Image Deblurring Tutorial

Jian-Jiun Ding

Chun-Lin Liaw

September, 2022

Contents

1 Abstract **3**

2 Introduction **4**
2.1 Problem Statement 4
2.2 Method Classification 6
2.3 Problem Formulation 7

3 Conventional Methods **8**
3.1 Regularization 8
3.2 Iterative Regularization 14
3.3 Statistical 23
3.4 Chapter Summarize 29

4 Uniform Deblurring **30**
4.1 Non-blind Deblurring 30
4.1.1 Fast Image Deconvolution using Hyper-Laplacian Priors [7]
(code path: Deblurring_Gather/fastdeconv/test_fast_deconv.m)
. 30
4.1.2 A General Framework for Regularized, Similarity-Based Image Restoration [6]
(code path: Deblurring_Gather/DeblurringCode/mainDemo.m)
. 33
4.1.3 Image Deblurring Using a Pyramid-Based Richardson–Lucy Algorithm [3]
. 37
4.2 Blind Deblurring 39
4.2.1 Digital Image Restoration for Phase-Coded Imaging Systems [16] 39
4.2.2 Blind Deconvolution Using a Normalized Sparsity Measure [8]
(code path: Deblurring_Gather/online_code/test_blind_deconv.m)
. 43
4.2.3 Deblurring Shaken and Partially Saturated Images [17] 46
4.2.4 Deblurring Text Images via L0-Regularized Intensity and Gradient Prior [11]
(code path: Deblurring_Gather/text_deblurring_code_v4/demo_text_deblurring.m)
. 49
4.2.5 Blind Image Deblurring Using Dark Channel Prior [12]
(code path: Deblurring_Gather/cvpr16_deblurring_code_v1/demo_deblurring.m) 53

5 Non-uniform Deblurring **57**
5.1 Image and Depth from a Conventional Camera with a Coded Aperture [9]
(code path: Deblurring_Gather/DeconvolutionCode-LevinEtAl07)
. 57

5.2 Spatially-Varying Out-of-focus Image Deblurring with L1-2 Optimization and A Guided Blur Map [14]
(code path: Deblurring_Gather/ICASSP2012/main.m) . . . 61
5.3 Unnatural L0 Sparse Representation for Natural Image Deblurring [20]
(code path: Deblurring_Gather/Non_Uniform_Pcode/runNon_Uniform_L0_Deblur.m) . . . 64

6 Deep Learning in Image Deblurring **68**
6.1 Non-blind Deblurring . . . 68
6.1.1 Deep Convolutional Neural Network for Image Deconvolution [19]
(code path: Deblurring_Gather/dcnn_nips14/run_deblur_all_3chs_finetune.m) . . . 68
6.1.2 Restoring an Image Taken through a Window Covered with Dirt or Rain [4] . . . 70
6.2 Blind Deblurring . . . 72
6.2.1 Convolutional Neural Networks for Direct Text Deblurring [5] . . . 72

7 A Short Comment on This Tutorial **74**

Appendices **75**

A Appendix A:
Issues of MATLAB Implementation **76**

B Appendix B:
Iteratively Reweighted Least Squares(IRLS) **78**

References **79**

Abstract

Thanks to the explosive growth of social media, people nowadays tend to share images with each other frequently. Together with the highly developed technology, it is extremely convenient to take pictures almost anywhere nowadays. However, these hand-taken images might suffer from certain kinds of blur effect. Researchers thus have proposed numerous post-processing methods to recover blur images to sharp ones.

In this tutorial , we aim to address this problem by introducing some well-known methods by classifying these methods into different categories.

Introduction

2.1 Problem Statement

First things first, let's formulate two dimensional blurring process as follows

$$b(\mathbf{x}) = \int k(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')\ d\mathbf{x}' + n(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = \{x_{vertical}, x_{horizontal}\}$, $b(x)$ is the blurred signal, $k(\mathbf{x}, \mathbf{x}')$ is the *point spread function*, $f(\mathbf{x}')$ is the original signal, and $n(\mathbf{x})$ denotes the noise. The goal is to reconstruct $f(\mathbf{x}')$ as precise as possible. If the point spread function is invariant over entire integral space, we can translate $k(\mathbf{x}, \mathbf{x}')$ to $k(\mathbf{x} - \mathbf{x}', 0)$ which is a function of $\mathbf{x} - \mathbf{x}'$. Thus, rewriting (1) gives

$$b(\mathbf{x}) = \int k(\mathbf{x} - \mathbf{x}')f(\mathbf{x}')\ d\mathbf{x}' + n(\mathbf{x}) \tag{2}$$

Noted that (2) shows that the deblurring process can also be seen as a deconvolution problem in space-variant case. As a result, two dimensional deconvolution is a sub-problem of image deblurring, but not the other way. In this tutorial, space-invariant cases are focused rather than space-variant.

To give a concrete idea of (2), we consider three kinds of blurred image with different blur kernels: Gaussian blur, motion blur, and out-of-focus blur. For a Gaussian blur with zero mean and unit variance, its kernel is written as

$$k(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\mathbf{x}^2}{2}}. \tag{3}$$

The result is shown below,

Figure 2.1: *left*: original image, *right*: blurred image

One can see that the space-invariant kernel blurs the entire image uniformly. As for motion blur, it stimulates hand shake. The kernel is written as

$$k(\mathbf{x}) = \frac{1}{T}\int_0^T \delta[\mathbf{x} - \mathbf{a}(t)]dt \tag{4}$$

where $\mathbf{a}(t)$ is the time-dependent 2D translation trajectory of the shake motion. The figure below is generated in MATLAB using a $(45°, 10\%)$ linear motion blur,

Figure 2.2: *left*: original image, *right*: blurred image

The above blurring processes are space-invariant since every pixel in the image experience same convolution process. Out-of-focus, on the other hand, is an example of space-variant blurring. The effect stems from the fact that when taking a picture, the camera is actually mapping objects of different *depth of field* (DOF) to a image plane. Figure below shows a scenario where the camera focus on different object in a same environment.

Figure 2.3: *left*: focus on soldier, *right*: focus on the poker

In this case, the kernel can be written as

$$k(\mathbf{x}) = \frac{1}{\pi D^2}\chi_{coc}(\mathbf{x}) \tag{5}$$

Since the complete explanation requires domain knowledge in optics, this tutorial is not going any further. Roughly speaking, if a point source on a object is not focused on, the point source will become a *circle of confusion* (COC) with radius D. Additionally, the characteristic function of the COC is $\chi_{coc}(\mathbf{x})$. It is obvious that D and $\chi_{coc}(\mathbf{x})$ varies from object to object in different places. Therefore, out-of-focus blur is a space-variant blurring process.

2.2 Method Classification

Carrying on the previous section, the mission of image deblurring is to solve equation (2) given $b(\mathbf{x})$. Now, if $k(\mathbf{x}, \mathbf{x}')$ is also acquired, we call the problem a ***non-blind deblurring*** process. On the contrary, when $k(\mathbf{x}, \mathbf{x}')$ is not available, it becomes a ***blind deblurring*** process where both $f(\mathbf{x}')$ and $k(\mathbf{x}, \mathbf{x}')$ have to be estimated. Therefore, image deblurring methods can be grouped into two categories based on whether $k(\mathbf{x}, \mathbf{x}')$ is known. In another perspective, different mathematical means are used to solve the problem. In this tutorial, three basic groups are introduced: regularization methods, iterative regularization methods, and statistical methods.

The methods classified by different mathematical tools are mainly referred to [1] while the non-blind and blind deblurring methods mostly come from milestone papers throughout these years. The reason of such framework is that in [1], the authors provide important fundamentals of image deblurring and explicit categories of each method. However, many novel methods proposed in the past few decades are using two or more mathematical tools at the same time. What's more, blind deblurring problem becomes vial as the need of single image deblurring has arisen. To sum up, this tutorial can be broke down into two part in the big picture: highlights of fundamentals in [1] and surveys of recent progress in image deblurring. However, the following content aims to illustrate the concept behind each methods rather than detailed derivation. For a better understanding, referring to the sources attached to each methods is highly recommended.

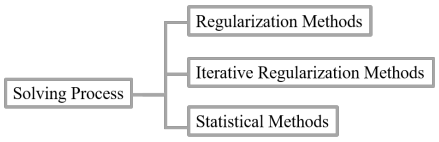

Figure 2.4: Classification Based on [1]

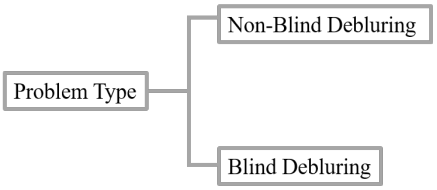

Figure 2.5: Rough Classification of Recent Progress in Image Deblurring

2.3 Problem Formulation

Before going into further discussion, one may notice that equation (2) implies that images are continuous signals. However, images are in fact discrete and often regard as matrices in practical. In the case of 1D vector of size N, equation (2) is rewrite as

$$g_m = \sum_{k=0}^{N-1} K_{m-k} f_k + w_m = (\mathbf{A}f)_m + w_m \tag{6}$$

or equivalently

$$\mathbf{g} = \mathbf{K} * \mathbf{f} + \mathbf{w} = \mathbf{A}\mathbf{f} + \mathbf{w} \tag{7}$$

where $\mathbf{A} = K_{m-n}$. By expanding (6), the matrix $\mathbf{A}$ is in the form of

$$\mathbf{A} = \begin{bmatrix} K_0 & K_{N-1} & \dots & K_1 \\ \vdots & \ddots & \vdots & \\ K_{N-1} & K_{N-2} & \dots & K_0 \end{bmatrix} \tag{8}$$

With such periodicity, $\mathbf{A}$ is recognized as a *cyclic matrix* and a particular case of *Toeplitz matrix* as well. In the following content, equation (7) is adapted in most cases as default formulation of blurring process.

To solve (7), a straightforward method is to apply Fourier transform on (7) directly: $\hat{K}(w)\hat{f}(w) + \hat{w}(w) = \hat{g}(w)$. Dividing $\hat{K}(w)$ on both side, one get $\hat{f}_{recover}(w) = \frac{\hat{g}(w)}{\hat{K}(w)} = \hat{f}(w) + \frac{\hat{w}(w)}{\hat{K}(w)}$. The second term implies that the result will encounter inevitable failure at the frequency band where $\hat{K}(w)$ equals 0. Additionally, low SNR of the blurred image will also result in poor restored image.

Conventional Methods

3.1 Regularization

As its name suggests, regularization methods solve the problem by adding constrains on the target image. First, define the discrepancy function as

$$\varepsilon^2(f;g) = \|Af - g\|^2 \tag{9}$$

where A is a 2D cyclic matrix corresponding to the point spread function in (2) and g is the known blurred image. Additionally, another energy function $E(f)$ is defined as

$$E^2(f) = \|f\|^2 \tag{10}$$

The problem is as such: *Given a prescribed energy $E^2(f)$, find f that minimize $\varepsilon^2(f;g)$*. The effect of adding energy constraints is to suppress noise amplification. Recall that the before-mentioned Fourier transform method will amplify noise component to infinite at places where kernel equals zero. Taking conditions (9)(10), this minimizing problem can be solved by Lagrange multipliers

$$\Phi_\mu(f;g) = \|Af - g\|^2 + \mu\|f\|^2 \tag{11}$$

By the use of Parseval equality, (11) can be rewrote as

$$\Phi_\mu(f;g) = \frac{1}{(2\pi)^q}\int_B \left|\hat{K}(w)\hat{f}(w) - \hat{g}(w)\right|^2 dw + \frac{1}{(2\pi)^q}\int_B \left|\hat{f}(w)\right|^2 dw \tag{12}$$

(12) can be further rearrange as

$$\Phi_\mu(f;g) = \frac{1}{(2\pi)^q}\int_B (\left|\hat{K}(w)\right|^2 + \mu)\left|\hat{f}(w) - \frac{\hat{K}^*(w)\hat{g}(w)}{\left|\hat{K}(w)\right|^2 + \mu}\right|^2 dw + \frac{\mu}{(2\pi)^q}\int_B \frac{|\hat{g}(w)|^2}{\left|\hat{K}(w)\right|^2 + \mu}dw \tag{13}$$

One can discover that $\Phi_\mu(f;g)$ will have minimal value when the first term equal 0. Under such condition, the restored $\hat{f}_\mu(w)$ is given by

$$\hat{f}_\mu(w) = \frac{\hat{K}^*(w)\hat{g}(w)}{\left|\hat{K}(w)\right|^2 + \mu} \tag{14}$$

Note that (14) is the Fourier transform of restored image. The final result is obtained by inverse Fourier transform

$$f_\mu(x) = \frac{1}{(2\pi)^q} \int \frac{\hat{K}^*(w)\hat{g}(w)}{\left|\hat{K}(w)\right|^2 + \mu} e^{ix\cdot w} dw \tag{15}$$

Recall that $f_\mu(w)$ is obtained under the assumption: *given a prescribed energy $E^2(f)$, find f that minimize $\varepsilon^2(f;g)$*. By calculating the energy of $f_\mu(w)$, we can get the following inequality

$$\begin{aligned} E^2(f) = \int |f(x)|^2 \, dx = \frac{1}{(2\pi)^q} \int_B \frac{\left|\hat{K}(w)\right|^2}{(\left|\hat{K}(w)\right|^2 + \mu)^2} |\hat{g}(w)|^2 \, dw &\leq \frac{1}{\mu(2\pi)^q} \int_B |f(w)|^2 \, dw \\ &\leq \frac{1}{\mu}\|g\|^2 \end{aligned} \tag{16}$$

That is to say, by choosing a proper μ, we can ensure that the energy of the restored image $f_\mu(w)$ is under a specific value since the energy of blurred image $\|g\|^2$ is already known. In fact, consider the dual problem: *given a prescribed discrepancy value $\varepsilon^2(f;g)$, find f that minimize energy $E^2(f)$*. We can find that the Lagrange multiplier of this problem is also (11). Thus, using the result in (15), we can again calculate the discrepancy value

$$\begin{aligned} \varepsilon^2(f_\mu; g) &= \frac{1}{(2\pi)^2} \int \left| \frac{\left|\hat{K}(w)\right|^2}{\left|\hat{K}(w)\right|^2 + \mu} \hat{g}(w) - \hat{g}(w) \right|^2 dw \\ &= \frac{1}{(2\pi)^2} \int_B \frac{\mu^2 |\hat{g}(w)|^2}{(\left|\hat{K}(w)\right|^2 + \mu)^2} dw + \frac{1}{(2\pi)^2} \int_{\bar{B}} |\hat{g}(w)|^2 \, dw \end{aligned} \tag{17}$$

In the second of (17), we separate the frequency domain integral into two parts. The first term is the same as the first line, while the second term assumes that $\hat{K}(w)$ is band-limited and thus $\hat{K}(w) = 0$ in $\bar{B}$. If $\mu = 0$, (17) becomes $\|g\|^2$. And as $\mu \to \infty$, (17) is simply $\|g_{out}\|^2$ where g_{out} represents the out-of-band component of blurred image. Therefore, the discrepancy function is bounded by

$$\|g_{out}\|^2 < \varepsilon < \|g\|^2 \tag{18}$$

Given a valid prescribed discrepancy energy $\varepsilon^2(f_\mu; g)$ a corresponding μ can be found such that $E^2(f)$ is minimized.

