ISSN (Online) 2394-2320
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Special Issue
National Conference on "Recent Trends, Advancement and Applications of Digital Image Processing" (NCDIP 2016)

Image Deblurring: A Matlab Based Approach Using Algorithms and Filters

[1]Dr.S. Omkumar, [2]M. Sivakumar, [3]Karthiga Mohan
[1]Associate Professor, [2]Research Scholar, [3]UG Student,
Department of ECE, SCSVMV University, Kanchipuram

Abstract:- **This paper shows how to use blind deconvolution to deblur images. The algorithm of blind deconvolution can be employed efficiently if no information about the noise and blurring is obtained. This algorithm retains the picture or image and the point-spread function (PSF) at the same time. In each iteration, the accelerated Richardson-Lucy algorithm is applied. The characteristics of additional optical system like camera can be employed as input parameters to enhance the quality of the image restoration. PSF constraints can be passed in through a user-specified function. The concept of deconvolution can be applied efficiently when constraints are applied on the recovered image and limited information is obtained about the additive noise. The noisy and blurred noisy image is retained by a least square restoration algorithm that employs a regularized filter. Wiener deconvolution can be useful when the point-spread function and noise level are either known or estimated.The simulation process is carried out by Matlab2015R to check the functionality.**

Keywords: **PSF, NSR, Binned Image, Quantized Image, Regularized Filter, Blind Deconvolution algorithm, least square restoration algorithm.**

I. INTRODUCTION

To deblur an image, we proposed Blind Deconvolution algorithm, Lucy-Richardson algorithm and Regularized deconvolution, Wiener deconvolution can be used effectively. The algorithms are maximizes the likelihood that the resulting image, when convolved with the resulting PSF, is an instance of the blurred image, assuming Poisson noise statistics. The blind deconvolution algorithm can be used effectively when no information about the distortion (blurring and noise) is known. The deconvblind function restores the image and the PSF simultaneously, using an iterative process similar to the accelerated, damped Lucy-

Richardson algorithm.

The deconvblind function, just like the deconvlucy function, implements several adaptations to the original Lucy-Richardson maximum likelihood algorithm that address complex image restoration tasks. Using these adaptations, you can

- ♦ Reduce the effect of noise on the restoration
- ♦ Account for nonuniform image quality (e.g., bad pixels)
- ♦ Handle camera read-out noise

Causes of Blurring

The blurring, or degradation, of an image can be caused by many factors:

- ♦ Movement during the image capture process, by the camera or, when long exposure times are used, by the subject.
- ♦ Out-of-focus optics, use of a wide-angle lens, atmospheric turbulence, or a short exposure time, which reduces the number of photons captured
- ♦ Scatteredlightdistortioninconfocal Microscopy.

Deblurring Model

A blurred or degraded image can be approximately described by this equation **g = Hf + n**, where
g = The blurred image
H= The distortion operator, also called the point spread function (PSF)
F= The original true image
n= Additive noise, introduced during image acquisition, that corrupts the image.

Importance of the PSF

Based on this model, the fundamental task of deblurring is to deconvolve the blurred image with the PSF that exactly describes the distortion. Deconvolution is the process of reversing the effect of convolution. The quality of the deblurred image is mainly determined by knowledge of the PSF.

ISSN (Online) 2394-2320
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Special Issue
National Conference on "Recent Trends, Advancement and Applications of Digital Image Processing" (NCDIP 2016)

II. ALGORITHM BASED APPROACH

A. *Deblurring with the Blind Deconvolution Algorithm*

Step 1: Read the input Image.
Step 2: Simulate a Blur model.
Step 3: Restore the Blurred Image Using PSFs of Various Sizes.
Step 4: Analyzing the Restored PSF.
Step 5: Improving the Restoration.
Step 6: Using Additional Constraints on the PSF Restoration.

The algorithm maximizes the likelihood that the resulting image, when convolved with the resulting PSF, is an instance of the blurred image, assuming Poisson noise statistics. The blind deconvolution algorithm can be used effectively when no information about the distortion (blurring and noise) is known. The deconvblind function restores the image and the PSF simultaneously, using an iterative process similar to the accelerated, damped Lucy-Richardson algorithm.

