Fourier Filtering for Removing Motion Blur in Images

Modern Optics – Keto

Joseph Simmons, Benjamin Topper, Avi Wolfson

5/4/2007

1. Introduction & Objectives

Almost every system or signal that scientists and engineers deal with can be viewed in several different ways. They can exist as a function of space, defined by physical parameters like length, width, height, color intensity, and others. They can exist as a function of time, defined by changes in any measurable characteristic. They can also exist as a function of frequency, defined by the composition of periodicities that make up light, sound, space, or any other dynamic system or signal. Furthermore, since analysis techniques differ depending on which domain the signal is being analyzed in, spatial and temporal signals can be converted into the frequency domain, or vice-versa, for mathematical convenience or more effective data acquisition. Fourier and LaPlace transforms are the functions used for the conversion between these domains.

Frequency domain analysis is performed by considering the individual frequency components of the full range of frequencies that one such signal is comprised of. A useful application for this method is in considering problems like motion blur in images. Since devices such as cameras don't capture an image in an instant, but rather over an exposure time, rapid movements cause the acquired image to have blur that represents one object occupying multiple positions over this exposure time. In a blurred image, edges appear vague and washed out meaning that over those areas their frequency components will be similar. Ideally, the edges would be sharp and that would be reflected by a significant frequency difference along those edges. This project explored the efficiency of using frequency domain techniques to remove motion blur from images.

The overall approach consisted of taking an image, converting it into its spatial frequencies, developing a point spread function (PSF) to filter the image with, and then converting the filtered result back into the spatial domain to see if blur was removed. This was performed in several steps, each of which built from having a greater understanding of the one preceding it. The first step was taking a normal (i.e. not blurred) image, creating a known blurring PSF, and then filtering the image so as to add blur to it. The next step was removing this blur by various methods, but with the information about the PSF that was used to create the blur. After that, de-blurring was performed without knowing anything about nature of the blurring PSF, except for its size. Finally, an algorithm was developed for removing blur from an already blurry image with no information regarding the blurring PSF.

2. Methods[1]

2.1 Blurring

The first process that was performed was creating a point spread function to add blur to an image. The blur was implemented by first creating a PSF filter in MatLab that would approximate linear motion blur. This PSF was then convolved with the original image to produce the blurred image. Convolution is a mathematical process by which a signal, in this case the image, is acted on by a system, the filter, in order to find the resulting signal. The amount of blur added to the original image depended on two parameters of the PSF: length of blur (in pixels), and the angle of the blur. These attributes were altered to generate different amounts of blur, but ultimately a length of 31 pixels and an angle of 11 degrees were found to add sufficient motion blur to the image.

[1] See Appendix 5.1 for all code segments used in implementing these algorithms

2.2 Known PSF Deblurring

After a known amount of blur was introduced into the image, an attempt was made to restore the now blurred image to its original form. This was done using several algorithms. In our treatment, a blurred image, i, results from:

$$i(x)=s(x)\otimes o(x)+n(x)$$

Where s is the point spread function and is convolved with the 'perfect' image o. In addition, some additive noise, n, may be present. The de-blurring algorithms used here (Lucy-Richardson, Wiener, and regularized), all attempt to get o, from the equation above by means of deconvolution.

The Weiner filter is an inverse filter that employs a linear deconvolution method. Linear deconvolution means that the output (o) is a linear combination of the input. With an inverse filter we imagine that there exists inverse Fourier transform of a transfer function y(x) such that

$$o(x)=y(x)\otimes i(x).$$

We may change the 1st equation to the frequency domain by using a Fourier transform. Neglecting noise, we then have:

$$I(\omega)=\tau(\omega)O(\omega).$$

Now the image may have a band limit Ω. In this case it is not good to work close to this limit. It has been found that the optimum band width, Ω_p is given by:

$$\Omega_p = \Omega\left[1 - \sqrt{\frac{\phi_n}{\phi_o}}\right]$$

Where ϕ_o and ϕ_n are the power spectra of the object and noise, respectively.
Given all of this, Norbert Wiener found the optimum transfer function to be

$$Y(\omega) = \frac{\tau^* \phi_o}{|\tau|^2 \phi_o + \phi_n}$$

So, for the restored image, we have

$$O(\omega)=Y(\omega)\ I(\omega)$$

Because the Weiner filter is a linear filter, it is computationally less intensive but it also gives poorer results when noise is introduced. Higher quality filters, such as the Lucy-Richardson, are non-linear.