By far, we've illustrated the characteristics of regularization method and gave brief derivations of the result. The main idea is to *add constraints on restored image*. By using regularization term, one cane ensure that the discrepancy $\varepsilon^2(f;g) = \|Af - g\|^2$ and energy $E^2(f) = \|f\|^2$ will not exceed a given value. One can find that the main advantage of regularization method is its computing speed. The result is an one-step calculation as formulated in (15). However, regularization method has a few drawbacks as well. For example, putting energy constraints on a image is a rather superficial way of image deblurring since it does not consider any characteristics of natural images other than energy. Additionally, regularization method has a trade-off between resolution and ringing effect.
The ringing effect is also known as Gibbs oscillations. To address this phenomenon, let's rewrite (15) as

$$f_\mu(x) = \frac{1}{(2\pi)^q} \int_B \hat{W}(w) \frac{\hat{g}(w)}{\hat{K}(w)} e^{ix\cdot w} dw \tag{19}$$

where

$$\hat{W}(w) = \frac{\left|\hat{K}(w)\right|^2}{\left|\hat{K}(w)\right|^2 + \mu} \tag{20}$$

The reason of rewriting (15) into (20) is to show that the result $f_\mu(w)$ is in fact the multiplication of two elements in frequency domain(i.e., convolution in spatial domain). The second element $\frac{\hat{g}(w)}{\hat{K}(w)}$ is the direct division in frequency domain. Recall that in page 6, we've discuss this may cause noise amplification. As a result, one can find that regularization method is nothing more than *adding a filter* $\hat{W}(w)$ *to the direct division of blurred image and kernel.*

Now, we can further investigate the relationship between Gibbs oscillations and (20). The following plot shows the relation of μ and the inverse Fourier transform of $\hat{W}(w)$.

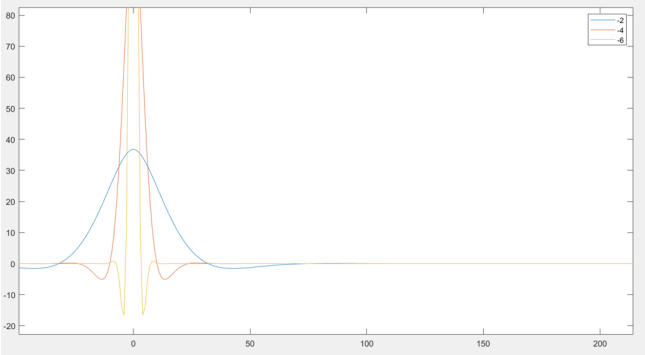

Figure 3.1: *horizontal axis*: $\log_{10}\mu$, *vertical axis*: $W(w)$

According to (11), bigger μ implies stronger energy in the restored image. In other words, an increasing μ will suppress the oscillation, while a decreasing μ will generate a sharper image compared to the blurred image. For a better understanding, a simple example is given below

Figure 3.2: (from left to right, from top to bottom):original image, motion blur, AWGN, $\mu = 0.01$ restoration

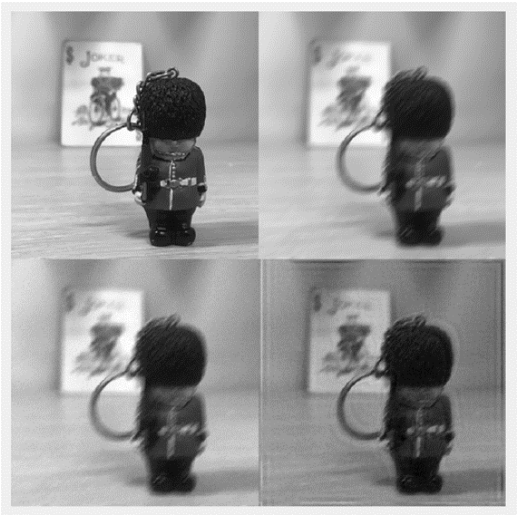

Figure 3.3: (from left to right, from top to bottom):original image, motion blur, AWGN, $\mu = 0.05$ restoration

Here we discuss two possible ways of choosing parameter μ:

- *the Miller method:*
  If we put constrains on the set of all objects such that $\|Af - g\|^2 \leq \varepsilon^2, \quad \|f\|^2 \leq E^2$, μ is chosen as $\mu_{Miller} = (\frac{\varepsilon}{E})^2$
- *the L curve method:*
  If we plot μ against $E(\mu)$ and $\varepsilon(\mu)$, the curve looks like the letter L. The value of μ at the turning point is chosen. The below figure shows the L curve in this case($\mu = 0.0001 : 0.001 : 0.2$).

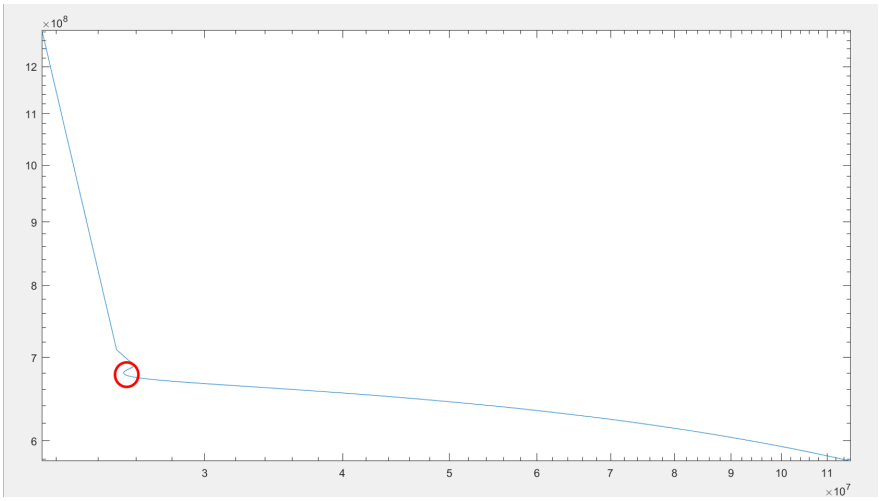

Figure 3.4: x-axis: $\varepsilon(\mu)$, y-axis: $E(\mu)$

It is worth notice that the methods of choosing μ may not result in the optimal solution visually. The reason that μ only regularize power of image and its discrepancy. Therefore they merely represent the optimal solution from the standpoint of signal power.

The results are stimulated via MATLAB using discrete version of (15).

$$(\mathbf{f}_\mu)_{m,n} = \frac{1}{HW} \sum_{k=1}^{H} \sum_{l=1}^{W} \frac{\hat{K}*_{k,l}}{\left|\hat{K}_{k,l}\right|^2 + \mu} e^{i\frac{2\pi}{H}km} e^{i\frac{2\pi}{W}ln} \tag{21}$$

H is the height and W is the width of the blurred image. For additional implement details, please refer to the code. As one can see, When $\mu = 0.01$, there are sever ringing effects in the restored image. On the other hand, when $\mu = 0.05$, the oscillations reduce but the resolution drops at the same time. The trade-off between resolution and oscillation shows the limitation of regularization method in some way. As a matter of fact, the above deblurring process using regularization is a special case of Tikhonov regularization in linear regression. A standard approach of Tikhonov regularization is as follows:

Given a matrix A and vector $\boldsymbol{b}$, find a vector $\boldsymbol{x}$ such that $A\boldsymbol{x}$=$\boldsymbol{b}$.

Sometimes the problem is ill-posed because there might be more than one $\mathbf{x}$ satisfy the equation or no $\mathbf{x}$ is valid. Hence, the problem seeks to minimize $\|A\mathbf{x} - \mathbf{b}\|_2^2$. If we include a regularization term into the equation, the optimize function $\|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\Gamma\mathbf{x}\|_2^2$ can be solved numerically as

$$\hat{x} = (A^\top A + \Gamma^\top \Gamma)^{-1} A^\top \mathbf{b} \tag{22}$$

where Γ Tikhonov matrix , often choose as $\Gamma = \alpha I$. The object function becomes $\|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\alpha\mathbf{x}\|_2^2$ which is in exact same form as (11).

Now, we try to solve the original problem (11) in terms of (22). Assume that there is a optimal restored image f_μ for equation (11) such that for any complex scalar α and complex vector h

$$\begin{aligned}\Phi(f_\mu; g) \leq \Phi(f_\mu + \alpha h; g) = &\|Af_\mu - g\|_2^2 + \mu\|f_\mu\|_2^2 \\ &+ \alpha[(Ah, Af_\mu - g) + \mu(h, f_\mu)] \\ &+ \alpha^*[(Af_\mu - g, Ah) + \mu(f_\mu, h)] \\ &+ |\alpha|^2 (\|Ah\|_2^2 + \mu\|h\|_2^2)\end{aligned} \tag{23}$$

Note that A here is a linear operator rather than a matrix. (Every matrix is a linear operator but not all the linear operator can be represented as matrix.) Since the fourth term in (23) is always positive, the second and the third term must be 0 to ensure the inequality holds. The brackets $(\cdot, \cdot)$ in the equation is inner product in Hilbert space. Using the property of adjoint operator $(Ax, y) = (x, A^*y)$, we have

$$\alpha(h, A^*Af_\mu + \mu f_\mu - A^*g) + \alpha^*(A^*Af_\mu + \mu f_\mu - A^*g, h) = 0 \tag{24}$$

The commutative property further gives $(A^*Af_\mu + \mu f_\mu - A^*g, h) = 0$ and equivalently $(A^*A + \mu I)f_\mu = A^*g$ for any complex h. Finally, we can compare this result to (22).

$$f_\mu = (A^*A + \mu I)^{-1}A^*g \tag{25}$$

According to Riesz representation theorem, if A has a transform matrix M with respect to orthonormal basis, then A^* corresponds to the conjugate transpose M^H, resulting the complex representation of (22)

$$f_\mu = (M^HM + \mu I)^{-1}M^Hg \tag{26}$$

To sum up, in this section we discuss the fundamentals of image deblurring in terms regularization. This method plays a part in image filtering. And the implement result shows that the restoration is relatively fast , yet its performance is affected by the trade-off between resolution and ringing. In the next section, we will introduce iterative regularization method which is considered as an improved version of the current one.

3.2 Iterative Regularization

In the previous chapter, we try to solve image deblurring problem by a single equation (21). However, we can actually solve this problem in a multi-step manner. To begin with, let's revisit equation (25). If $\mu = 0$ (i.e., without regularization term), then the equation becomes

$$A^*Af = A^*g \tag{27}$$

The convolution equation is thus generated by replacing A^*A and A^*g by $\bar{A}$ and $\bar{g}$ respectively,

$$\bar{A}f = \bar{g} \tag{28}$$

A straightforward way of solving (28) iteratively is to approach the optimal answer each step via

$$f_{k+1} = f_k + \tau(\bar{g} - \bar{A}f_k) \tag{29}$$

where f_k is the output of the kth iteration and τ is *relaxation parameter*. Note that the second term on the right hand side is a vector representing the distance and direction between $\hat{g}$ and $\hat{A}f_k$. Then, we rearrange the equation and put it in frequency domain,

$$\hat{f}_{k+1}(w) = \tau\hat{\bar{g}}(w) + (1 - \tau\hat{H}(w))\hat{f}_k(w) \tag{30}$$

One can easily see that expanding (30) gives

$$\begin{aligned}\hat{f}_k(w) =&(1 - \tau\hat{H}(w))^k\hat{f}_0(w) \\ &\tau\{1 + (1 - \tau\hat{H}(w))(1 - \tau\hat{H}(w))^2 + \ldots (1 - \tau\hat{H}(w))^{k-1}\}\hat{\bar{g}}(w)\end{aligned} \tag{31}$$

Using the sum of geometric progression, (31) gives

$$\hat{f}_k(w) = (1 - \tau\hat{H}(w))^k\hat{f}_0(w) + \{1 - (1 - \tau\hat{H}(w))^k\}\frac{\hat{\bar{g}}(w)}{\hat{H}(w)} \tag{32}$$

Take a closer look at the equation, one can find that the restored image is a linear combination of general solution $\frac{\hat{\bar{g}}(w)}{\hat{H}(w)}$ and the initial estimation image $\hat{f}_0(w)$

Notice that (32) is the result *without* regularization term. In the following content, we first illustrate the original van Cittert and Landweber methods, then take regularization term into account, and finally introduce the steepest descent and the conjugate gradient method(CG).

First, recall that for a image blurring model $g = Af + w$, we've derived (27) as its solution. And based on (27), an iterative method to solve the problem is introduced as (32). In such case, $\hat{H}(w)$ is the Fourier transform of $\bar{A}$, and by definition $\bar{A} = A * A$ which gives $H(w) = |K(w)|^2$. Now, consider a super simple case where there are no noise were added ($Af = g$). In this case, $\bar{A} = A$ and $H(w) = K(w)$. The above concept is the main difference of the van Cittert method and the Landwever method.