The deconvblind function, just like the deconvlucy function, implements several adaptations to the original Lucy-Richardson maximum likelihood algorithm that address complex image restoration tasks. Using these adaptations, you can

- Reduce the effect of noise on the restoration
- Account for nonuniform image quality (e.g., bad pixels)
- Handle camera read-out noise

Step 1: Read Image

The example reads in an intensity image. The deconvblind function can handle arrays of any dimension.

Step 2: Simulate a Blur

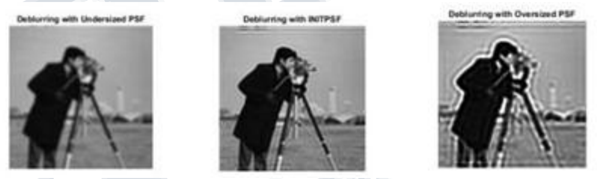
Simulate a real-life image that could be blurred (e.g., due to camera motion or lack of focus). The example simulates the blur by convolving a Gaussian filter with the true image (usingimfilter). The Gaussian filter then represents a point-spread function, PSF.

Step 3: Restore the Blurred Image Using PSFs of Various Sizes

To illustrate the importance of knowing the size of the true PSF, this example performs three restorations. Each time the PSF reconstruction starts from a uniform array--an array of ones.

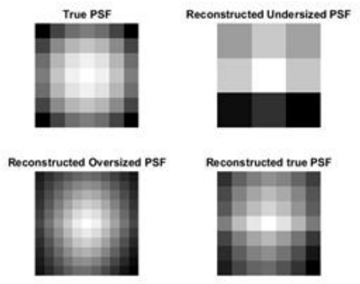
The first restoration, J1 and P1, uses an undersized array, UNDERPSF, for an initial guess of the PSF. The size of the UNDERPSF array is 4 pixels shorter in each dimension than the true PSF.
The second restoration, J2 and P2, uses an array of ones, OVERPSF, for an initial PSF that is 4 pixels longer in each dimension than the true PSF.

The third restoration, J3 and P3, uses an array of ones, INITPSF, for an initial PSF that is exactly of the same size as the true PSF.



Step 4: Analyzing the Restored PSF

All three restorations also produce a PSF. The following pictures show how the analysis of the reconstructed PSF might help in guessing the right size for the initial PSF. In the true PSF, a Gaussian filter, the maximum values are at the center (white) and diminish at the borders (black).



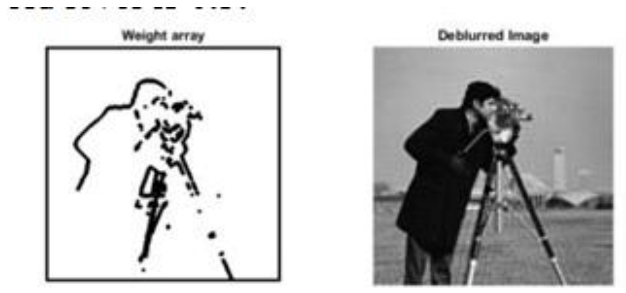
The PSF reconstructed in the first restoration, P1, obviously does not fit into the constrained size. It has a strong signal variation at the borders. The corresponding image, J1, does not show any improved clarity vs. the blurred image, Blurred.

The PSF reconstructed in the second restoration, P2, becomes very smooth at the edges. This implies that the restoration can handle a PSF of a smaller size. The corresponding image,J2, shows some deblurring but it is strongly corrupted by the ringing.

Finally, the PSF reconstructed in the third restoration, P3, is somewhat intermediate between P1 and P2. The array, P3, resembles the true PSF very well. The corresponding image, J3, shows significant improvement; however it is still corrupted by the ringing.

Step 5: Improving the Restoration

The ringing in the restored image, J3, occurs along the areas of sharp intensity contrast in the image and along the image borders. This example shows how to reduce the ringing effect by specifying a weighting function. The algorithm weights each pixel according to the WEIGHT array while restoring the image and the PSF. In our example, we start by finding the "sharp" pixels using the edge function. By trial and error, we determine that a desirable threshold level is 0.3.



