Final Project Report
Blind Deconvolution with Richardson-Lucy Algorithm

Xiaoyu Bi
xiaoyubi@andrew.cmu.edu

Abstract

Blind deconvolution aims to recover the latent image from a blurred one while the point spread function is unknown. This a hard problem and many advanced techniques are applied to solve the problem. Instead, we try to research on a simple iterative method based on Richardson-Lucy algorithm to conduct blind deconvolution and show its effectiveness under certain conditions with experiment results.

1. Introduction and Literature

To recover the original image from a blurred one is a hard problem, especially when you do not know the point spread function (PSF), which we can also simply call the blurred kernel. The above task could be done with blind deconvolution, and we have learned about that in lectures. However, within most recent advanced blind deconvolution algorithms, there are complicated optimization problems, signal processing, Fourier transformations and even deep neural networks. Without many experiences and practices in blind deconvolution, it could cost plenty of effort to get familiar with those difficult concepts. And this problem inspires the main question for the project, could we find an easy approach to complete blind deconvolution?

The short answer would be Yes. But before looking into the hard blind deconvolution problem, let's first handle a simpler one, non-blind deconvolution with a known point spread function. Richardson-Lucy algorithm [4] [3] is proposed to solve non-blind deconvolution and it is an iterative approach for recovering a latent image that has been blurred by a known PSF. During each iteration, there are only a few simple steps without advanced operations. And it has been proved to converge to a maximum likelihood solution. Though Richardson-Lucy algorithm did not solve the exact blind deconvolution problem that we want, it inspires the followers. Later, based on the idea of R-L algorithm, [2] proposes a new algorithm for blind deconvolution. In this project, we will call this new algorithm the "Blind Richardson-Lucy Algorithm".

In Section 2, we will introduce the detail of Richardson-Lucy algorithm and show some experimental results with it. We will also discuss how to mitigate some artifact problems caused by Richardson-Lucy algorithm. In Section 3, we will show the detail of the Blind Richardson-Lucy algorithm with some sample results. Finally, we will conclude the project with some limitations and future works in Section 4.

2. Richardson-Lucy Algorithm

A blurred image could be denoted as following:

$$B = I \circledast K + N \tag{1}$$

where B is the blurred image, I is the latent image, $\circledast$ stands for the convolution operation, K is the point spread function (blur kernel) and N is noise added to the blurred image. The goal of Richardson-Lucy algorithm is to recover the latent image I, given that blurred image B and PSF K are already known.

2.1. Algorithm Detail

Richardson-Lucy algorithm [4] [3] is an iterative approach for recovering a latent image that has been blurred by a known point spread function. It is named after William Richardson and Leon Lucy, who described it independently in the 1970s. Obviously, this specific algorithm cannot handle blind deconvolution, but we will see how it will help later. The iteration process of this algorithm could be captured by the following formula:

$$\hat{I}^{t+1} = \hat{I}^t \cdot \left(\frac{B}{\hat{I}^t \circledast K} \circledast K^*\right) \tag{2}$$

where $\hat{I}^t$ is the latent image estimation at iteration t, and K^* is the flip of point spread function K. Flip just means reversing the order of every element. With this format, only several simple operations are required in each iteration, making it very computationally efficient. Besides, it has been proved that it is able to converge to the maximum

likelihood solution for the latent image and the complete derivation could be found at [1].

We will use the Peak Signal-to-Noise Ratio (PSNR) as the evaluation metrics for this algorithm in later experiments. Peak Signal-to-Noise Ratio is the ratio between the maximum possible power of a signal and the power of corrupting noise, and we use Mean Squared Error (MSE) here as the noise. Therefore, the PSNR value could calculated as:

$$PSNR = 20 \cdot \log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \tag{3}$$

where MAX_I is the maximum possible pixel value of the original image (in range $[0, 1]$ in our case) and MSE if the mean squared error between recovered image and original image. Generally the higher this PSNR value is, the better restoration quality we have.

2.2. Sample Results

All sample results shown in this paper are using the same original image Figure 1. For simplicity, we only consider the deconvolution on gray-scale images. The extension to color images should be straightforward, because you only need to conduct deconvolution separately in three different color channels.

Figure 1. The Original Image

Figure 2 shows the result of deconvolution with Richardson-Lucy algorithm, where Figure 2 (a) is the blurred image with PSNR **24.86** and Figure 2 (b) is the recovered image with PSNR **21.30**. Comparing the blurred image and recovered image, we can clearly tell that the recovered image is visually less blurry, however it has a lower PSNR value. The problem here is because of the convolution operation. To make the output have the same size as input, convolution operation will automatically pad zeros

Figure 2. Restoration with R-L Algorithm

around the boundaries, which will finally result in the "ringing" artifacts at boundaries of the recovered image. So as to get a better global performance in terms of PSNR value, we need some modifications on R-L algorithm.