- *the van Cittert method:*
  The method that handles blurred image without additive noise.

$$\hat{f}_k(w) = (1 - \tau\hat{K}(w))^k \hat{f}_0(w) + \{1 - (1 - \tau\hat{K}(w))^k\} \frac{\hat{\bar{g}}(w)}{\hat{K}(w)} \tag{33}$$

- *the Landweber method:*
  The method that consider general blurred image with noise.

$$\hat{f}_k(w) = (1 - \tau \left|\hat{K}(w)\right|^2)^k \hat{f}_0(w) + \{1 - (1 - \tau \left|\hat{K}(w)\right|^2)^k\} \frac{\hat{\bar{g}}(w)}{\left|\hat{K}(w)\right|^2} \tag{34}$$

In the following discussion, we focus on the Landweber method. The final result is obtained by setting $\hat{f}_0(w) = \mathbf{0}$ (i.e., the first estimation is a zero matrix) and an inverse Fourier transform of (34)

$$\begin{aligned} f_k(x) &= \frac{1}{(2\pi)^q} \int_B \hat{W}^{(k)}_{Land}(w) \frac{\hat{g}(w)}{\hat{K}(w)} e^{ix\cdot w} dw \\ \hat{W}^{(k)}_{Land}(w) &= 1 - (1 - \tau \left|\hat{K}(w)\right|^2)^k \end{aligned} \tag{35}$$

Compare (35) to Tikhonov regularization in (19), one can find that the Landweber method is in fact another kind of filter acting on the general solution $\frac{\hat{g}(w)}{\hat{K}(w)}$. That is to say, the number of iteration is nothing more than a parameter in filtering just as μ in the Tikhonov regularization method. When $\hat{K}(w)$ is small,

$$\begin{aligned} \hat{W}(w)_{Tikhonov} &\approx \frac{1}{\mu} \left|\hat{K}(w)\right|^2 \\ \hat{W}(w)_{Landweber} &\approx k\tau \left|\hat{K}(w)\right|^2 \end{aligned} \tag{36}$$

The window function of Tikhonov and Landweber can relate to each other by $\mu = 1/\tau k$

As we can see from the above discussion, the Landweber method is not a genuine iterative method since it can be represented in the form of single filtering process in (32). Therefore we consider a stricter problem:

$$\|Af - g\| = minimum,\ f \in C \tag{37}$$

The constraint on f ensures that the restored image must be in a given closed and convex set C. The following algorithm inherits the notations in (28)

1. compute $\hat{f}_k^{(C)}(w)$ from $f_k^{(C)}(x)$
2. compute

$$\hat{h}_{k+1}(w) = \tau \hat{K}^*(w)\hat{g}(w) + (1 - \tau \left|\hat{K}(w)\right|^2)\hat{f}_k^{(C)}(w) \tag{38}$$

3. compute $h_{k+1}(x)$ from $\hat{h}_{k+1}(w)$
4. compute $f_{k+1}^{(C)}(x) = (P_C h_{k+1})(x)$

The idea here is to project the interior result of each iteration onto the given set C. The method is thus named *the projected Landwever method*. The operator P_C depends on how one constrains the restored result. For example, P_C can be a simple non-negative function or a saturation function

$$non - negative : P_C f(x) = \begin{cases} f(x) & if \quad f(x) > 0 \\ 0 & if \quad f(x) \leq 0 \end{cases} \tag{39}$$

$$saturation : P_C f(x) = \begin{cases} f(x) & if \quad a \leq f(x) \leq b \\ a & if \quad f(x) < a \\ b & if \quad f(x) > b \end{cases} \tag{40}$$

So far, we've introduce the general iterative method and projected Landweber method. Recall that the above manners are based on (27) where no regularization terms considered. We now discuss how to solve the problem with regularization term iteratively.

$$\Phi_\mu(f; g) = \|Af - g\|^2 + \mu\|f\|^2 = minimum, \quad f \in C \tag{41}$$

As we've proved in (25), the solution of $\Phi_\mu(f; g)$ has the form

$$(\bar{A} + \mu I)f = \bar{g} \tag{42}$$

Similar to (29), optimal solution is obtained by iteratively approaching g

$$f_{k+1} = f_k + \tau(\bar{g} - (\bar{A} + \mu I)f_k) \tag{43}$$

Thus, the algorithm of the projected Landweber method of constrained regularized solution is as follows

1. compute $\hat{f}_k^{(C,\mu)}(w)$ from $f_k^{(C,\mu)}(x)$
2. compute

$$\hat{h}_{k+1}^{(\mu)}(w) = \tau \hat{K}^*(w)\hat{g}(w) + (1 - \tau \left|\hat{K}(w)\right|^2 + \mu)\hat{f}_k^{(C,\mu)}(w) \tag{44}$$

3. compute $h_{k+1}^{(\mu)}(x)$ from $\hat{h}_{k+1}^{(\mu)}(w)$
4. compute $f_{k+1}^{(C,\mu)}(x) = (P_C h_{k+1}^{(\mu)})(x)$

We've introduced the van Cittert method, the Landweber method, and the projected Landweber method. Still, there's two algorithms we'd like to investigate: the steepest descent and the conjugate gradient method. The following content set a comparison between both methods. First of all, both methods aim to iteratively solve a large linear systems

$$\mathbf{A}f = \mathbf{g} \tag{45}$$

where $\mathbf{A}$ is a symmetric and positive definite matrix. Note that finding a solution $\mathbf{x}_{opt}$ to (45) is equivalent to minimize the below equation

$$\eta(f; \mathbf{g}) = \frac{1}{2} f^T \mathbf{A} f - f^T \mathbf{g} \tag{46}$$

The reason is that the minimum f is obtain by $\nabla\eta(f) = 0$, which is $\mathbf{A}f - \mathbf{g} = 0$ (i.e., the original problem). Now we can discuss the difference between these two approaches.

- *Steepest Descent*
  The idea of steepest descent is to move in the direction of gradient locally at each step. The local gradient of (46) is given by $\nabla_f \eta(f; \mathbf{g}) = \mathbf{A}f - \mathbf{g}$ which coincides with the negative residual $\mathbf{r} = \mathbf{g} - \mathbf{A}f$ of each step. The the general iteration is shown below

$$f_{k+1} = f_k + \tau r_k \tag{47}$$

  The subscript indicates at the k^{th} iteration. After the direction being made, the step size τ must be decided as well. By simple calculation and (46), the relation of consecutive steps follows

$$\eta(f_{k+1}; \mathbf{g}) = \eta(f_k; \mathbf{g}) + \frac{1}{2}\tau^2 \|\mathbf{A}r_k\|^2 - \tau \|r_k\|^2 \tag{48}$$

  Again the τ_k that minimizes $\eta(f_{k+1}; \mathbf{g})$ is obtain by

$$\nabla_\tau \eta(f_{k+1}; \mathbf{g}) = \frac{\|r_k\|^2}{\|\mathbf{A}r_k\|^2} \tag{49}$$

Recall that in image restoration, the problem we're solving is (28) where the symbol corresponds to $\mathbf{A}$ and $\mathbf{g}$ is $\bar{A}$ and $\bar{g}$ respectively. The resulting steepest descent is thus performed as a three-step iteration:

Initialize $\hat{f}_0(w) = 0$
Initialize $\hat{\bar{r}}_0(w) = \hat{K}^*(w)\hat{g}(w)$

for $i < iteration$ **do**

$$\tau_k = \frac{\|r_k\|^2}{\|\mathbf{A}r_k\|^2} \tag{50}$$

$$\hat{f}_{k+1}(w) = \hat{f}_k(w) + \tau_k \hat{\bar{r}}_k(w) \tag{51}$$

$$\hat{\bar{r}}_{k+1}(w) = \hat{\bar{r}}_k(w) - \tau \left|K(w)\right|^2 \hat{\bar{r}}_k \tag{52}$$

end for

output $f_{k+1} = inv\,\mathcal{F}(f_{k+1}(w))$

The final output is given by inverse Fourier transform. The reason why all calculations are done in Fourier domain is because of computation efficiency. In spatial domain, (52) needs two 2D convolution. However, in frequency domain, convolution becomes element-wise multiplication. It's worth mentioning that (52) can be derived from the definition of residual $\mathbf{r} = \mathbf{g} - \mathbf{A}f$ and (50). What's more, the inner product (matrix multiplication in this case) of two consecutive direct is zero

$$\langle r_{k+1}, r_k \rangle = \|r_k\|^2 - \tau_k \|\mathbf{A}r_k\|^2 = 0 \tag{53}$$

Below is a visualization of steepest descent where it starts from a point on the level surface of $\eta(f; \mathbf{g})$. At each step, it move through the gradient direction and reach the next minimum point along this direction.

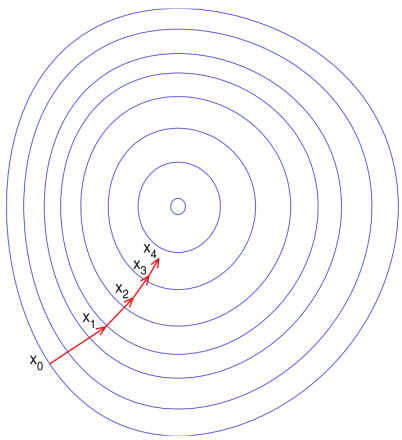

Figure 3.5: example from wiki

The result of steepest descent deblurring can is shown below. The images corresponding to each iteration are ordered from left to right and from top to bottom.

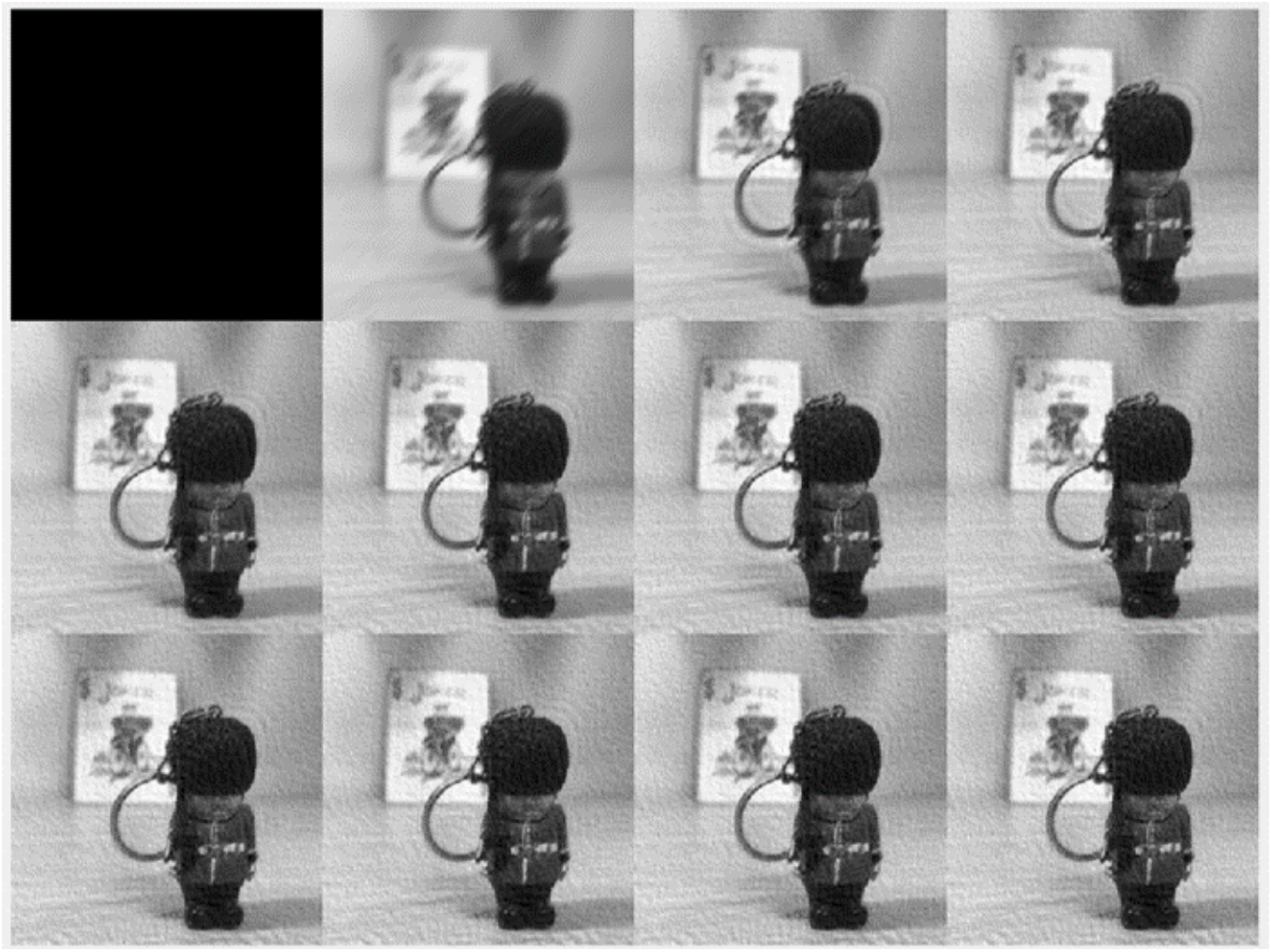

Figure 3.6: iteration outputs (from left to right and from top to bottom) 1^{st} *iteration* $\hat{f} = \mathbf{0}$, 2^{nd} *iteration*, 3^{rd} *iteration*...

One can find the optimal result is around the fifth and the seventh iteration. The later ones tend to experience ringing effects due to the absence of regularization. Similar to the steepest descent, the conjugate gradient method starts from a point on the objective function and move towards the optimal result. The difference is that the conjugate gradient method move in the direction of *conjugate gradient* rather that gradient.