The image is restored by calling deconvblind with the WEIGHT array and an increased number of iterations (30). Almost all the ringing is suppressed.[7]

Step 6: Using Additional Constraints on the PSF Restoration

The results shows how you can specify additional constraints on the PSF. The function, FUN, below returns a modified PSF array which deconvblind uses for the next iteration.

In this approach, FUN modifies the PSF by cropping it by P1 and P2 number of pixels in each dimension, and then padding the array back to its original size with zeros. This operation does not change the values in the center of the PSF, but effectively reduces the PSF size by 2*P1 and 2*P2 pixels.

B. Deblurring Images Using the Lucy-Richardson Algorithm

Step 1: Read the input Image
Step 2: Simulate a Blur and Noise
Step 3: Restore the Blurred and Noisy Image
Step 4: Iterate to Explore the Restoration
Step 5: Control Noise Amplification by Damping
Step 6: Create Sample Image
Step 7: Simulate a Blur
Step 8: Provide the WEIGHT Array
Step 9: Provide a finer-sampled PSF

Lucy – Richardson Algorithm can be used effectively when the point-spread function PSF (blurring operator) is known, but little or no information is available for the noise. The blurred and noisy image is restored by the iterative, accelerated, damped Lucy-Richardson algorithm. The additional optical system (e.g. camera) characteristics can be used as input parameters to improve the quality of the image restoration.[1]

Step 1: Read Image

The example reads in an RGB image and crops it to be 256-by-256-by-3. The deconvlucy function can handle arrays of any dimension.[2]

Step 2: Simulate a Blur and Noise

Simulate a real-life image that could be blurred (e.g., due to camera motion or lack of focus) and noisy (e.g., due to random disturbances). The example simulates the blur by convolving a Gaussian filter with the true image (using imfilter). The Gaussian filter then represents a point-spread function, PSF.

The example simulates the noise by adding a Gaussian noise of variance V to the blurred image (using imnoise). The noise variance V is used later to define a damping parameter of the algorithm.

Step 3: Restore the Blurred and Noisy Image

Restore the blurred and noisy image providing the PSF and using only 5 iterations (default is 10). The output is an array of the same type as the input image.

ISSN (Online) 2394-2320
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Special Issue
National Conference on "Recent Trends, Advancement and Applications of Digital Image Processing" (NCDIP 2016)

Step 4: Iterate to Explore the Restoration

The resulting image changes with each iteration. To investigate the evolution of the image restoration, you can do the deconvolution in steps: do a set of iterations, see the result, and then resume the iterations from where they were stopped. To do so, the input image has to be passed as a part of a cell array (e.g., start first set of iterations by passing in {BlurredNoisy}instead of BlurredNoisy as input image parameter).

luc1_cell = deconvlucy({BlurredNoisy},PSF,5); In that case the output, luc1_cell, becomes a cell array. The cell output consists of four numeric arrays, where the first is the BlurredNoisy image, the second is the restored image of class double, the third array is the result of the one-before-last iteration, and the fourth array is an internal parameter of the iterated set. The second numeric array of the output cell-array, image luc1_cell{2}, is identical to the output array of the Step 3, image luc1, with a possible exception of their class (the cell output always gives the restored image of class double).

To resume the iterations, take the output from the previous function call, the cell-array luc1_cell, and pass it into the deconvlucy function. Use the default number of iterations (NUMIT = 10). The restored image is the result of a total of 15 iterations.

Step 5: Control Noise Amplification by Damping

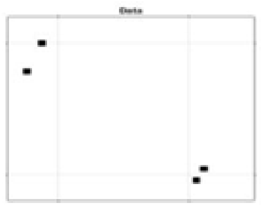
The latest image, luc2, is the result of 15 iterations. Although it is sharper than the earlier result from 5 iterations, the image develops a "speckled" appearance. The speckles do not correspond to any real structures (compare it to the true image), but instead are the result of fitting the noise in the data too closely.

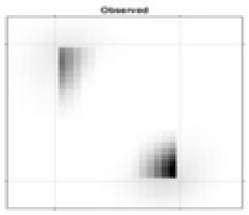
To control the noise amplification, use the damping option by specifying the DAMPAR parameter. DAMPAR has to be of the same class as the input image. The algorithm dampens changes in the model in regions where the differences are small compared with the noise. The DAMPAR used here equals 3 standard deviations of the noise. Notice that the image is smoother.