The Lucy-Richardson algorithm is derived from counting statistics by means of maximizing the likelihood of the solution. The image i is given by

$$i(x)=o(x)\otimes s(x)$$

It is assumed that the data in the image i(x) are distributed about the mean, $\bar{i}(x)$, of some probability distribution $P(i|\bar{i})$. We define the likelihood log to be the natural log of the probability and we seek to maximize this quantity. From this we get the Lucy-Richardson iteration

$$\hat{o}^{(k+1)}(x') = \hat{o}^k(x') \frac{\int \frac{i(x)}{\hat{i}^k(x)} s(x, x')\, dx}{\int s(x, x')\, dx}$$

Where $\hat{o}$ is and estimation of the un-blurred image, i is the original image, including noise, and $\hat{i}$ is given by

$$\hat{i}^k(x) = \int \hat{o}^k(x') s(x, x')\, dx'$$

The issue of image restoration may also be viewed as a minimization problem, in which the ultimate goal is to generate a positive, smooth solution. Ultimately this results in the Lucy-Richardson equation:

$$\sum \left| i(x) - \hat{o}(x) \otimes s(x) \right|^2 + \beta \int \left| \hat{o}(x) \right|^2 dx$$

β is some wighting factor that is chosen empirically.

2.3 Limited PSF De-blurring

When the amount of blur is not known at all (as in everyday pictures), we cannot use the previous algorithms as they all require prior knowledge of the point spread function that was used to bur it. Therefore, this time we used another deconvolution function called DECONVBLIND**.** Its main inputs were the original blurred image, an initial guess of the PSF, and the number of iterations it should execute its filter for.

The deconvblind algorithm was used it in two different ways. In one, it generated an initial PSF of a chosen value for LEN and THETA (the direction and angle parameters of the blur), and then get size of that PSF for the initial PSF guess, but with all elements set to one. In the other, the size of the PSF is simply guessed at, again, with all elements set equal to one.

Additionally, two optional arguments could be added to "deconvblind" to get a deblurred image of a greater quality. The one we used extensively is called WEIGHT. We used that option with the "edge" function to find the edges in the original picture: the function "edge" assigns a value of one when the function finds edges, and a value of zero elsewhere. Therefore, when the deconvolution function is applied, it can distinguish different objects on the picture and will exclude (assign a zero value) to all the "bad" pixels (i.e. the blur). It is also possible to assign a threshold for the good pixels to decide which ones should be enhanced or dimmed.

Another option that we tried is called DAMPAR. It compared the original image with the deblurred image every time it iterates. When the difference between the same pixels in the two images gets below a predefined threshold, the iteration stops for that pixel so that it is attenuated. The consequence on the final deblurred image is the suppression of a lot of noise, preserving the details of the image everywhere else as the iteration process continues for the other pixels.

The accuracy of this technique depends heavily on the quality of the initial guess of the PSF. If the initial guess for the PSF is very different from the real value and the number of iterations is too small, the image will not be deblurred. In fact, poor guess can even make the "restored" image look even worse than the original image.

2.4 Deblurring with no PSF Information

The above technique also has limitations as far as real life applications are concerned. Since it takes advantage of some knowledge of the PSF used for blurring (the PSF's size), it is inappropriate for use when no information regarding the original PSF is known. This is not a very useful technique if we want to make a program that can be used by anyone to deblur an arbitrary image, as they would have to have information about this PSF that they probably just don't know. Therefore, we found a way to build an algorithm that could systematically deblur images without having to change parameters when changing the image to deblur. We found that instead of having to guess an initial PSF, we could only tell the size of the matrix we want the PSF to be, then letting the algorithm iterate to find the best values. This size turned out to be a 15x15 PSF matrix because it fits with almost any kind of picture and gives very good results without taking too much time. All the optional inputs that were discussed in the previous section were used with this technique too.

An additional challenge to successful deblurring was that sometimes deblurring would cause some ringing to appear on the deblurred image. This ringing effect is caused

by the high frequency drop-off: the deconvolution functions uses discrete Fourier transforms, which assumes periodic frequency pattern of an image. Therefore, there is high frequency drop-off where edges are. The function that can be used to solve that problem is called EDGETAPER. We used edgetaper before applying the deconvolution, and it blurred the edges of the original input image slighty. Since the edges were then less sharp the ringing effect was reduced.