- *Conjugate Gradient*
  Similar to the steepest descent method, the core philosophy of conjugate gradient is moving towards the optimal solution iteratively. The difference is that, conjugate gradient method moves in the directions that are *conjugate* to each other whereas steepest descent only makes sure that the directions are orthogonal to each other. Given two vectors $\mathbf{u}, \mathbf{v}$, we say that they are conjugate with respect to $\mathbf{A}$ if

$$\langle \mathbf{u}, \mathbf{A}\mathbf{v} \rangle = 0 \tag{54}$$

  Where $\langle , \rangle$ denotes the inner product in Hilbert space. Here we consider the case of vector inner product(i.e., $\langle \mathbf{u}, \mathbf{A}\mathbf{v} \rangle \triangleq \mathbf{u}^\top \mathbf{A}\mathbf{v}$) for simplicity. The object function for minimization is the same as in (46). Define residual

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}f_k \tag{55}$$

  Instead of moving the direction of gradient, we apply Gram-Schmidt method to ensure that each step $\mathbf{p}_k$s are *conjugate* to each other.

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{p}_i^T \mathbf{A}\mathbf{r}_k}{\mathbf{p}_i^T \mathbf{A}\mathbf{p}_i} \mathbf{p}_i \tag{56}$$

  The output of k^{th} iteration is thus

$$\begin{aligned} f_{k+1} &= f_k + \alpha_k \mathbf{p}_k \\ \alpha_k &= \frac{\mathbf{p}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} \end{aligned} \tag{57}$$

  The result of α_k is obtained by substituting f_{k+1} and f_k into (46) and minimizing with respect to α_k. The iterative conjugate gradient method is summarized as follows ($\mathbf{A}$ is replaced by $\bar{A}$ due to the assumption in (28))

  Initialize $r_0 = p_0 = g$
  Initialize $f_0 = 0$

  for $i < iteration$ **do**

$$\alpha_k = \frac{\|r_k\|^2}{p_k^T \bar{A} p_k} \tag{58}$$

$$r_{k+1} = r_k - \alpha_k \bar{A} p_k \tag{59}$$

$$\beta_k = -\frac{\|r_{k+1}\|^2}{\|r_k\|^2} \tag{60}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{61}$$

$$f_{k+1} = f_k + \alpha_k p_k \tag{62}$$

  end for

We make a final remark on the mathematical meaning of the conjugate gradient method. The coefficient α_ks and β_ks are chosen such that

$$\langle r_{k+1}, r_k \rangle = 0, \ \langle p_{k+1}, \bar{A} p_k \rangle = 0 \tag{63}$$

In linear algebra, r_ks form an orthogonal basis of Krylov subspace whereas p_ks form an $\bar{A}$-*orthogonal* basis of the same Krylov subspace. The following graph from Wikipedia visualize the difference between the steepest descent and the conjugate gradient method in a 2 dimensional case.

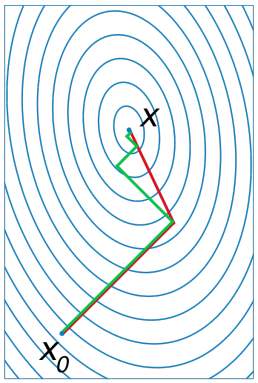

Figure 3.7: steepest descent(green) v.s conjugate gradient(red)

Last but not the least, although the above algorithm operates in spatial domain, the MATLAB implementation is in frequency domain as below. The hat symbol on top is the corresponding Fourier transform of each matrices. The detailed issues can be found in Appendix A.

Initialize $\hat{r}(w)_0 = \hat{p}(w)_0 = \hat{K}^*(w)\hat{g}(w)$
Initialize $\hat{f}(w)_0 = 0$

for $i < iteration$ **do**

$$\alpha_k = \frac{\int\limits_B |\hat{r}_k(w)|^2 dw}{\int\limits_B |\hat{K}^*(w)|^2 |\hat{p}(w)|^2 dw} \tag{64}$$

$$\hat{r}_{k+1}(w) = \hat{r}_k(w) - \alpha_k |\hat{K}(w)|^2 \hat{p}_k(w) \tag{65}$$

$$\beta_k = \frac{\int\limits_B |\hat{r}_{k+1}(w)|^2 dw}{\int\limits_B |\hat{r}_k(w)|^2 dw} \tag{66}$$

$$\hat{p}_{k+1}(w) = \hat{r}_{k+1}(w) + \beta_k \hat{p}_k(w) \tag{67}$$

$$\hat{f}_{k+1}(w) = \hat{f}_k(w) + \alpha_k \hat{p}_k(w) \tag{68}$$

end for

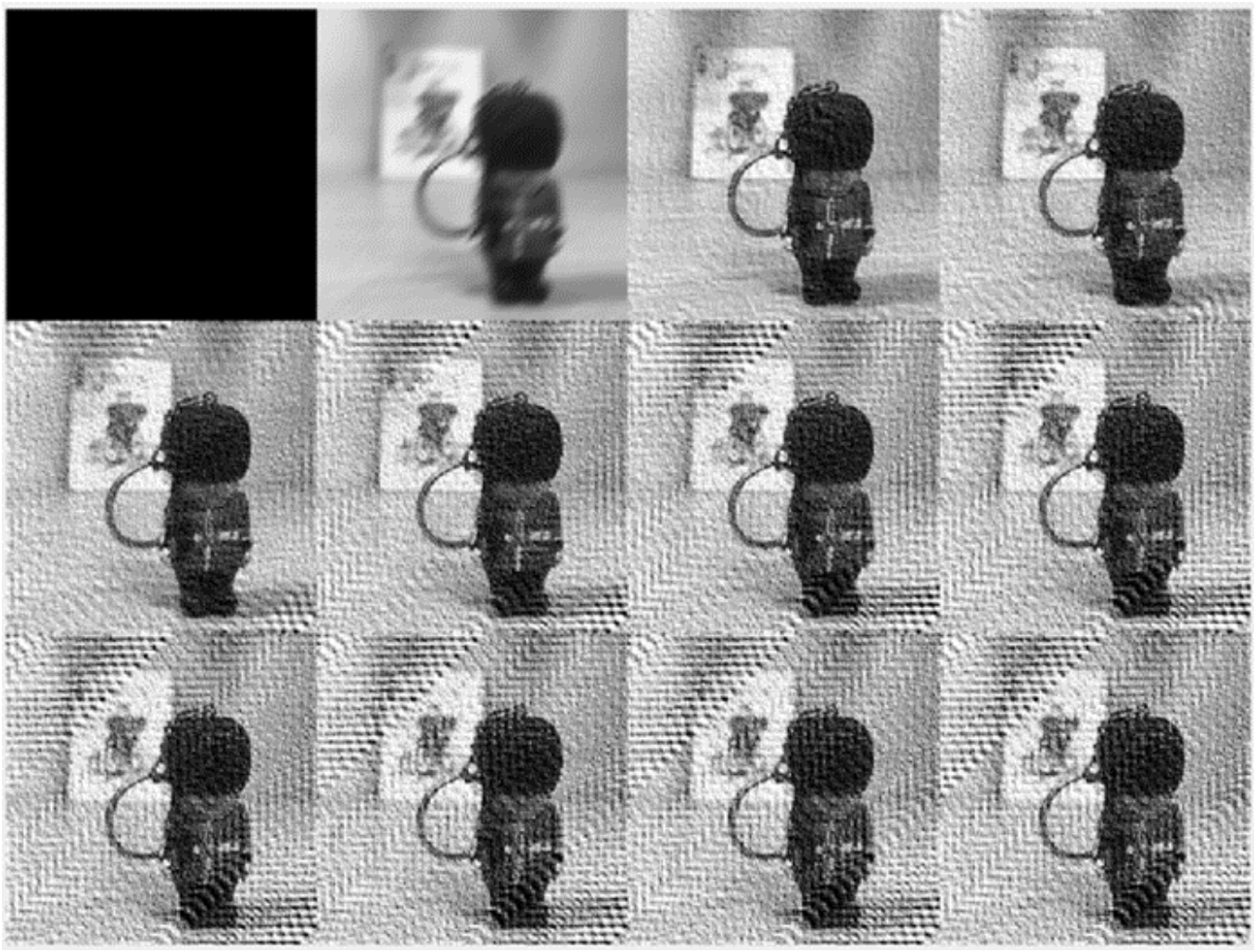

Figure 3.8: iteration outputs (from left to right and from top to bottom) 1^{st} *iteration* $\hat{f} = \mathbf{0}$, 11^{th} *iteration*, 21^{th} *iteration*...

The implementation result is shown above. Comparing the result to Figure 3.5, one can see that the conjugate gradient method converges faster than the steepest descent as it reaches visually optimal image at the third iteration. In the later iterations, the restored image experiences inevitable ringing effects.

3.3 Statistical

In this section, we're going to explore image deblurring methods based on statistical models. The concept of statistical modeling is that one can utilize some characteristics of the blurring process. If we assume that the latent image is deterministic, the *maximum likelihood*(ML) method can be adopted. If we model the latent image as a random process, the *Bayesian* method is used otherwise.

Starting with ML method, we rewrite equation (7) to $\boldsymbol{g} = \boldsymbol{A}f_0 + \boldsymbol{\omega}$, where f_0 and $\boldsymbol{A}$ are deterministic. $\boldsymbol{\omega}$ denotes random variable of additive noise. The resulting random vector of such random variable is

$$\boldsymbol{\eta}(\boldsymbol{f}) = \boldsymbol{A}\boldsymbol{f} + \boldsymbol{\nu} \tag{69}$$

$\boldsymbol{f}$ is the set of all possible latent image. Equation (69) implies that for any given set $\boldsymbol{f}$, the distribution of $\boldsymbol{\eta}(\boldsymbol{f})$ is available if $\boldsymbol{\omega}$ is also known. Thus, the density function of $\boldsymbol{\eta}(\boldsymbol{f})$ is $p_\eta(\boldsymbol{g}|\boldsymbol{f})$. The problem is to find the member f that maximizes $p_\eta(\boldsymbol{g}|\boldsymbol{f})$. Since $\boldsymbol{f}$ and $\boldsymbol{A}$ are both deterministic,

$$p_\eta(\boldsymbol{g}|\boldsymbol{f}) = p_{\boldsymbol{\nu}}(\boldsymbol{g} - \boldsymbol{A}\boldsymbol{f}) \tag{70}$$

(70) is the so-called *likelihood function*. The optimal solution is acquired by maximizing the likelihood function.

Here we discuss two types of noise: Gaussian noise and Poisson noise.

- *Gaussian noise:*
  The joint density of normally distributed random variable

  $$p_\eta(\boldsymbol{g}|\boldsymbol{f}) = \frac{1}{\sqrt{(2\pi)^{N^2}|\boldsymbol{S_v}|}} e^{\frac{-(\boldsymbol{S_v^{-1}}(\boldsymbol{g}-\boldsymbol{A}\boldsymbol{f}))\cdot(\boldsymbol{g}-\boldsymbol{A}\boldsymbol{f}))}{2}} \tag{71}$$

  $\boldsymbol{S_v}$ is the covariance matrix of $\boldsymbol{\nu}$. Taking the logarithm, the object function is then

  $$l(\boldsymbol{f}) = \frac{-(\boldsymbol{S_v^{-1}}(\boldsymbol{g} - \boldsymbol{A}\boldsymbol{f})) \cdot (\boldsymbol{g} - \boldsymbol{A}\boldsymbol{f}))}{2} - \frac{1}{2} ln[(2\pi)^{N^2}|\boldsymbol{S_v}|] \tag{72}$$

  Maximizing (72) is equivalent to minimizing the *discrepancy*

  $$\varepsilon_\nu^2(\boldsymbol{f};\boldsymbol{g}) = (\boldsymbol{S_v^{-1}}(\boldsymbol{g} - \boldsymbol{A}\boldsymbol{f})) \cdot (\boldsymbol{g} - \boldsymbol{A}\boldsymbol{f})) \tag{73}$$

  The minimum value is obtained by setting the differential of (73) equals to zero. Here we omit detailed derivation and show the result directly

  $$\boldsymbol{A^*S_v^{-1}A}\boldsymbol{f} = \boldsymbol{A^*S_v^{-1}}\boldsymbol{g} \tag{74}$$

  (74) is the generalized equation of (27). In the case that $\boldsymbol{S_v}$ is multiples of identity matrix(i.e., the noise is uncorrelated), (74) becomes (27). That is to say, *For stationary Gaussian noise, the maximum-likelihood method will acquire the same result as the least-squares method.* This is based on the fact that both Gaussian noise and the least-squares method utilize the $\|\cdot\|^2$ feature of the image.

- *Poisson noise:*
  The additive noise follows Poisson distribution

$$p_\eta(\boldsymbol{g}|\boldsymbol{f}) = \prod_{m=1}^{N^2} e^{-(\boldsymbol{Af})_m} \frac{(\boldsymbol{Af})_m^{g_m}}{(g_m)!} \tag{75}$$

  Taking the logarithm, the object function is then

$$l(\boldsymbol{f}) = \sum_{m=1}^{N^2} \{g_m ln(\boldsymbol{Af})_m - (\boldsymbol{Af})_m - ln(g_m)!\} \tag{76}$$

  We tries to find the maximum value by differentiating (76)

$$\begin{aligned} (\nabla l)_n(\boldsymbol{f}) = \frac{\partial l(\boldsymbol{f})}{\partial f_n} &= -\alpha_n + (\boldsymbol{A^T} \frac{\boldsymbol{g}}{\boldsymbol{Af}})_n \\ \alpha_n &= \sum_{m=1}^{N^2} A_{m,n} \end{aligned} \tag{77}$$

Equation (77) relates to the Richardson-Lucy method of image deblurring. However, the explicit derivation is beyond the scope of this tutorial. Here we give some intuitive understanding of Richardson-Lucy method for simplicity. Firstly, instead of applying $-\alpha_n + (\boldsymbol{A^T} \frac{\boldsymbol{g}}{\boldsymbol{Af}})_n = 0$ directly, we add two constraints

$$\begin{aligned} f_n \frac{\partial l(\boldsymbol{f})}{\partial f_n}|_{f=\tilde{f}} = 0; \quad n = 1, 2..., N^2 \\ \frac{\partial l(\boldsymbol{f})}{\partial f_n}|_{f=\tilde{f}} \leq 0; \quad if\ \tilde{f}_n = 0 \end{aligned} \tag{78}$$

A visualization is provided in the figure below. The idea is that the maximum value must be non-negative. Therefore, the first equation implies that either the gradient or $\tilde{f}_n$ itself equals zero. The second inequality implies that if the obtained result is zero, its slope(blue line) must be less than zero so that other positive numbers will result in smaller values.