The next part of this example explores the WEIGHT and SUBSMPL input parameters of the deconvlucy function, using a simulated star image (for simplicity & speed).

Step 6: Create Sample Image

The example creates a black/white image of four stars.



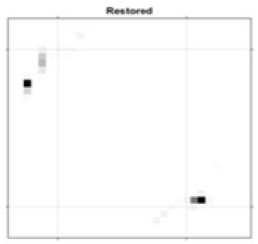


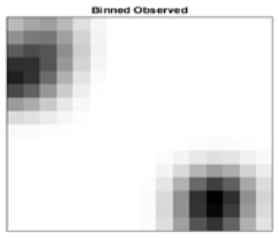
Step 7: Simulate a Blur

The example simulates a blur of the image of the stars by creating a Gaussian filter, PSF, and convolving it with the true image. Now simulate a camera that can only observe part of the stars' images (only the blur is seen). Create a weighting function array, WEIGHT, that consists of ones in the central part of the Blurred image ("good" pixels, located within the dashed lines) and zeros at the edges ("bad" pixels - those that do not receive the signal). To reduce the ringing associated with borders, apply the edgetaper function with the given PSF.

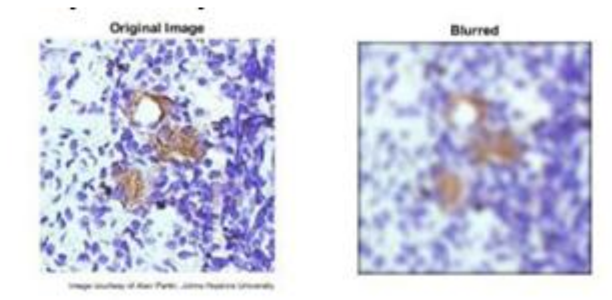
Step 8: Provide the WEIGHT Array

The algorithm weights each pixel value according to the WEIGHT array while restoring the image. In our example, only the values of the central pixels are used (where WEIGHT = 1), while the "bad" pixel values are excluded from the optimization. However, the algorithm can place the signal power into the location of these "bad" pixels, beyond the edge of the camera's view. Notice the accuracy of the resolved star positions.[3]





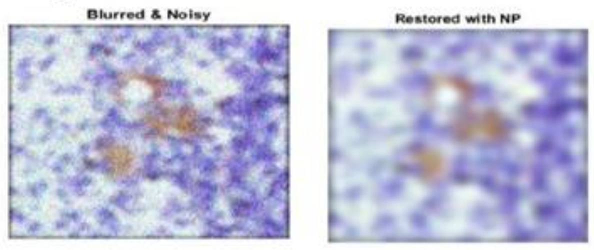
The example reads in an RGB image and crops it to be 256-by-256-by-3. The deconvreg function can handle arrays of any dimension.



Step 2: Simulate a Blur and Noise

Simulate a real-life image that could be blurred (e.g., due to camera motion or lack of focus) and noisy (e.g., due to random disturbances). The example simulates the blur by convolving a Gaussian filter with the true image (using imfilter). The Gaussian filter represents a point-spread function, PSF.

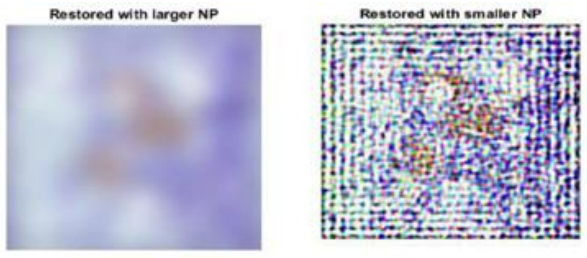
We simulate the noise by adding a Gaussian noise of variance V to the blurred image (using imnoise).



Step 3: Restore the Blurred and Noisy Image

Restore the blurred and noisy image supplying noise power, NP, as the third input parameter. To illustrate how sensitive the algorithm is to the value of noise power, NP, the example performs three restorations.

