BIL 415 - Image Processing Practicum

HACETTEPE UNIVERSITY
Department of Computer Engineering

Problem Set-2
Fall 2018-2019
Dr. Erkut Erdem
TA. Ozge Yalcinkaya

Due Date: 23:00 pm on Friday, November 23rd, 2018

Motion Deblurring in Frequency Domain

In this assignment, you are going to restore some sloppy images that have motion blur. We usually encounter with motion blur on images especially while doing long-exposure photography. There are many different methods to handle this situation but you are going to implement the most basic method, deconvolution [1], in frequency domain, where we know the motion blur kernel, usually called as Point Spread Function (PSF). In the final part, you will implement another straight forward method for deblurring, Richardson–Lucy deconvolution [2], in frequency domain. You will use only grayscale images in this assignment.

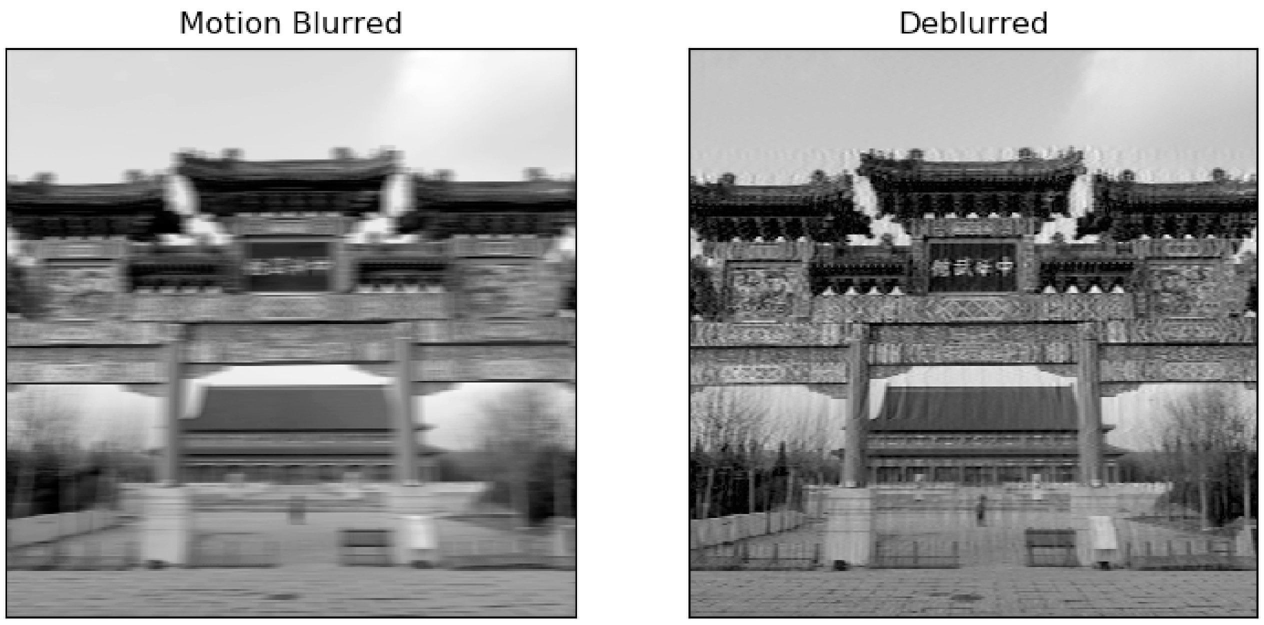

Figure 1: An example to deblurring.

PART 1: Exploring Deconvolution

Some motion blurred images and the original images are given in the folder "part1" from [3]. The relation between the blurred and the original image can be formulated as in the following equation:

$$B = I * h + \epsilon$$

where $*$ denotes convolution, I is the original image, h is blur kernel, also called point spread function, ϵ is some unknown noise and B is the blurred image. In frequency domain, convolution can be done by doing element-wise multiplication with the original image and the kernel. Therefore, in frequency domain, the deconvolution can be done by dividing B with a known blur kernel h, by discarding the noise. However, in this part, since you have the original image as well, you can find the kernel that is effected by the noise in frequency domain, simply by dividing blur image with the original image. After that, you can get the restored image by diving B with kernel and use inverse fft. Is there any difference with the original image?

Examine the Fourier Transforms of the given images and your findings such as the kernel by using Numpy's "fft2" function in magnitude spectrum (use fftshift). Make your comments on the magnitudes such as what is the direction and the size of the applied kernel. How do you come up with these ideas?

PART 2: Finding the Blur Kernel

In real life, we don't have the original image so we are not able to find the exact transformation from blur image to restored image. Generally, the methods are developed according to a known kernel (PSF) and try to minimize the effect of noise, since the noise is unknown. There are methods that implements "Blind Deconvolution" where the PSF is also unknown.

In this part, 3 motion blurred images are given in folder "part2". You are going to examine the magnitude spectrum of each image to guess the applied kernels. The orientation of the frequency components will give you the direction and the number of them will give you the size of the motion blur kernel. You may examine some known motion blur kernels to figure this out (*Hint: Two of them has the same size and the other one is smaller.*). Explain all your assumptions over the magnitude spectrum of the kernels.