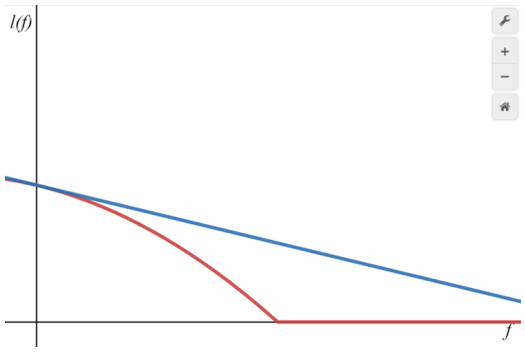

Figure 3.9: slope(blue line), two dimension objective curve(red line)

By Kuhn-Tucker Conditions, one can ensure that the above two equation can be fulfilled. Additionally, the first equation in (78) is the necessary and sufficient condition for the result in (77). The Richardson-Lucy method combines (77) and (78) as

$$\alpha_n \tilde{f}_n = \tilde{f}_n(\boldsymbol{A^T}\frac{\boldsymbol{g}}{\boldsymbol{Af}})_n; \quad n = 1, 2, ..., N^2 \tag{79}$$

Applying successful approximations, (79) is modified to operate on 2D image

$$(\tilde{f}_{k+1})_{m,n} = (\tilde{f}_k)_{m,n}(\mathsf{K}^\mathsf{T} * \frac{g}{\mathsf{K} * \tilde{f}_k})_{m,n} \tag{80}$$

As one may notice that α_n disappears in (80). This is because Richardson-Lucy assume that $\alpha_n = \sum_{m=1}^{N^2} A_{m,n} = 1$. The result of Richardson-Lucy is shown below.

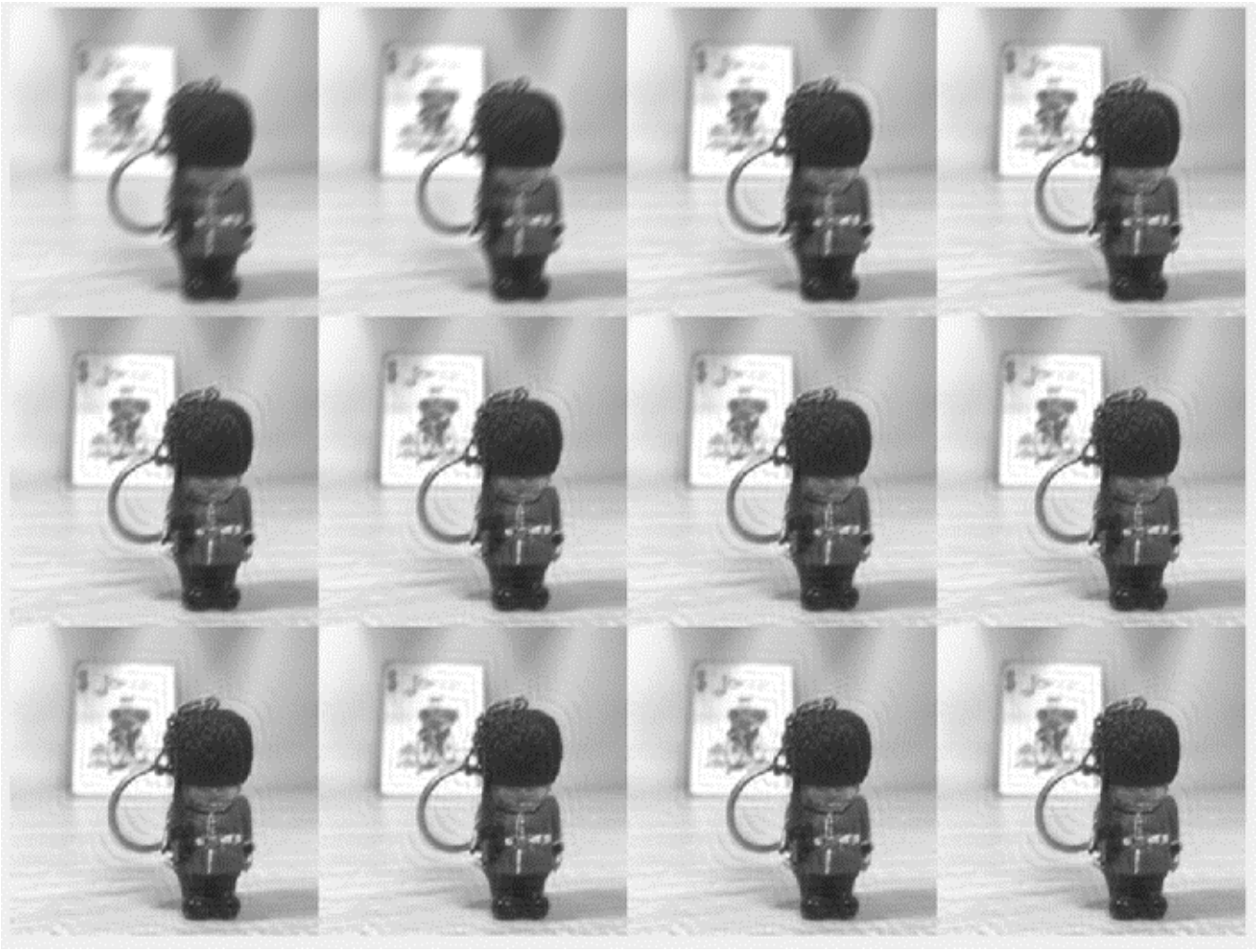

Figure 3.10 Iteration Outputs (from left to right and from top to bottom) 1^{st} *iteration*, 11^{th} *iteration*, 21^{th} *iteration*...

Now, let's move to the case where latent image is *not deterministic*. Comparing with equation (69), the latent image is now a random vector $\boldsymbol{\phi}$ with known density function.

$$\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{\phi} + \boldsymbol{\nu} \tag{81}$$

To analyse the probability distribution, we use Bayes' theorem

$$\begin{aligned} p_\phi(\boldsymbol{f}|\boldsymbol{g}) &= \frac{p_\eta(\boldsymbol{g}|\boldsymbol{f})p_\phi(\boldsymbol{f})}{p_\eta(\boldsymbol{g})} \\ &= \frac{p_\eta(\boldsymbol{g}|\boldsymbol{f})p_\phi(\boldsymbol{f})}{\int p_\eta(\boldsymbol{g}|\boldsymbol{f})p_\phi(\boldsymbol{f})\, d\boldsymbol{f}} \end{aligned} \tag{82}$$

In the above equation, $p_\phi(\boldsymbol{f})$ is called *a priori* information whereas $p_\phi(\boldsymbol{f}|\boldsymbol{g})$ denotes *a posteriori* density function. If both $p_\eta(\boldsymbol{g}|\boldsymbol{f})$ and $p_\phi(\boldsymbol{f})$ are known, one can have the complete knowledge of $p_\phi(\boldsymbol{f}|\boldsymbol{g})$ and $p_{\phi\eta}(\boldsymbol{f}, \boldsymbol{g})$. The restored image can be obtained either by maximizing $p_\phi(\boldsymbol{f}|\boldsymbol{g})$ or by the *a posteriori* expectation $\hat{\boldsymbol{f}} = E\{\boldsymbol{\phi}\boldsymbol{\eta}\} = \int \boldsymbol{f} p_\phi(\boldsymbol{f}|\boldsymbol{g}) d\boldsymbol{f}$. However, there are two main challenges of Bayesian methods. First challenge is that it is computational expensive. Second challenge is the choice of $p_\phi(\boldsymbol{f})$. By definition, $p_\phi(\boldsymbol{f})$ means the probability of a given image $\boldsymbol{f}$. The higher the probability is, the more $\boldsymbol{f}$ is possible to be a good image. In other words, one have to find a probability model that can evaluate natural image accurately. With that being said, in resent years, some good probability model $p_\phi(\boldsymbol{f})$s were proposed based on Bayesian method which will discuss in the next chapter. Before then, we show that the famous Wiener filter is a special case of Bayesian method.

To begin with, four assumptions were made:

1. the additive noise is a Gaussian random vector with zero mean and covariance matrix $\boldsymbol{S_\nu}$. Thus,

$$p_\nu(\boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^{N^2}|\boldsymbol{S_\nu}|}} e^{\frac{-(\boldsymbol{S_\nu}^{-1}\boldsymbol{w}\cdot\boldsymbol{w})}{2}} \tag{83}$$

2. object $\boldsymbol{\phi}$ in (81) is also a Gaussian random vector with zero expectation value and covariance matrix $\boldsymbol{S_\phi}$. Thus,

$$p_\phi(\boldsymbol{f}) = \frac{1}{\sqrt{(2\pi)^{N^2}|\boldsymbol{S_\phi}|}} e^{\frac{-(\boldsymbol{S_\phi}^{-1}\boldsymbol{f}\cdot\boldsymbol{f})}{2}} \tag{84}$$

3. According to equation (81), it can be derived that $E\{\eta\} = \boldsymbol{0}$ and $\boldsymbol{S_\eta} = \boldsymbol{A}\boldsymbol{S_\phi}\boldsymbol{A}^* + \boldsymbol{S_\nu}$(please refer to the book for detailed derivation). Additionally, the independence of $\boldsymbol{\nu}$ and $\boldsymbol{phi}$ give the resulting probability density

$$p_\eta(\boldsymbol{g}) = \frac{1}{\sqrt{(2\pi)^{N^2}|\boldsymbol{S_\eta}|}} e^{\frac{-(\boldsymbol{S_\eta}^{-1}\boldsymbol{g}\cdot\boldsymbol{g})}{2}} \tag{85}$$

4. noise $\boldsymbol{\nu}$ and the object $\boldsymbol{\phi}$ are independent so that $(\boldsymbol{S}_\phi)_{n,m} = E\{\phi_n \nu_m^*\} = 0$. Thus,

$$\begin{aligned} p_{\phi\eta}(\boldsymbol{f},\boldsymbol{g}) &= p_\nu(\boldsymbol{g}-\boldsymbol{A}\boldsymbol{f})p_\phi(\boldsymbol{f}) \\ &= \frac{1}{\sqrt{(2\pi)^{2N^2}|\boldsymbol{S_\nu}||\boldsymbol{S_\phi}|}} e^{\frac{-\Phi(\boldsymbol{f},\boldsymbol{g})}{2}} \end{aligned} \tag{86}$$

$$\Phi(\boldsymbol{f},\boldsymbol{g}) = (\boldsymbol{S_\nu^{-1}}(\boldsymbol{g}-\boldsymbol{A}\boldsymbol{f}))\cdot(\boldsymbol{g}-\boldsymbol{A}\boldsymbol{f})) + (\boldsymbol{S_\phi^{-1}}\boldsymbol{f}\cdot\boldsymbol{f}) \tag{87}$$

With the four premises, the target object function according to (82) is

$$p_\phi(\boldsymbol{f}|\boldsymbol{g}) = \frac{1}{\sqrt{(2\pi)^{N^2}\frac{|\boldsymbol{S_\nu}||\boldsymbol{S_\phi}|}{|\boldsymbol{S_\eta}|}}} e^{\frac{-\Phi(\boldsymbol{f},\boldsymbol{g})+(\boldsymbol{S_\eta^{-1}}\boldsymbol{g}\cdot\boldsymbol{g})}{2}} \tag{88}$$

The above conditional probability is also a Gaussian distribution. Thus equation (88) can be rewrite as

$$p_\phi(\boldsymbol{f}|\boldsymbol{g}) = \frac{1}{\sqrt{(2\pi)^{N^2}|\boldsymbol{S_{\phi|\eta}}|}} e^{\frac{-(\boldsymbol{S_{\phi|\eta}^{-1}}(\boldsymbol{f}-\hat{\boldsymbol{f}})\cdot(\boldsymbol{f}-\hat{\boldsymbol{f}}))}{2}} \tag{89}$$

where $\boldsymbol{S_{\phi|\eta}}$ is the *a posteriori* covariance matrix and $\hat{\boldsymbol{f}}$ is the maximum value. By equating (88) and (89), we get

$$\boldsymbol{S_{\phi|\eta}} = (\boldsymbol{A^* S_\nu^{-1} A} + \boldsymbol{S_\phi^{-1}})^{-1} \tag{90}$$

One can see that maximizing (88) (89) is equal to minimizing $\Phi(\boldsymbol{f},\boldsymbol{g})$ in (87). The corresponding derivative with respect to $\boldsymbol{f}$ is

$$\begin{aligned} (\boldsymbol{A^* S_\nu^{-1} A} + \boldsymbol{S_\phi^{-1}})\hat{\boldsymbol{f}} &= \boldsymbol{A^* S_\nu^{-1} g} \\ \hat{\boldsymbol{f}} &= \boldsymbol{S_{\phi|\eta} A^* S_\nu^{-1} g} \end{aligned} \tag{91}$$

The restored image is obtained by applying a matrix $\boldsymbol{R_0}$ on the blurred image $\boldsymbol{g}$.

$$\boldsymbol{R_0} = \boldsymbol{S_{\phi|\eta} A^* S_\nu^{-1}} = (\boldsymbol{A^* S_\nu^{-1} A} + \boldsymbol{S_\phi^{-1}})^{-1}\boldsymbol{A^* S_\nu^{-1}} = \boldsymbol{S_\phi A^*}(\boldsymbol{A S_\phi A^*} + \boldsymbol{S_\nu})^{-1} \tag{92}$$

Equation (92) is the exactly the Wiener filter. Finally, if we further model the Gaussian distribution in assumption as white process, $\boldsymbol{S_\nu} = \varepsilon^2\boldsymbol{I}$ and $\boldsymbol{S_\phi} = E^2\boldsymbol{I}$.

$$\boldsymbol{R_0} = (\boldsymbol{A^* A} + (\frac{\boldsymbol{\varepsilon}}{\boldsymbol{E}})^{\boldsymbol{2}}\boldsymbol{I})^{-\boldsymbol{1}}\boldsymbol{A^*} \tag{93}$$

The result can be regarded as a special case of Tikhonov regularization method comparing to equation (25).