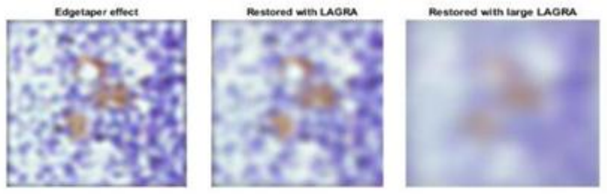
The first restoration, reg1, uses the true NP. Note that the example outputs two parameters here. The first return value, reg1, is the restored image. The second return value, LAGRA, is a scalar, Lagrange multiplier, on which the deconvreg has converged. The second restoration, reg2, uses a slightly over-estimated noise power, which leads to a poor resolution.[4]



The third restoration, reg3, is given an under-estimated NP value. This leads to an overwhelming noise amplification and "ringing" from the image borders.

Step 4: Reduce Noise Amplification and Ringing

Reduce the noise amplification and "ringing" along the boundary of the image by calling the edgetaper function prior to deconvolution. Note how the image restoration becomes less sensitive to the noise power parameter.



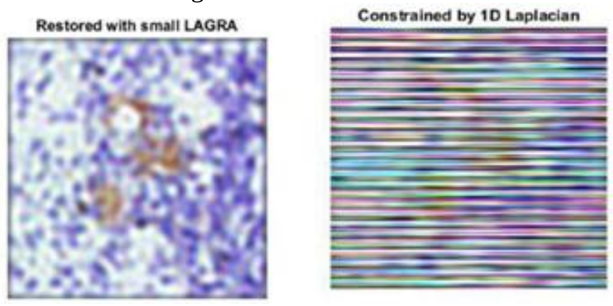
Step 5: Use the Lagrange Multiplier

Restore the blurred and noisy image, assuming that the optimal solution is already found and the corresponding Lagrange multiplier, LAGRA, is given. In this case, any value passed for noise power, NP, is ignored.

To illustrate how sensitive the algorithm is to the LAGRA value, the example performs three restorations. The first restoration (reg5) uses the LAGRA output from the earlier solution (LAGRA output from first solution in Step 3).

The second restoration (reg6) uses 100*LAGRA which increases the significance of the constraint. By default, this leads to over-smoothing of the image.

The third restoration uses LAGRA/100 which weakens the constraint (the smoothness requirement set for the image). It amplifies the noise and eventually leads to a pure inverse filtering for LAGRA = 0.



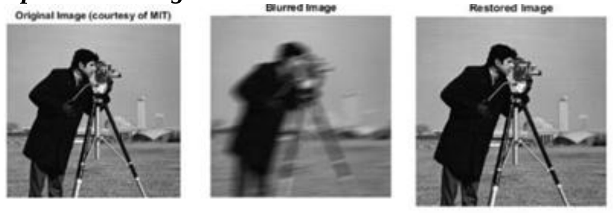
Step 6: Use a Different Constraint

Restore the blurred and noisy image using a different constraint (REGOP) in the search for the optimal solution. Instead of constraining the image smoothness (REGOP is Laplacian by default), constrain the image smoothness only in one dimension (1-D Laplacian).

D. Deblurring Images Using a Wiener Filter

Step1: Read Image
Step2: Simulate a Motion Blur
Step3: Restore the Blurred Image
Step4: Simulate Blur and Noise
Step5: Restore the Blurred and Noisy Image: First Attempt
Step6: Restore the Blurred and Noisy Image: Second Attempt
Step7: Simulate Blur and 8-Bit Quantization Noise
Step8: Restore the Blurred, Quantized Image: First Attempt
Step9: Restore the Blurred, Quantized Image: Second Attempt.

Step1: Read Image



Step2: Simulate a Motion Blur

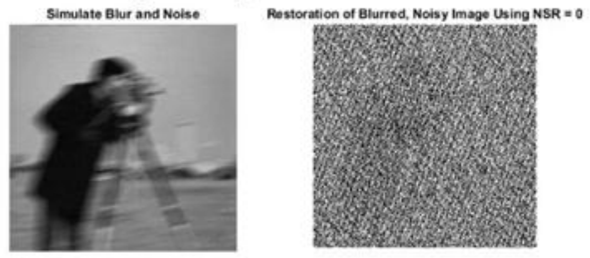
Simulate a blurred image that you might get from camera motion. Create a point-spread function, PSF, corresponding to the linear motion across 31 pixels (LEN=31), at an angle of 11 degrees (THETA=11). To simulate the blur, convolve the filter with the image using imfilter.