It was pretty difficult to get good results with that technique, as it was difficult to find values for weight threshold, number of iterations, and PSF matrix size that could fit any picture. After much experimentation, it turned out that the weight threshold should be set between .10 and .25, the PSF matrix size should be set to 15x15, and the number of iterations should be any number more than 30.[2]

3. Results

3.1 Qualitative Analysis[3]

Two main approaches were used to evaluate the results of the aforementioned procedures. The first was a simple qualitative measure of blur removal. A known amount of blur, but no noise, was added to an image, and then the image was filtered to remove this known amount of blur using Wiener, regularized and Lucy-Richardson deblurring methods. The regularized and Wiener techniques produced what appeared to be the best results. They were largely able to restore the image to its original form, although it was grainier. This grainy effect was especially prevalent in regions that had been low in contrast prior to the initial blurring. We believe this to be due to the fact that in low contrast regions the blur factor smeared relatively similar tones into fewer, indistinguishable ones that could not be perfectly restored. It was surprising that the Lucy-Richardson method produced the worst results in this instance, as it is a nonlinear technique, and supposedly more advanced. However, after Gaussian noise was added to the image in addition to blur, the Lucy-Richardson algorithm actually performed the best. This context help make sense of the previous problem because when just blur is added, only a linear modification is being made and so the linear Wiener restoration technique should work the best. Introducing Gaussian noise, and thus a degree of spatial nonlinearity, caused the nonlinear Lucy-Richardson method to produce the best results.

As information about the PSF that was used to perform the blurring was removed from the algorithms, the efficacy of blur removal dropped. In the blind deblurring method, the majority of the blur itself was removed, but a lot of the image's original detail was lost and a "ringing" effect could be seen across the entire image. The ringing was a result of using a PSF designed to remove blur in areas of high contrast (edges, where blur should be most prominent) over the entire image, and thus creating high contrast in waves across the image. The image quality was vastly improved when the edge checking function was implemented so that only true edges would receive this edge deblurring treatment.

3.2 Quantitative Analysis[4]

While qualitative analysis is a good first step for determining whether an image processing technique has succeeded or not, quantification of the results is necessary. Quantification allows for a more exact measurement of improvement, and more importantly, allows for comparison between the efficacies of different methods. In this instance, quantification was performed by finding the mean pixel intensity value over a

[2] See Appendix 5.1 program 4 for more details
[3] See Appendix 5.3 for images used
[4] See Appendix 5.2 for result tables

region, the standard deviation of the pixel values over that same region, and then determining the image contrast ratio; the ratio of standard deviation to the mean. This ratio normalizes the standard deviation so that any changes to the mean intensity caused by our filtering technique would not influence determination of filter quality, and acts as a direct measure of image contrast. Contrast ratio for a restored image should be higher than that of the original blurred image. The reason for this is that blurring an image causes the pixels surrounding a moving edge, the area where motion blur occurs, to become washed out and all take on similar intensity values. This leads to a low standard deviation relative to the mean. Once the image is filtered, however, contrast between the edge and the object should be restored and this contrast will cause a higher value for the standard deviation to mean ratio. Comparisons between blurred and filtered images were made using a parameter of percent improvement[5].

According to Table 1, the technique we developed for blind filtering increased the contrast ratio across each of the images that it was used on[6] by a little more than 1% each. While this does suggest that the overall contrast of each image was improved, it is not the most meaningful way to consider the data. Since blur does not occur uniformly across an entire image, but rather most significantly along moving edges, the most appropriate way to measure the success of our method is to find the contrast ratio across a severely blurred region. This was accomplished by isolating a highly blurred subset of the original image, and comparing it to the same subset in the filtered image. Table 2 shows that over regions of high blur, the contrast ratio for the filtered image is ten to twenty percent greater than in the original blurred image – a significant increase. The high contrast ratio came from an image (Fish) with computer added blur while the lower contrast ratios were found in images (Woman and Train) with natural motion blur due to movement during image acquisition exposure time. Because the computer generated blur was more severe than natural blur, it makes sense that a greater degree of contrast restoration occurred in the image that was blurred computationally.