We show the result in the following figure. The left image is obtained by the MATLAB built-in function `deconvwnr`. The middle and right images are obtained by passing μ_{Miller} and μ_L to the regularization method implemented before respectively.

Figure 3.11: A simple comparison

As one may know, the main objective of Wiener filter is to minimize the overall mean square error in the process of inverse filtering. The problem is that, the signal-to-noise(noise-to-signal) ratio is not available in many cases. In contrast, the L curve method is a method of exhaustion which takes more time to search for a good μ.

3.4 Chapter Summarize

In this chapter, we discuss methods to restore blurred image caused by *space-invariant* kernel and additive noise. The methods can be approximately categorize into three groups: regularization, iterative regularization, and statistical. The regularization methods can be seen as the filtering of a blurred image whereas the iterative regularization methods provides multiple results in deblurring process which is more flexible comparing to regularization methods. On the other hand, statistical methods explore the characteristics of additive noise. The following table summarize the algorithms in this chapter.

	Name	**Algorithm**	**MATLAB implementation**
one-step	Tikhonov regularization	$f_\mu = (A^*A + \mu I)^{-1}A^*g$	`Regularization_Function.m`
one-step	Wiener filter	$f_\mu = (A^*A + (\frac{\varepsilon}{E})^2 I)^{-1}A^*g$	`Wiener_method.m`
iterative	steepest descent	(50)∼(52)	`steepest_descent_Function.m`
iterative	conjugate gradient	(58)∼(62)	`CG_Function.m`
iterative	Richardson Lucy	$(f_{k+1}) = (f_k)(\mathsf{K}^\mathsf{T} * \frac{g}{\mathsf{K} * f_k})$	`RL_Function.m`

$*$_Function.m files in the folder are the main implementation of corresponding algorithm. On the other hand, $*$_method.m files are scripts controlling input variables, blurring process and plotting. In the next chapter, we will focus on the recent progress of image deblurring containing both blind deblurring and non-blind deblurring.

Uniform Deblurring

In this chapter, several blind and non-blind image deblurring methods is introduced. We will introduce several iconic methods that have outstanding performances. In fact, most of the processes have inherited the concepts from the former chapter with additional prior information of images.

4.1 Non-blind Deblurring

4.1.1 Fast Image Deconvolution using Hyper-Laplacian Priors [7] (code path: Deblurring_Gather/fastdeconv/test_fast_deconv.m)

In this paper, author inherit the concept of Bayesian method and propose a $p_\phi(\boldsymbol{f})$ which will be denoted as $p(\boldsymbol{x})$ in the following discussion.Following (82), the conditional probability can be generalized as

$$p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{k}) \propto p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{k})p(\boldsymbol{x}) \tag{94}$$

where $\boldsymbol{x}$ is the latent image, $\boldsymbol{k}$ is the known kernel, and $\boldsymbol{y}$ is the input blurred image. The authors found that the gradient of sharp images are likely to follow the *Hyper-Laplacian* distribution.

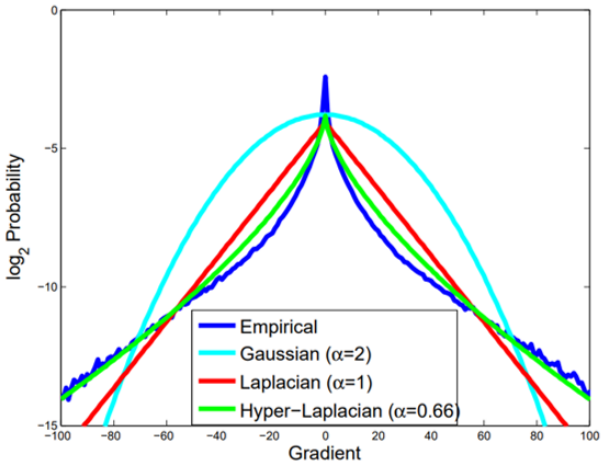

Figure 4.1: empirical image gradient versus different distribution models

One can see from the above figure that Hyper-Laplacian introduce a variable α to control the distribution curve. In this paper, $\alpha = \frac{1}{2}, \frac{2}{3}$ are discussed. Following this assumption, maximizing the conditional probability $p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{k})$ is equal to minimizing the cost of $-\log p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{k}) = -\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{k}) - \log p(\boldsymbol{x})$ which gives the objective function

$$\min_{\boldsymbol{x}} \sum_{i=1}^{N} (\frac{\lambda}{2}(\boldsymbol{x} \oplus \boldsymbol{k} - \boldsymbol{y})_i^2 + \sum_{j=1}^{J} |(\boldsymbol{x} \oplus f_j)_i|^\alpha) \tag{95}$$

where $\oplus$ is the two dimensional convolution operator for image gradient(i.e., $f_1 = [1-1],\ f_2 = [1-1]^T$) and the corresponding matrix representation is $F_i^j \boldsymbol{x} = (\boldsymbol{x} \oplus f_j)_i$. λ is the weighting value controlling the strength of regularization. N is the number of total pixel(i being pixel index). $J = 2$ in the case of 2D image.

Having α as the exponential term, it is rather hard to solve (95) by taking partial derivative according to $\boldsymbol{x}$. Using the *half-quadratic* method, we bring in two *auxiliary variables* w_i^1 and w_i^2 to each pixel so that $\boldsymbol{x}$ will no longer be in $|.|^\alpha$. The new objective function is then

$$\min_{\boldsymbol{x},\boldsymbol{w}} \sum_{i=1}^{N} (\frac{\lambda}{2}(\boldsymbol{x} \oplus \boldsymbol{k} - \boldsymbol{y})_i^2 + \frac{\beta}{2}(\|F_i^1 \boldsymbol{x} - w_i^1\|_2^2 + \|F_i^2 \boldsymbol{x} - w_i^2\|_2^2) + |w_i^1|^\alpha + |w_i^2|^\alpha) \tag{96}$$

The basic idea is that, when $\beta \to \inf$ (96) will approach (95) since w_i must equal to $F_i \boldsymbol{x}$ to lower the effect of β. That is to say, the entire algorithm is to solve (96) with an increasing β iteratively.

Since we're minimizing two variables $\boldsymbol{x}$ and $\boldsymbol{w}$, the optimization process are separated into two sub-problems.

- $\boldsymbol{x}$ *sub-problem:*
  $\boldsymbol{x}$ sub-problem solves $\boldsymbol{x}$ by fixing $\boldsymbol{w}$. The derivative with respect to $\boldsymbol{x}$ is given as

  $$({F^1}^T F^1 + {F^2}^T F^2 + \frac{\lambda}{\beta} K^T K)\boldsymbol{x} = {F^1}^T \boldsymbol{w}^1 + {F^2}^T \boldsymbol{w}^2 + \frac{\lambda}{\beta} K^T \boldsymbol{y} \tag{97}$$

  where $K\boldsymbol{x} = \boldsymbol{x} \oplus \boldsymbol{k}$. Applying 2D FFT to (97), the result of restored $\boldsymbol{x}$ is given as

  $$\boldsymbol{x} = \mathcal{F}^{-1}(\frac{\mathcal{F}(F^1)^* \circ \mathcal{F}(\boldsymbol{w}^1) + \mathcal{F}(F^2)^* \circ \mathcal{F}(\boldsymbol{w}^2) + (\lambda/\beta)\mathcal{F}(K)^* \circ \mathcal{F}(y)}{\mathcal{F}(F^1)^* \circ \mathcal{F}(F^1) + \mathcal{F}(F^2)^* \circ \mathcal{F}(F^2) + (\lambda/\beta)\mathcal{F}(K)^* \circ \mathcal{F}(K)}) \tag{98}$$

  In the above equation, $*$ denotes complex conjugate. $\circ$ denotes element-wise multiplication and the division is also element-wise.

- $\boldsymbol{w}$ *sub-problem:*
  $\boldsymbol{w}$ sub-problem, on the other hand, is fixing $\boldsymbol{x}$ and finding the minimal value. Therefore,

  $$w_{opt} = \arg\min_{w} |w|^\alpha + \frac{\beta}{2}(w - v)^2 \tag{99}$$

  where $v = F_i^j \boldsymbol{x}$. One can see that the complexity of equation (99) depends on α. Here the authors discuss two methods to find w_{opt}. The first one is rather straightforward. Given α, w_{opt} depends only on β and v. In such case, a look-up table is used (β ranges from 1 to 256, and v ranges from -0.6 to 0.6). A look-up table allow users to obtain w_{opt} really fast. For example, it

only takes 1.5 seconds to restore the 1024x1024 blurred image below. However, a look-up table is just an approximation. The authors also discuss the possible analytic solution. For example, if $\alpha = \frac{1}{2}$, the minimal value in (99) is given as

$$w^3 - 2vw^2v^2w - sign(v)/4\beta^2 = 0 \tag{100}$$

There are three possible combinations of roots in (100): (a) 3 imaginary roots, (b) 2 imaginary roots and 1 real root $\tilde{w}$, and (c) 3 real roots. w_{opt} in each case are: (a) $w_{opt} = 0$, (b) $w_{opt} = \tilde{w} \quad if \quad 2v/3 < \tilde{w} < v \quad else \quad 0$ (c) the root that lies between 0 and v, and further from 0. The paper provide the explicit derivation of analytic solutions of $\alpha = 1/2$ and $\alpha = 2/3$. Nevertheless, MATLAB implementation uses look-up table for faster performance. One can refer to the paper if is interested. The complete algorithm and result is shown below.

Algorithm 1 Fast image deconvolution using hyper-Laplacian priors

Require: Blurred image $\mathbf{y}$, kernel $\mathbf{k}$, regularization weight λ, exponent α (¿0)
Require: β regime parameters: $\beta_0, \beta_{\text{Inc}}, \beta_{\text{Max}}$
Require: Number of inner iterations T.
1: $\beta = \beta_0$, $\mathbf{x} = \mathbf{y}$
2: Precompute constant terms in Eqn. 4.
3: **while** $\beta < \beta_{\text{Max}}$ **do**
4: $iter = 0$
5: **for** $i = 1$ to T **do**
6: Given $\mathbf{x}$, solve Eqn. 5 for all pixels using a LUT to give $\mathbf{w}$
7: Given $\mathbf{w}$, solve Eqn. 4 to give $\mathbf{x}$
8: **end for**
9: $\beta = \beta_{\text{Inc}} \cdot \beta$
10: **end while**
11: **return** Deconvolved image $\mathbf{x}$

Figure 4.2: Eqn.5:(99), Eqn.4:(98)

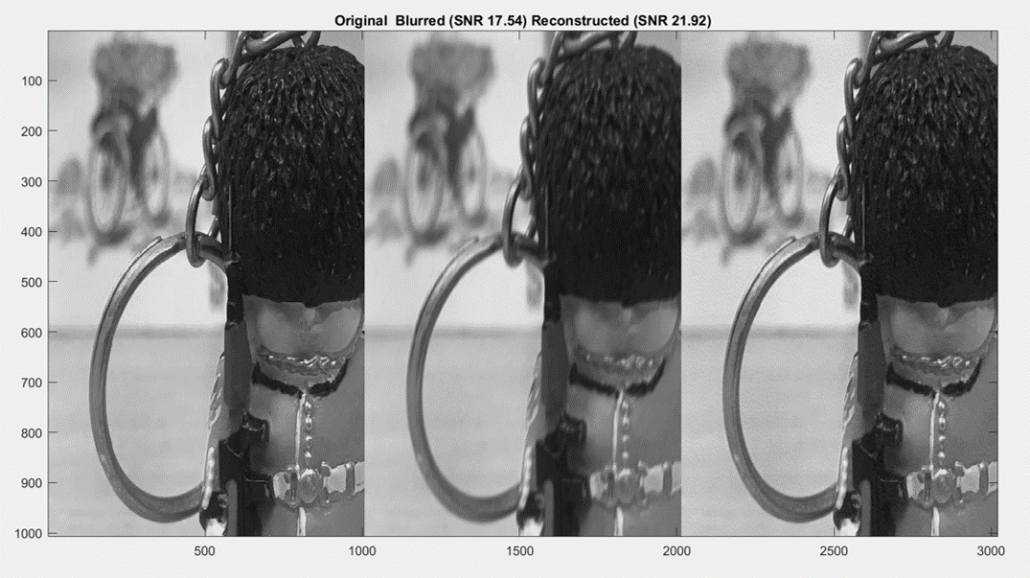

Figure 4.3: result of [7]

4.1.2 A General Framework for Regularized, Similarity-Based Image Restoration [6] (code path: Deblurring_Gather/DeblurringCode/mainDemo.m)

Different from other methods, here we consider image as weighted graph $G = (V, E, K)$ consisting of a finite set V of vertices(image pixels) and a finite set E of edges (i, j) with the corresponding weights $K(i, j)$ which measure similarity between vertices (pixels) i and j in the graph. The values of the image can be denoted as a vector $\boldsymbol{z} = [z(1), \cdots, z(N)]^T$. Now, define the difference on an edge $(i, j) \in E$ of the graph G is

$$(dz)(i, j) = \sqrt{K(i, j)}(z(j) - z(i)) \tag{101}$$

The point is to minimize the overall differences in the graph G. Therefore, the regularization term can be written as

$$R(\boldsymbol{z}) = \frac{1}{2}\sum_{i=1}^{N}\|\nabla z(i)\|^2 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j,j\sim i} K(i, j)(z(i) - z(j))^2 \tag{102}$$

and the corresponding objective function is thus

$$E(\boldsymbol{z}) = \|\boldsymbol{y} - A\boldsymbol{z}\|^2 + \eta R(\boldsymbol{z}) \tag{103}$$