Step3: Restore the Blurred Image

The simplest syntax for deconvwnr is deconvwnr(A, PSF, NSR), where A is the blurred image, PSF is the point-spread function, and NSR is the noise-power-to-signal-power ratio. The blurred image formed in Step 2 has no noise, so we'll use 0 for NSR.[5]

Step4: Simulate Blur and Noise

Now let's try adding noise.



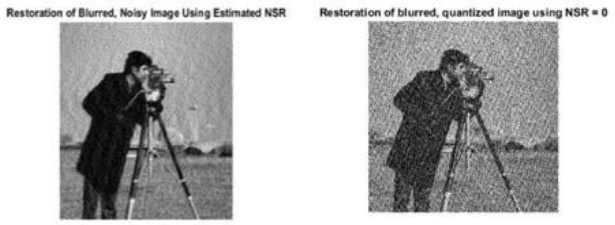
Step5: Restore the Blurred and Noisy Image: First Attempt.

In our first restoration attempt, we'll tell deconvwnr that there is no noise (NSR = 0). When NSR = 0, the Wiener restoration filter is equivalent to an ideal inverse filter. The ideal inverse filter can be extremely sensitive to noise in the input image, as the next image shows:

The noise was amplified by the inverse filter to such a degree that only the barest hint of the man's shape is visible.

Step6: Restore the Blurred and Noisy Image: Second Attempt

In our second attempt we supply an estimate of the noise-power-to-signal-power ratio.


Restoration of Blurred, Noisy Image Using Estimated NSR
Restoration of blurred, quantized image using NSR = 0

Step7: Simulate Blur and 8-Bit Quantization Noise

Even a visually imperceptible amount of noise can affect the result. Let's try keeping the input image in uint8 representation instead of converting it to double.

Step8: Restore the Blurred, Quantized Image: First Attempt

Again, we'll try first telling deconvwnr that there is no noise.

Step9: Restore the Blurred, Quantized Image: Second Attempt.

Restore the Blurred, Quantized Image: Second Attempt. Next, we supply an NSR estimate to deconvwnr.[6]

Restoration of Blurred, Quantized Image Using Computed NSR

III. CONCLUSION

This paper describes how to use blind deconvolution and Damped Richardson-Lucy Algorithm to deblur images. And also Regularized deconvolution and Wiener deconvolution can be used effectively when constraints are applied on the recovered image and limited information is known about the additive noise. The blurred and noisy image is restored by a constrained least square restoration algorithm that uses a regularized filter. Wiener deconvolution can be useful when the point-spread function and noise level are either known or estimated.The simulation results shows that the functionality of the above said deconvolution methods using algorithmic and filter based approaches.

REFERENCES

[1] Jian-Jiun Ding; Wei-De Chang; Yu Chen; Szu-Wei Fu; Chir-Weei Chang;Chuan-Chung Chang, "Image deblurring usinga pyramid-based Richardson-Lucy algorithm", IEEE,2014

[2] Hongbin Wang; Paul C. Miller, "Scaled Heavy-Ball Acceleration of the Richardson-Lucy Algorithm for 3D Microscopy Image Restoration", IEEE,2014

[3] Jiunn-Lin Wu; Chia-Feng Chang; Chun-Shih Chen."An improved Richardson-Lucy algorithm for single image deblurring using local extrema filtering" IEEE,2012.

[4] Elad Shaked; Sudipto Dolui; Oleg V. Michailovich, "Regularized Richardson-Lucy algorithm for reconstruction of Poissonian medical images" IEEE,2011

[5] Matteo Pedone; Eduardo Bayro-Corrochano; Jan Flusser; "Quaternion Wiener Deconvolution for Noise Robust Color Image Registration", IEEE,2015

[6] Sina Jafarpour; Ali Pezeshki; Robert Calderbank, "Experiments with Compressively Sampled Images and a New Debluring-Denoising Algorithm", IEEE 2008.

[7] Ming Zhao; Wei Zhang; Zhile Wang; Fugang Wang, "Spatially adaptive image deblurring based on nonlocal means", IEEE 2010.