Finally, comparisons were drawn between the regularized, Lucy-Richardson, and Wiener filtering methods for an image with computationally generated blur under both no noise added, and Gaussian noise added conditions. Table 3 confirms the qualitative observations that were made, demonstrating that Weiner and regularized techniques show almost twice as much improvement over the blurred image than Lucy-Richardson does. The results shown in Table 4, the efficacy of these techniques when noise is added, appear odd at first glance. The extremely high values for Wiener and regularized techniques are actually a result of their inability to filter out the noise across the region of interest. This makes the contrast appear much higher than it should be for a "well processed" image. The lower, but still reasonable, contrast value found in the Lucy-Richardson method in this instance actually represents that method's superiority for instances in which noise is added.

4. Conclusions

Through this project, several techniques for frequency domain image processing were explored. In the simplest of these, motion blur was added to a deblurred image. In the most advanced, blur was filtered out of a partially blurred image when no information regarding the blurring PSF was known. This was accomplished by optimizing an edge detection algorithm, finding how to set appropriate thresholds for restoring blurred out

[5] Percent Difference = (Filtered Contrast Ratio – Blurred Contrast Ratio)/Blurred Contrast Ratio * 100%

[6] See Appendix 5.3 for these images

PSFs, and discerning how many filtering iterations were necessary to remove the blur. Ultimately improvements on the order of ten to twenty percent were obtained.

Several standard blur filtering techniques were also compared. If the only blurring added was linear, then the linear Wiener and regularized techniques were the most efficient. When non-linear noise was added, then the non-linear Lucy-Richardson technique was proven best.

There were minor shortcomings, however. Some of the MatLab functions utilized required that the images passed to them as two dimensional arrays. Since images are stored as an JxKx3 matrix, where J and K are the image size and "3" is the number of color layers in the image (red, blue, green), images had to be black and white in order to be used by those functions. This posed a problem at first and our initial solution converted all three layers to black and white which had the side effect of displaying our image three times. We were able to solve this problem through the MatLab function "RGB2Gray," which converts the image to one that is gray scaled. Discovering how to maintain the color information of these images would be a good improvement to this project.

Blur analysis also has valuable uses in modern optics. A technique called laser speckle analysis[7] is performed by shining a coherent laser beam onto a surface with dynamic flow, and the light reflected from this surface is imaged. The flow causes motion blur in the acquired image, and the amount of this blur is proportional to the flow rate. Analyzing the blur allows conclusions to be drawn about relative flow rates on the surface. In biomedical applications this leads to identification of veins, arteries, capillaries and occlusions in vasculature. Blur filtering is also important in security applications, where blur must be removed from facial features or objects in order to recognize a subject.[8]

All members of this team contributed substantially to the success of this project. Joseph Simmons explored the theory behind the deblurring techniques used and helped teach the rest of us the concepts behind frequency domain transformations, and the deblurring algorithms used. Benjamin Topper optimized the computational algorithms, and obtained the data used for quantitative analysis. Avi Wolfson developed the first draft of many of the computational algorithms, and did the majority of the writing. We worked on this project over the course of one month, and believe that we have increased our competency in the subject of image blur filtering accordingly.

---

[7] Ayata et al. *Laser Speckle Flowmetry for the Study of Cerebrovascular Physiology in Normal and Ischemic Mouse Cortex.* Cerebral Blood Flow Metabolism. 2004;24(7):744-755

[8] Podilchuk, C. *Preprocessing for Enhanced Face Recognition.* Accessed from http://www.caip.rutgers.edu/wiselab

5. Appendix

5.1 Programs & Code Segments

Note: All programs utilized for this project were generated in MatLab

Program 1: Blurring Function

```
function [Blurred PSF] = AddBlur(I)

figure; imshow(I); title('Original Image');%displays original image
LEN = 31; %the length of the motion blur
THETA = 11; %the angle of the motion blur
PSF = fspecial('motion',LEN,THETA); % create PSF
Blurred = imfilter(I,PSF,'circular','conv'); %convolves the input image
with the PSF
figure; imshow(Blurred); title('Blurred Image'); %displays blurred image
figure; imshow(PSF, [], 'InitialMagnification', 'fit') ; %displays PSF
```

