1= padarray(G,[M,M,M],0,'both');  %R R
%
for i=1+M:size(G1,1)-M
    for j= 1+M:size(G1,2)-M
        for k=1+M:size(G1,3)-M
              ls= zeros(M,M,M);    % initialize 3dmask
      ls= G1(i-((M-1)/2):i+((M-1)/2),j-((M-1)/2):j+((M-1)/2),k↩
```

```
            -((M-1)/2):k+((M-1)/2)); %take the intensity values ↩
            at 3d mask
          mls(i-M,j-M,k-M)=sum(ls(:))/(M*M*M); %calculate local ↩
             means intensities
          end
      end
  end

  %% local variance of intensites
  % calculate local variance with local mean intensities lvar , ↩
     with 3dmask R

  lvar= zeros(size(G));
  G2= padarray(G,[M,M,M],0,'both');  %R R
  Gs= G2.^2;
  for i=1+M:size(Gs,1)-M
      for j= 1+M:size(Gs,2)-M
          for k=1+M:size(Gs,3)-M
                las= zeros(M,M,M);    % initialize 3dmask
        las= Gs(i-((M-1)/2):i+((M-1)/2),j-((M-1)/2):j+((M-1)/2),↩
           k-((M-1)/2):k+((M-1)/2)); %take the intensity values↩
            at 3d mask
          lvar(i-M,j-M,k-M)=sum(las(:))/(M*M*M)- mls(i-M,j-M,k-M↩
             )^2; %calculate local variance intensities
          end
      end
  end

  %% calculate output of each intensity

  for r=1:size(G,1)
      for s=1:size(G,2)
          for t=1:size(G,3)
               if var(t)==1 || vart==size(G,3)
                   mvar(t)= (var(t)+var(t+1))/2;%mean var, ↩
                      previous current and next band %variance
               else
                   mvar(t)= (var(t-1)+var(t)+var(t+1))/3;
               end
              Z(r,s,t)=  mls(r,s,t)+ ((lvar(r,s,t)-mvar(t))/lvar↩
                 (r,s,t))*(G(r,s,t)-mls(r,s,t));
          end
      end
  end

  end
```

Listing 3: *Adaptive local noise reduction filter*

```
    function f = adaploc(B,M,N)

%B is the image to be filter and M and N is the size of the filter
%preferably in odd numbers.

B =double(B);
%Pad the matrix with zeros on all sides
C = padarray(B,[floor(M/2),floor(N/2)]);

sz = size(B,1)*size(B,2);

lvar = zeros([size(B,1) size(B,2)]);
lmean = zeros([size(B,1) size(B,2)]);
temp = zeros([size(B,1) size(B,2)]);
NewImg =  zeros([size(B,1) size(B,2)]);

for i = 1:size(C,1)-(M-1)
    for j = 1:size(C,2)-(N-1)


        temp = C(i:i+(M-1),j:j+(N-1));
        tmp =  temp(:);
             %Find the local mean and local variance for the local↩
                 region
        lmean(i,j) = mean(tmp);
        lvar(i,j) = mean(tmp.^2)-mean(tmp).^2;

    end
end

%Noise variance and average of the local variance
nvar = sum(lvar(:))/sz;

%If noise_variance > local_variance then local_variance=↩
   noise_variance

lvar = max(lvar,nvar);
 %Final_Image = B- (noise variance/local variance)*(B-local_mean);
 NewImg = nvar./lvar;
 NewImg = NewImg.*(B-lmean);
 NewImg = B-NewImg;

 %Convert the image to uint16 format.
 f = uint16(NewImg);

end
```

Listing 4: *Out of focus blur method*

```
%Test for out of focus blur

clear all
clc

% Load 3D image
 m= load('D:\Masterarbeit_Jabir\SravanAdded_Temp\↩
    Cups_CameraStraight_JabirSelected.mat');
 n= m.VoxelArray;
 img= voxelarraytest(n);

 %Sample slice of 3D image
 f= img(:,:,7);

 %Get the fft transform of the image

 F= fft2(f);
 S= fftshift(log(1+abs(F)));
 S=gscale(S);

%Get the padded size of original image for filtering

PQ=paddedsize(size(f));

[U,V]= dftuv(PQ(1), PQ(2));

%Estimate the radius for the out of focus blur model
% Displays the circle and select the radius from the workspace↩
    'radii'
[centers, radii]= imfindcircles(S,[6 10],'Sensitivity',0.95);
imshow(S)
viscircles(centers, radii,'Color','r');


%%
% Using out of focus blur model to get the filter H to use in ↩
   the wiener
% filter
H= focus(6.973857334126879,U,V);

%Loop for each slice of the 3dimage

G=zeros(size(img));
for i=1:size(img,3)

    f=img(:,:,i);

%Now use the wiener filter with small value in noise
```

```
%Theoretical value of snr
snr=10^(12/10);
K= 1./snr;

  W= wiener(H,K);
 g=dftfilt(f,W);

 G(:,:,i)=g;

end
% Convert the matrix file to gray scale image based on the range of values
% in the matrix G
G=mat2gray(G);

function H= focus(R,U,V)

%Bessel function of first kind nu=1 and the radius multiplied by square
%root of u2 and v2

%Estimate for blur
if R<12
r= -0.16427*(R^5) + 5.96525*(R^4)-84.90475*(R^3)+593.7951*(R^2)-2047.95*R+2844.03;
else
    r=2.39010*(exp(-6))*(R^4)-0.000628*(R^4)+0.05973*(R^2)-2.54973*R+47.02154;
end

x=r.*sqrt(U.^2+V.^2);

J1= besselj(1,x);
H=J1./x;

%Eliminate any NAn that can occur especially on H1,1()
H(isnan(H))=0;

end
```