A being the blurring matrix. Since most of the mathematical knowledges involved in this paper are beyond the scope of this tutorial, we will explore the algorithm in a top-down manner without going into too much mathematical details. The overall algorithm is as below

Algorithm 1 Iterative Restoration Algorithm

Inputs: blurred noisy image $\mathbf{y}$, blurring matrix A
Output: deblurred estimate $\mathbf{z}^\star$
Initializations:
1. Estimate the noise standard deviation $\hat{\sigma}$ in $\mathbf{y}$ using algorithm in [43].
2. Denoise $\mathbf{y}$ using denoising algorithm in [16] to derive $\hat{\mathbf{z}}^{(0)}$.
3. Set $k = 0$.
while not converged **do**
- Compute K from $\hat{\mathbf{z}}^{(k)}$ using (a)
- Apply Sinkhorn algorithm in [30] to K to get the diagonal matrix C.
- Compute the filtering matrix as $W = C^{-1/2}KC^{-1/2}$.
- Solve objective function in (b) using CG to compute $\hat{\mathbf{z}}^{(k+1)}$.
- Set $\mathbf{z}^\star = \hat{\mathbf{z}}^{(k+1)}$, and $k = k + 1$.
end while
return $\mathbf{z}^\star$

Figure 4.4: complete algorithm

Firstly, noise estimation and noise removal are performed on the blurred image y. The first step of the iteration is constructing weight matrix K

$$\text{(a): } K(i,j) = e^{\frac{-\|\hat{\boldsymbol{z}}_i - \hat{\boldsymbol{z}}_j\|^2}{h^2}} \tag{104}$$

where are patches around pixel i and j of the image $\hat{\boldsymbol{z}}$, and h is a smoothing parameter. This operation as be seen as evaluating the similarity $K(i,j)$ between patch centered at i and j. If these two patch are similar, $K(i,j)$ is large, forcing pixel i and pixel j close to each other in the reconstructed image.

The second step of the iteration comes from the fact that

$$R(\boldsymbol{z}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j,j\sim i} K(i,j)\left(\frac{z(i)}{\sqrt{C(i,i)}} - \frac{z(j)}{\sqrt{C(j,j)}}\right)^2 = \boldsymbol{z}^T\left(I - C^{\frac{-1}{2}}KC^{\frac{-1}{2}}\right)\boldsymbol{z} \tag{105}$$

which is called *normalized graph Laplacian*. In fact other types of graph Laplacian are discussed in the paper as well. Please refer to the paper for more detail.

Reference	Graph Laplacian	Symmetric	DC eigenvector	Stochastic property
[23], [25]	$D-K$	Yes	Yes	No
[27]	$I-D^{-1/2}KD^{-1/2}$	Yes	No	No
[28], [29]	$I-D^{-1}K$	No	Yes	$D^{-1}K$ is row-stochastic
ours	$\mathbf{I-C^{-1/2}KC^{-1/2}}$	**Yes**	**Yes**	**W is doubly-stochastic**

Figure 4.5: properties of different graph Laplacians

After obtaining the diagonal matrix C, the filtering matrix W is computed by $W = I - C^{\frac{-1}{2}}KC^{\frac{-1}{2}}$. Recall that the objective function at the beginning becomes

$$E(\boldsymbol{z}) = (\boldsymbol{y} - A\boldsymbol{z})^T\{I + \beta(I - W)\}(\boldsymbol{y} - A\boldsymbol{z}) + \eta\boldsymbol{z}^T(I - W)\boldsymbol{z} \tag{106}$$

where $\beta \geq -1$ and $\eta > 0$ are the parameters to be tuned based on noise and blur. The minimum value can be obtained by setting the gradient of objective function equals to 0, giving

$$\text{(b): } (A^T\{I + \beta(I - W)\}A + \eta(I - W))\boldsymbol{z} = A^T\{I + \beta(I - W)\}\boldsymbol{y} \tag{107}$$

Since the linear equation is in the form of $\bar{A}\boldsymbol{x} = \boldsymbol{b}$, it can be solved by the conjugate gradient method(section 3.2) to get new estimation $\hat{\boldsymbol{z}}^{(k+1)}$ for the next iteration. The overall algorithm can be simplify as the flow chart below

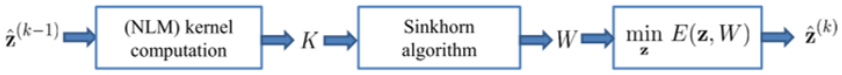

Figure 4.6: flow char of the proposed algorithm

One may notice that the first term in (b) has an additional $\{I + \beta(I - W)\}$ comparing to the cost function at the beginning. There are two advantages discussed in the paper. Firstly, since the matrix $(I - W)$ is a high-pass filter, with $\beta > 0$, $I + \beta(I - W)$ behaves like a sharpening filter on the residuals $\boldsymbol{y} - A\boldsymbol{z}$. Secondly, the solution in (b) can be represent as

$$\begin{aligned} \hat{\boldsymbol{z}} =& F(A, W)A^T\{I + \beta(I - W)\}\boldsymbol{y} \\ & F(A, W) = \{A^T\{I + \beta(I - W)\}A + \eta(I - W)\}^{-1} \end{aligned} \tag{108}$$

Consider the inverse operation in the above equation, experiments have shown that $F(A, W)$ has much less conditional numbers than that of A^TA, resulting a solution with less errors.

A sample result

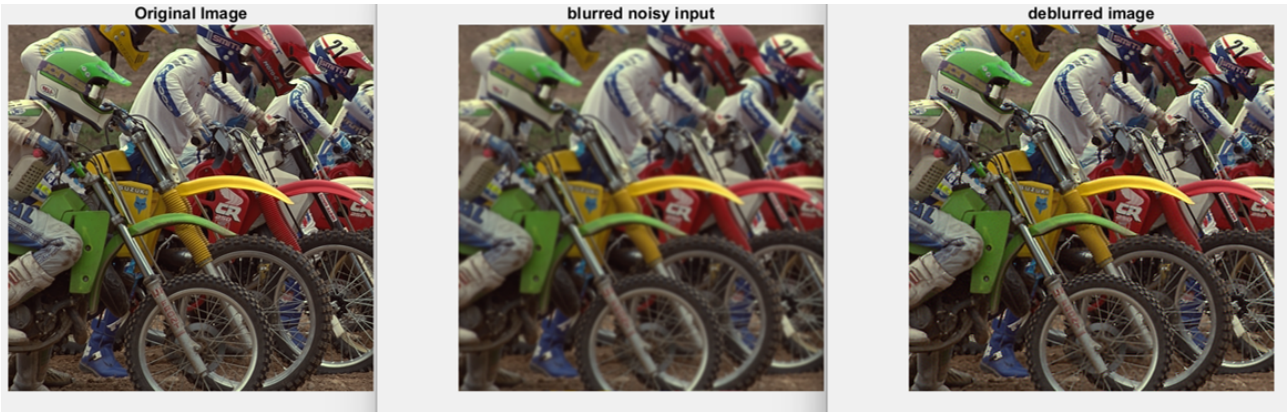

Figure 4.7: sample result

4.1.3 Image Deblurring Using a Pyramid-Based Richardson–Lucy Algorithm [3]

This method is a modified version of the method introduced in section 4.1.2.

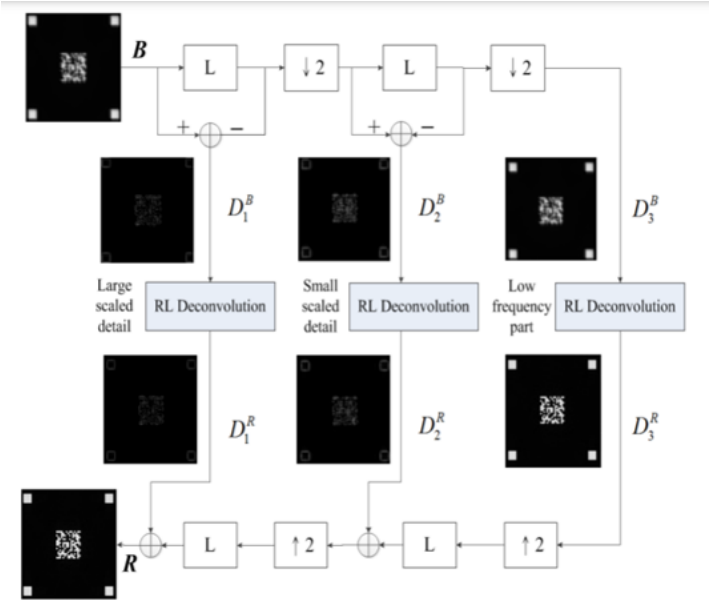

Figure 4.8: three-layer pyramid structure with RL deconvolution

Since the Richardson-Lucy algorithm has been already fully described in section 3.3, we will focus on the structure of the this algorithm. Given a known kernel K, the blurred image is down sampled to three level and then reconstructed in a coarse-to-fine manner. At the top half of the structure, the residual of the blurred image before and after convolving the low pass filter L is computed.

$$\begin{aligned} D_1^B &= B - B * L \\ D_2^B &= (B * L)_{\downarrow 2} - (B * L)_{\downarrow 2} * L \\ D_3^B &= [(B * L)_{\downarrow 2} * L]_{\downarrow 2} \end{aligned} \tag{109}$$

and

$$D_i^R = \text{Richardson-Lucy}(D_i^B) \text{ for } i =, 1, 2, 3 \tag{110}$$

The bottom half of the structure then reconstruct the estimated image R as

$$R = \{[(D_3^R)_{\uparrow 2} * L] + D_2^R\}_{\uparrow 2} * L + D_1^R \tag{111}$$

Note that Richardson-Lucy algorithm is performed on the residual of the blurred image instead of the original blurred image to avoid Gibbs phenomena. This is because the residual of an image tend

to have weaker amplitude comparing to the original image. Additionally, the reconstruction of first level D_1^R can be obtained by direct convolution of inverse kernel W and D_1^B to preserve large scale details.(i.e., $D_1^R = D_1^B * W$, where $W = \mathcal{F}^{-1}(\frac{1}{\mathcal{F}(K)})$).

4.2 Blind Deblurring

In this chapter, we discuss a few methods when blur kernel is not available(i.e., $k(x)$ and $f(x)$ in equation (2) are both unknown). Intuitively, if there are no other constrains on $k(x)$ and $f(x)$, this problem is impossible to solve since there are infinite pairs of latent images and blur kernels in equation (2). With that being said, one must utilize image characteristics as constrains so that the restored image conforms to specific feature.

4.2.1 Digital Image Restoration for Phase-Coded Imaging Systems [16]

In this paper, the authors propose a pyramid-based structure to recover the image blurred by a specific point spread function(PSF). The procedure at each level is shown below

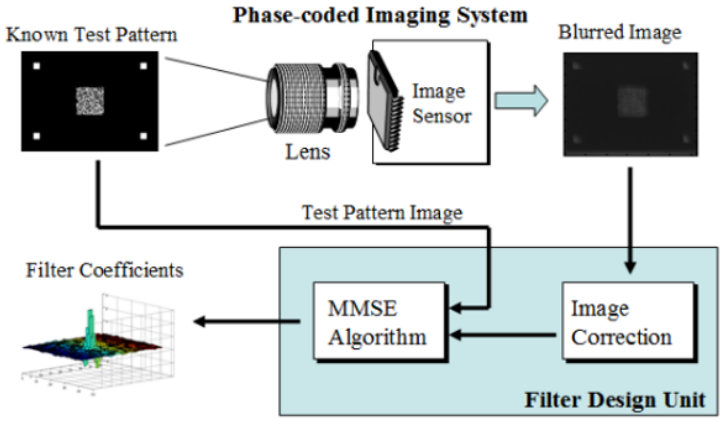

Figure 4.9: design flow of MMSE filter

The blurred image $B(i,j)$ can be modeled as

$$B(i,j) = \sum_{k+1}^{R}\sum_{k+1}^{Q} I(i+k, j+l)H(k,l) + N(i,j) \tag{112}$$

where I is the original image(known test pattern), H is the corresponding point-spread function in dimensions P and Q and N is the additive noise. The restored image is

$$\hat{I}(i,j) = \sum_{k=1}^{M}\sum_{l=1}^{N} B(i+k, j+l)W(k,l) \tag{113}$$

The output filter coefficients are the coefficients that minimize the mean-square-error(MSE) $E\{(I(i,j) - \hat{I}(i,j))^2\}$. By setting the derivation of MSE equals to 0, one can obtain the close form solution

$$
\begin{aligned}
&R_{IB}(k,l) = \sum_{p=1}^{M}\sum_{q=1}^{N} R_{BB}(k-p, l-q)W(p,q) \\
&\text{where } R_{IB}(k,l) = E\{I(i,j)B(i+k, j+l)\} \\
&\text{and } R_{BB}(k-p, l-q) = E\{B(i+p, j+q)B(i+k, j+l)\}
\end{aligned} \tag{114}
$$

The above equation can be rewritten in matrix from to solve W directly. That is, $\boldsymbol{r}_{IB} = \boldsymbol{R}_{BB}\boldsymbol{w}$, $\boldsymbol{w} = \boldsymbol{R}_{BB}^{-1}\boldsymbol{r}_{IB}$, where $\boldsymbol{r}_{IB}$ and $\boldsymbol{w}$ are vectors composed of R_{IB} and W respectively, and $\boldsymbol{R}_{BB}$ is a square matrix consisting of R_{BB}. The proposed pyramid structure is as follows

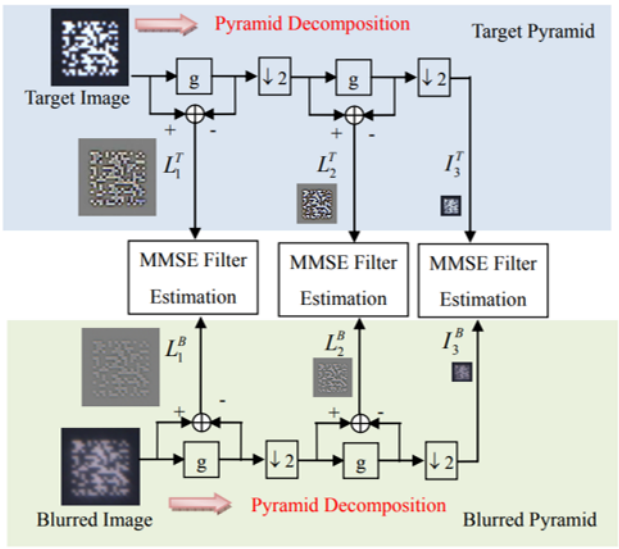

Figure 4.10: coefficients of each MMSE filter

In the above figure, three sets of coefficients are determined by passing a known pattern as input. The block g is the convolution of a Gaussian filter and $\downarrow 2$ is down sampling the image by 2. Note that input of MMSE filter is not the original image but the *residual* of the image before and after Gaussian filter g. Similarly, the reconstructing process is a three level pyramid but with the bottom part(i.e., blurred pyramid) being inverted.