Program 2 : Blurring Function with noise

```
function [Blurred PSF] = BlurNoise(I)

figure; imshow(I); title('Original Image');%displays original image
LEN = 31; %the length of the motion blur
THETA = 11; %the angle of the motion blur
PSF = fspecial('motion',LEN,THETA); % creates the PSF
Blurred = imfilter(I,PSF,'circular','conv'); %convolves the input image
with the PSF
Blurred = imnoise(Blurred, 'gaussian') ; %adds the noise
figure; imshow(Blurred); title('Blurred Image'); %displays blurred image
figure; imshow(PSF, [], 'InitialMagnification', 'fit') ; %displays PSF
```

Program 3 : DeBlurring using Wiener, regularized and Lucy-Richardson algorithms

Note : Input image should already be blurred

```
function [wnr reg lucy] = AviDeBlur(I, PSF)

%Wnr Method%
wnr = deconvwnr(I,PSF); % Weiner filtering function
figure;imshow(wnr); %displays deblurred image
title('Wiener Restore');

%Regularized Method%
reg = deconvreg(I, PSF) ; % regularized filter
figure; imshow(reg) ; %displays deblurred image
title('Regularized Restore') ;

%Lucy-Richardson method%
lucy = deconvlucy(I,PSF); %Lucy-Richardson filtering function
figure;imshow(lucy); %displays deblurred image
title('Lucy-Richardson, True PSF')
```

Program 4 : DeBlurring with no PSF information

Note : Input image is already blurred

```
function [J2 P2] = AviBlindDeBlur(I)

K=rgb2gray(I); %makes it a black&white image
figure; imshow(K); title('Original Image'); %display original image

PSF = fspecial('Motion',5,45);  %
INITPSF = ones(size(PSF)); %if you want to guess the PSF, include these
2 last lines. The first line allows you to guess the PSF, second line
creates a matrix the same size as the one you guessed with 1s elements.

INITPSF=ones(15:15); %guesses the size of the PSF
[J P]= deconvblind(K,INITPSF,150); %tries to deblur - 150 iterations
figure; imshow(J); title('Restored Image'); %displays the "restored"
image

WEIGHT = edge(K(:,:),'sobel',.10); %create high contrast detecting
weight array i.e. it finds the edges of objects/people on the image
which will help to refine guess on PSF - sobel method worked really
good and seemed to be the fastest. Best threshold for common pictures
seems to be between 0.10 and .25

se = strel('disk' , 2);
WEIGHT = 1- double(imdilate(WEIGHT, se));
WEIGHT([1:3 end-[0:2]], :) = 0; WEIGHT(:, [1:3 end-[0:2]]) = 0; %create
a morphological structuring element, here a disk of radius 2. Octagon
worked fine too but disk seems more commonly used

P1 = P; %
P1((P1 < 0.001))=0; %Refines the guess for the PSF by taking into
account the weight array.
[J2 P2] = deconvblind(K(:,:),P1,6,[],WEIGHT); %constructs new PSF and
deblurred image w/ weight array

figure; imshow(J2); %displays "new restored" image
title('Newly Deblurred Image');
```

Program 5 : Quantifying the results

Note : Inputs are blurred image ,filtered image, and size of area of interest(AoI)

```
function Quantify(Blurred, Filtered, x1, x2, y1, y2)

figure; imshow(Blurred); title('Blurred'); %blurred image
figure; imshow(Filtered); title('Filtered'); %filtered image
Blurred = Blurred(x1:x2, y1:y2); %picks area of interest
Filtered = Filtered(x1:x2, y1:y2); % picks same AoI (to be able to
compare them)
figure; imshow(Blurred); title('Blurred Segment'); %shows AoI on
blurred image
figure; imshow(Filtered); title('Filtered Segment'); %shows AoI on
filtered image

BlurredAvg = sum(sum(Blurred))/numel(Blurred) %mean pixel intensity for
blurred image
Blurred = (Blurred - BlurredAvg).^2 ; %difference between blurred pixel
intensity and mean pixel intensity
```

```
BlurredStdDev=sqrt((1/(numel(Blurred)-1))*sum(sum(Blurred))) %Standard
Deviation
BlurredRatio = BlurredStdDev/BlurredAvg %Ratio StdDeviation/Mean

FilteredAvg = sum(sum(Filtered))/numel(Filtered) %mean pixel intensity
for filtered image
Filtered = (Filtered - FilteredAvg).^2 ; %difference between blurred
pixel intensity and mean pixel intensity
FilteredStdDev = sqrt((1/(numel(Filtered)-1))*sum(sum(Filtered)))
%Standard Deviation
FilteredRatio = FilteredStdDev/FilteredAvg %Ratio StdDeviation/Mean
```