After finding 3 different kernels, visualize them in spatial domain. Then, apply them to another image in the frequency domain for testing. You should do zero padding to the kernel by centering it in order to have the same size with the original image. After that, you can get fft of the image and the kernel and do element-wise multiplication to get the motion blurred image. Compare the magnitude spectrum of your motion blurred images with the ones in folder "part2". They should be similar, if you have found the correct kernels.

PART 3: Deblurring

a) Deconvolution

Now you have blurred images (folder "part2") and their kernels. Try to restore the original images by doing deconvolution in frequency domain like you did in Part 1. You are going to obtain a noisy image instead of a deblurred one. Try to think about the reason and give your comments. Remember that, we didn't count the noise. (*Hint: You are dividing noisy frequencies with very small values.*)

b) Deconvolution with Regularization

The first thing you can do to apply a regularization so that zero the frequencies in the indices that point spread function is below a threshold, e.g 0.1. That is:

$$I'(x,y)_{fft} = \begin{cases} B(x,y)_{fft}/h(x,y)_{fft} & \text{if } abs(h(x,y)_{fft}) > threshold \\ 0 & \text{otherwise} \end{cases}$$

where B is the blurred image, h is the PSF and I' is the restored image. You can use Numpy's "where" function. An example result can be seen in Figure 1. Do experiments with different thresholds and explain your findings. (*Note: To avoid division by zero, you may add a very small value to $h(x,y)_{fft}$ such as 10^{-6}.*)

c) BONUS

Try to implement your own idea to deblur the image. You can obtain better or worse results, just compare them with Part 3.b. and give your comments why it is better or not. Give all method details, you may use built-in functions, but you will get the points accordingly. E.g. using some built-in deblur function from Scipy or some other library does not give you significant points for this part. Explore different debluring methods in frequency domain. You may also adapt your implementation from Part 4. You may check [4].

PART 4: Richardson–Lucy

In this final part, you will implement a basic iterative formulation to deblur an image. You will use the given starter code and implement the function by using frequency domain convolution. In this code, the original image, PSF, the blurred image and noise is already given. You will just implement the below iterative formulation by using the given kernel and the blurred image, with frequency domain convolution:

$$I'^{(t+1)} = I'^{t}.\frac{B}{I'^{t} * h} * h$$

where I' is the restored image over iteration t, B is the blured image, h is the PSF and $*$ is spatial convolution. You can initialize I'^0 as an image in the same size with the blurred image and filled with 0.5. What does this algorithm do? Explain in detail. Give intermediate step results if it is necessary.

The Implementation Details

1. You are not allowed to use some built-in functions such as "fftconvolve, richardson_lucy" from Scipy. Other than that, try to learn how to be more efficient by using the functions of Numpy, Scipy etc. You will basically use "fft2, ifft2, fftshift, ifftshift".
2. You are expected to write functions for each operation and give a main script to run all results. You can create classes for different parts that include the functions and you can import your classes to your main script.
3. You should pay attention to code readability such as comments, function/variable names and your code quality: 1) no hard-coding 2) no repeated code 3) cleanly separate and organize your code 4) use consistent style, indentation 5) write deterministic algorithms
4. Your code should save resulting images in different folders under "results/" as"part1, part2 part3a, part3b, part3c, part4".
5. Implement your code with Python 3 and use libraries from Anaconda. You can install any library that is not in Anaconda as well, such as OpenCV 3.

What should you write in the report?

- Give your resulting images step by step for each part.
- Give experimental results, used parameters and comments on the results in detail.
- Explain your implementation and code as well as your approach to the problem.
- A basic structure might be: 1) Introduction (what is the problem, how do you approach to this problem, what is the content of your report) 2) Implementation Details (the method you followed, the organization of your code, functions and details of your solution) 3) Experimental Results (all results for separate parts with different parameters and your comments on the results) 4) Conclusion (what are the results and what are the weaknesses of your implementation, in which parts you have failed and why, possible future solutions)
- You should write your report in LaTeX

Grading

The assignment will be graded out of 100:

- **20 (part 1):** CODE: 0 (no implementation), 5 (a partially correct solution), 15 (a correct solution) and REPORT: 5
- **20 (part 2):** CODE: 0 (no implementation), 5 (a partially correct solution), 15 (a correct solution) and REPORT: 5
- **30 (part 3):** CODE: 0 (no implementation), 10 (a partially correct solution), 20 (a correct solution) and REPORT: 10
- **30 (part 4):** CODE: 0 (no implementation), 10 (a partially correct solution), 20 (a correct solution) and REPORT: 10
- **25 (bonus):** CODE: 0 (no implementation), 10 (a partially correct solution), 20 (a correct solution) and REPORT: 5

Late Policy

You have three days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90, 80 and 70 for each late submission day). You have to submit your solution in deadline date + three days, otherwise it will not be evaluated.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] https://en.wikipedia.org/wiki/Deconvolution
[2] https://en.wikipedia.org/wiki/Richardson–Lucy_deconvolution
[3] http://webdav.is.mpg.de/pixel/benchmark4camerashake/
[4] https://www.cis.rit.edu/class/simg782/lectures/lecture_16/lec782_05_16.pdf