2.3. Sample Results with Boundary Wrapping

Fortunately, the modification is not very hard. To mitigate this kind of "ringing" artifacts at boundaries, we could wrap the boundary first and then apply the R-L algorithm. To achieve this, we use the $cv2.copyMakeBorder()$ function to create a new paddings around the image with $BORDER\_WRAP$ type of border. It is quite hard to explain what it means to "wrap" the boundary, but in 1D if you want to "wrap" a sequence "$abcdefgh$" it will look like: "$cdefgh|abcdefgh|abcdefg$". (There are also many other options including $BORDER\_REFLECT$, $BORDER\_REPLICATE$ and so on. We decide with $BORDER\_WRAP$ simply because it has the highest PSNR results.)

Figure 3. Restoration with R-L Algorithm and Boundary Wrapping

Figure 3 shows the result of R-L deconvolution with boundary wrapping, where Figure 3 (a) is the blurred image with PSNR **24.94** and Figure 3 (b) is the recovered image with PSNR **29.73**. With this modification, not only do we have visually compelling results after restoration, but also a higher PSNR value because there are clearly less artifacts on the boundaries. Besides, it only takes about 25 iterations to reach this result, which shows that the convergence of R-L algorithm is quite fast.

3. Blind Richardson-Lucy Algorithm

The simple iterative Richardson-Lucy algorithm has shown its effectiveness in non-blind deconvolution, but how about our initial goal of blind deconvolution?

3.1. Algorithm Detail

With both latent image I and point spread function K in Equation 1 unknown, of course we cannot directly apply Richardson-Lucy algorithm. But based on the idea how latent image is estimated during iterations, [2] proposes a similar methods to estimate point spread function as well during iterations:

$$\hat{I}^{(t+1)} = \hat{I}^{(t)} \cdot \left(\frac{B}{\hat{I}^{(t)} \circledast \hat{K}^{(t)}} \circledast \hat{K}^{(t)*}\right) \tag{4}$$

$$\hat{K}^{(t+1)} = \hat{K}^{(t)} \cdot \left(\frac{B}{\hat{I}^{(t+1)} \circledast \hat{K}^{(t)}} \circledast \hat{I}^{(t+1)*}\right) \tag{5}$$

where Equation (4) is the same as the previous R-L algorithm, and Equation (5) is to estimate the PSF every iteration in a similar way. The main difference between this blind R-L algorithm and the previous R-L algorithm is that we need two estimation of latent image and PSF in the first place. The latent image initialization is not very tough, but the result performance might heavily depend on the PSF initialization, especially depend on the kernel size instead of the kernel values. Though this method is still relatively simple with iterations, we do not have convergence guarantee this time. If with too many iterations, this method might overshoot to some extreme points.

3.2. Sample Results

Figure 4. Restoration with Blind R-L Algorithm

Figure 4 shows the result of restoration with blind R-L algorithm. The left image is blurred from the original image with a random 5*5 PSF and it has PSNR **25.69**. And the right image is recovered from the blurred image with a full 5*5 PSF initialization, which results in PSNR **26.28**. This restoration result with a PSF initialization of the same size as the true PSF is visually less blurry and has a higher PSNR value. Besides, it only takes 10 iterations to reach the result this time, which shows that the blind R-L algorithm is still fast to converge. However, the PSNR improvement is not as significant as the non-blind R-L algorithm. And if you look closely into the recovered image, there are actually "ringing" artifacts on edges of objects in the scene. (These artifacts might be hard to capture due to the resolution here.)

Figure 5. Restoration with Blind R-L Algorithm

Figure 5 shows the result of another restoration with blind R-L algorithm. The left image is blurred from the original image with a same random 5*5 PSF as in Figure 4 (a) and it also has PSNR **25.69**. And the right image is recovered from the blurred image with a full 9*9 PSF initialization, which results in PSNR **23.39**. We can obviously tell that there are lots of artifacts in the recovered image and it even has a lower PSNR value than the blurred image. We could conclude from the performance that the restoration of blind R-L algorithm depends heavily on the PSF initialization, especially on the size of PSF initialization. If the size of initial PSF estimation is close to the true PSF size, then we might have a good restoration performance, otherwise it could be even worse than the blurred image.

4. Conclusion

In this project, we implement the non-blind and blind Richardson-Lucy algorithms and show their effectiveness under some certain conditions with experiment results. Back to our main research question at the introduction section, we could answer that the iterative blind R-L algorithm is indeed a simple way to deal with blind deconvolution, but apparently it has lots of limitations for now. First, it requires a good estimation of the kernel size of PSF for good performance, which could be very hard in real-world deconvolution applications. Second, in terms of PSNR values, the restoration improvement in the "blind" setting is not very significant compared to that in the "non-blind" setting. More works on these limitations could be done in the future.

References

[1] Richardson–lucy deconvolution. https://en.wikipedia.org/wiki/Richardson%E2%80%93Lucy_deconvolution. Accessed: 2022-12-20. 2

[2] Henri Lanteri, Claude Aime, Hubert Beaumont, and Philippe Gaucherel. Blind deconvolution using the richardson-lucy algorithm. In *Optics in Atmospheric Propagation and Random Phenomena*, volume 2312, pages 182–192. SPIE, 1994. 1, 3

[3] Leon B Lucy. An iterative technique for the rectification of observed distributions. *The astronomical journal*, 79:745, 1974. 1

[4] William Hadley Richardson. Bayesian-based iterative method of image restoration. *JoSA*, 62(1):55–59, 1972. 1