5.2 Quantitative Analysis

Pixel Contrast Measurement for Blurred and Filtered Full Images

Image	Blurred Ratio	Filtered Ratio	Improvement (%)
Woman	0.0975	0.0986	1.13%
Train	0.0737	0.0746	1.22%
Fish	0.1027	0.1044	1.66%

Table 1: Deblurring Efficacy for Three Full Images

Pixel Contrast Measurement for Blurred and Filtered Image Segments

Image	Blurred Ratio	Filtered Ratio	Improvement (%)
Woman Segment	0.0323	0.0370	14.55%
Train Segment	0.0520	0.0573	10.19%
Fish Segment	0.0517	0.0630	21.86%

Table 2: Deblurring Efficacy for Three Image Segments

Pixel Contrast Measurement for Various Techniques (No Noise Added)

Technique	Blurred Ratio	Filtered Ratio	Improvement (%)
Wiener	0.0517	0.0606	17.21%
Regularized	0.0517	0.0606	17.21%
Lucy-Richardson	0.0517	0.0563	8.90 %

Table 3: Deblurring Efficacy for Three Filtering Techniques

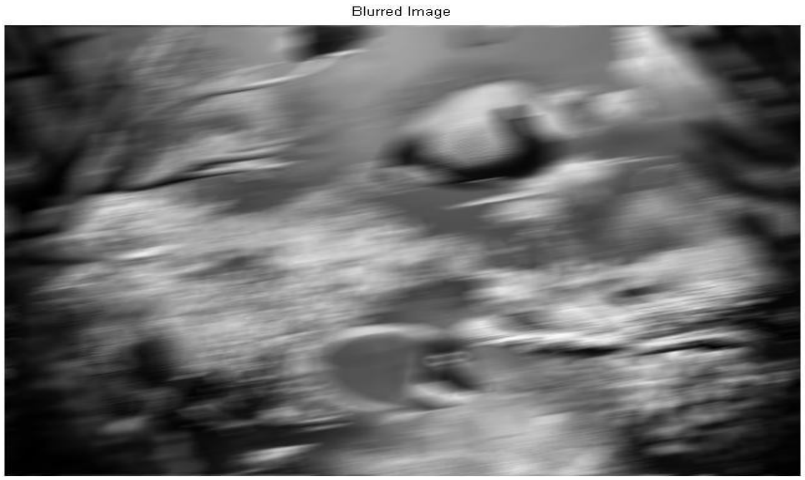
Pixel Contrast Measurement for Various Techniques (Noise Added)

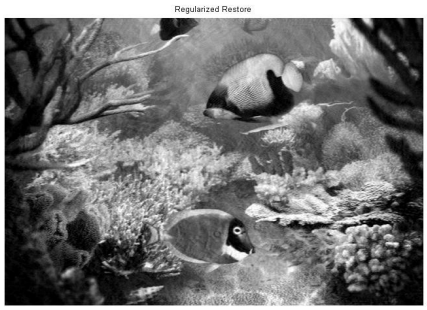
Technique	Blurred Ratio	Filtered Ratio	Improvement (%)
Wiener	0.0507	0.0692	36.49%
Regularized	0.0507	0.0691	36.29%
Lucy-Richardson	0.0507	0.0543	7.10%

Table 4: Deblurring Efficacy for Three Filtering Techniques with Added Noise

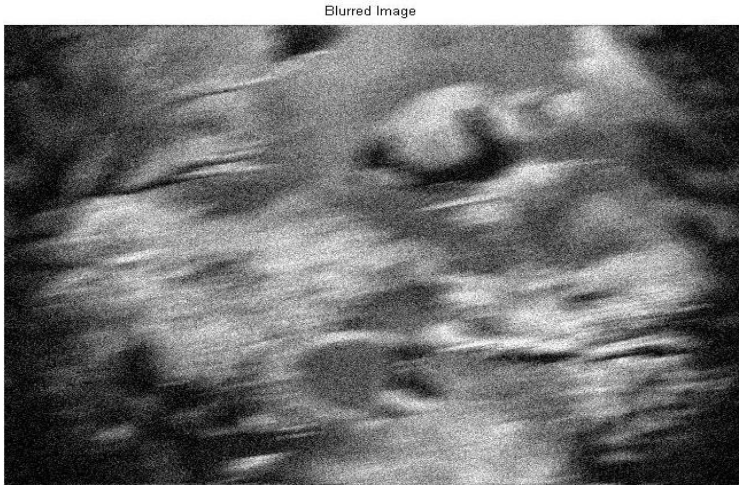
5.3 Images Used

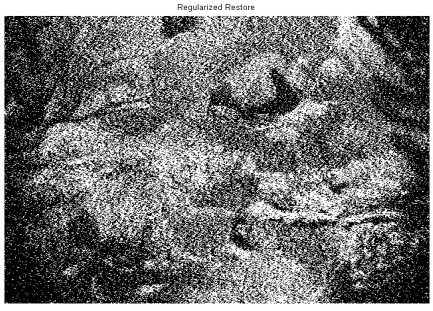
5.3a : Blurred Fish image - no noise



 Wiener filter	The blur function applied is linear therefore the Wiener filter (which is linear) turns out to be the best algorithm to deblut the image.
 Regularized filter	The regularized was a little worst than the Wiener filter to unblur the image.
 Lucy Richardson	The Lucy-Richardson turned out to be the worst for a simple linear blur, even though the image was ok.

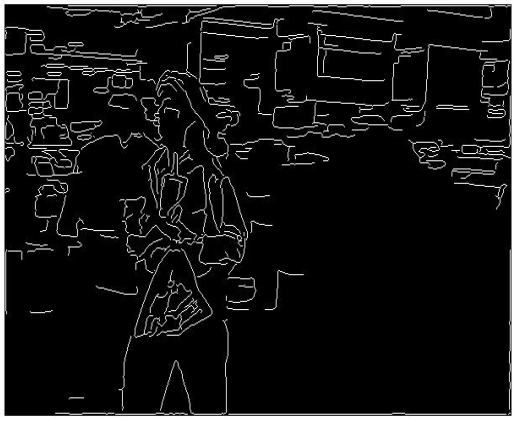
5.3b : Blurred Fish image - with gaussian noise



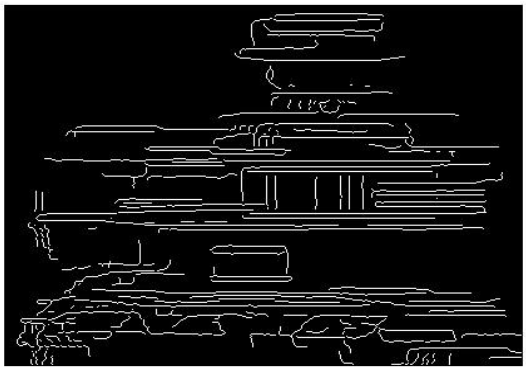
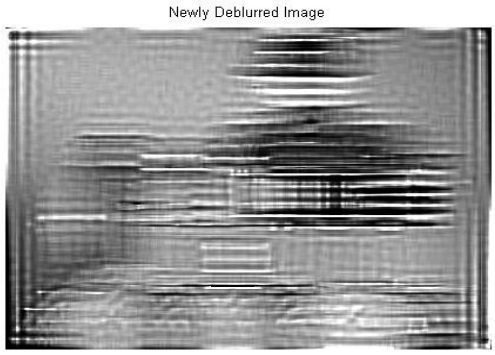
 Wiener filter	When gaussian noise is added to the blur, Wiener filter gave the worst result.
 Regularized filter	Regularized filter was a little better than Wiener but still it was a poor quality image.
 Lucy Richardson algorithm	The Lucy Richardson gave a very good result -much better than the two other filters. *Note : Printing degrades quality of image, the higher resolution image is a better quality.*

5.3c : Deconvlind - deblurring images with no information on the PSF

Woman

	Original blurred image
	Weight image : recognition of objects and people on the picture.
Newly Deblurred Image 	Final unblurred image - some ringing effect is visible, but image is globally unblurred.

Train

	Original blurred image.
	Weight image : recognition of the little train with the guy on it.
	Final unblurred image - with so much intial blur, i think this is the best we can get !