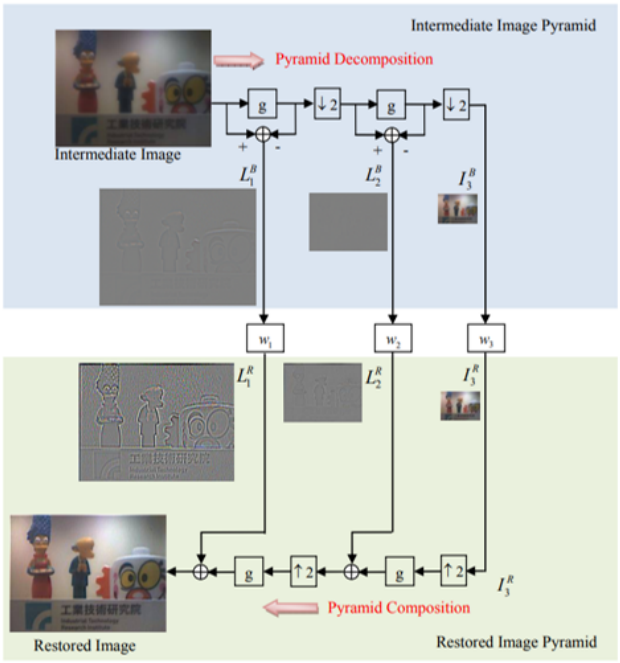

Figure 4.11: three-level pyramid-based restoration

The math expression of reconstruction can be written as

$$\begin{aligned} I_N^R &= w_N \star I_N^B \\ \hat{I}_i^R &= g \star (I_{i+1}^R)_{\uparrow 2} \\ I_i^R &= \hat{I}_i^R + w_i \star L_N^B \quad \text{for } 0 \leq i < N \end{aligned} \tag{115}$$

In fact, the pyramid-base structure in the above figure can be further improved with noise suppression. Assume that the original image is corrupted by noise after convolving with blur kernel. The noise distribution is denoted as $\sigma(y(x))\xi(x) = \eta_p(y(x)) + \eta_g$ where η_g is a signal-independent Gaussian component and $\eta_p(y(x))$ is a Poisson component depend on the value of blurred image $y(x)$. The additive noise variance of these two components is thus $\sigma^2(y(x)) = ay(x) + b$ since $var\{\eta_p(y(x))\} = ay(x)$ and η_g is a constant b. Both a, b depends only on the sensor hardware.

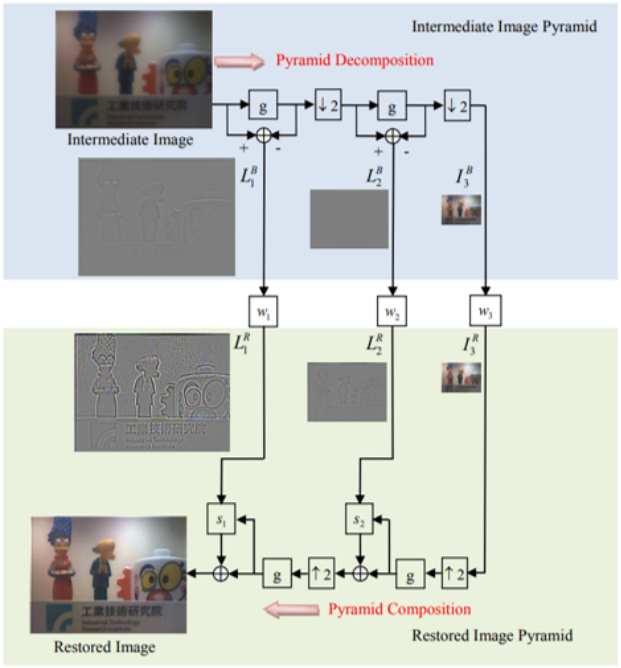

Figure 4.12: three-level pyramid-based restoration

One can see that threshold blocks s_i are added. The goal is to remove pixel values that exceed noise variance in L^R.i.e., at i^th level and pixel x, the new $\bar{L}_i^R(x)$ is

$$\begin{aligned}\bar{L}_i^R(x) &= s_i(L_i^R(x), \hat{I}_i^R(x)) \\ &= \begin{cases} L_i^R(x), & \text{for } (L_i^R(x))^2 \leq a_i \hat{I}_i^R(x) + b_i \\ 0 & \text{otherwise} \end{cases}\end{aligned} \tag{116}$$

4.2.2 Blind Deconvolution Using a Normalized Sparsity Measure [8] (code path: Deblurring_Gather/online_code/test_blind_deconv.m)

In this paper, a regularization term is proposed—ℓ_1/ℓ_2. The reason of using such regularization can be illustrated in the following figures:

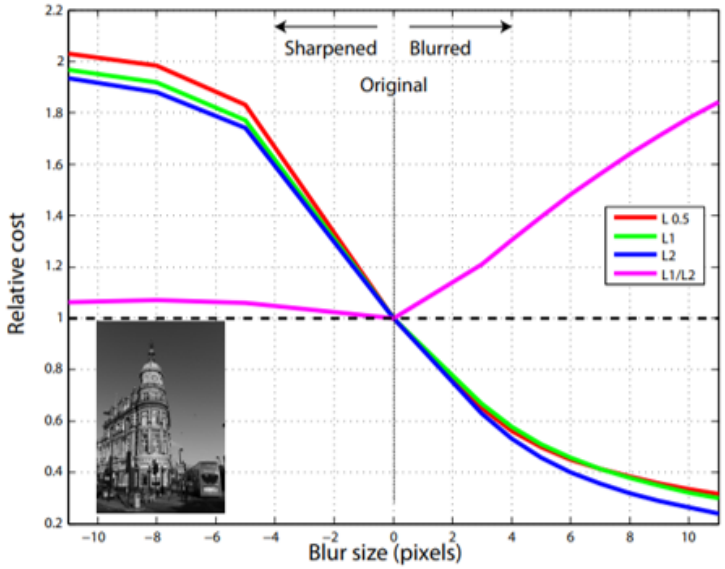

Figure 4.13: cost of ℓ_1/ℓ_2 and other regularization terms

First of all, one can see that for a given image, the higher the blur size is, the lower the relative cost will be. And the unsharp mask filter(i.e., negative blur size) will increase its cost on the other hand. In fact, the process of blurring can be seen as *smoothing* an image where the high frequency components are removed. Among these regularization terms, ℓ_1/ℓ_2 has desired characteristic—a minimum value located at original image.

Additionally, in the following figure, we consider the negative gradient direction of a two dimensional signal. The negative direction of the ℓ_1 norm points to the original(i.e., zero vector); the ℓ_0 norm has zero gradient everywhere; the negative gradient of ℓ_1/ℓ_2 norm moves to the closest axis and *preserves the distance from the origin.*

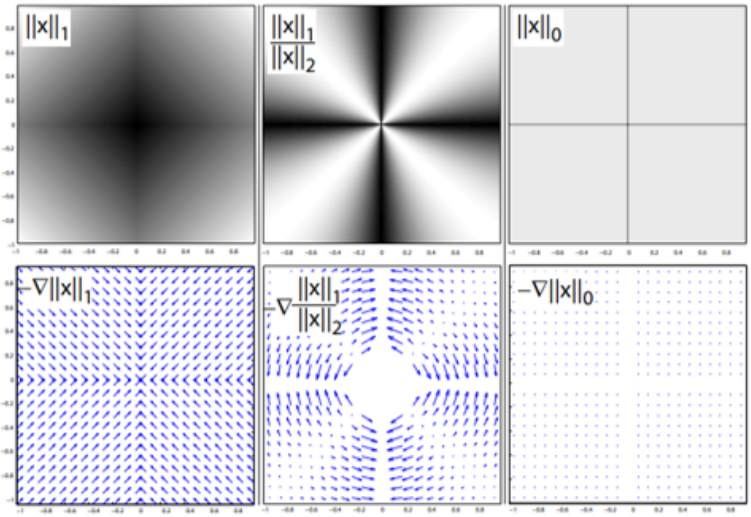

Figure 4.14: negative gradient vectors of ℓ_1/ℓ_2 and other regularization terms

Based on the features of ℓ_1/ℓ_2 of an image, the proposed objective function is as follows,

$$\min_{x,k} \lambda\|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi\|k\|_1 \tag{117}$$

where $\otimes$ denotes 2D convolution, k is the blur kernel, $y = [\nabla_x g, \nabla_y g]$ representing the concatenation of vertical and horizontal gradient of blurred image, and $k \geq 0, \quad \sum_i k_i = 1$ are assumed. The goal is to find a pair of latent image x and blur kernel k that minimize above equation. The algorithm solving objective function can be divide into two part: blind estimation of kernel and non-blind image deblurring.

Algorithm 1 : Overall Algorithm

Require: Observed blurry image g, Maximum kernel size h.
Apply derivative filters to g, creating a high-freq. image y.
1. Blind estimation of blur matrix K (Section 3.1) from y.
Loop over coarse-to-fine levels:
Alternate:
- Update sharp high-frequency image x (Section 3.1.1) using l_1/l_2 regularization.
- Update blurring matrix K (Section 3.1.2).
Interpolate solution to finer level as initialization.
2.Image recovery using non-blind algorithm of [12] (Section 3.2).
- Deblur g using K to give sharp image u.
return Sharp image u.

Algorithm 3 : x Update

Require: Blur kernel k from previous k update
Require: Image x^0 from previous x update
Require: Regularization parameter $\lambda = 20$
Require: Maximum outer iterations $M = 2$, inner its. $N = 2$
Require: ISTA threshold $t = 0.001$
1: **for** $j = 0$ to $M - 1$ **do**
2: $\lambda' = \lambda\|x^j\|_2$
3: $x^{j+1} =$ ISTA(k, λ', x^j, t, N)
4: **end for**
5: **return** Updated image x^M.

Algorithm 2 : Iterative Shrinkage-Thresholding Algorithm (ISTA)

Require: Operator K, regularization parameter λ
Require: Initial iterate x^0, observed image y
Require: Threshold t, maximum iterations N
1: **for** $j = 0$ to $N - 1$ **do**
2: $v = y - tK^T(Kx^j - y)$
3: $x^{j+1} = S_{t\lambda}(v)$
4: **end for**
5: **return** Output image x^N

Figure 4.15: complete algorithm

Blind estimation of kernel includes x update and k update. x update is Algorithm 3 in the above figure; the operator S in Algorithm 2 is the soft shrinkage operator on a vector $S_\alpha(x)_i = \max(|x_i| - \alpha, 0)\ \text{sign}(x_i)$. k update uses IRLS(appendix B) to solve

$$\min_k \lambda\|x \otimes k - y\|_2^2 + \psi\|k\|_1 \tag{118}$$

After acquiring the estimation of kernel K, the non-blind deblurring is performed since the kernel is known. The authors chose [7] (section 4.1.1) for its time efficiency and restored quality. The sample result is as follows

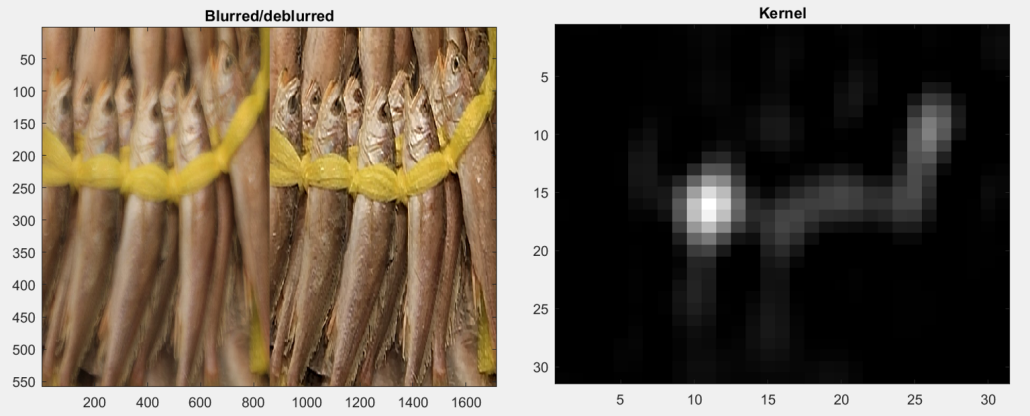

Figure 4.16: deblurred image and corresponding kernel of sample image

4.2.3 Deblurring Shaken and Partially Saturated Images [17]

In this paper, we consider the effect of camera exposure in optical images. Since the color scale is bounded by [0, 255], the bright region might experience saturation(i.e., values exceed 255 after being blurred are clipped to 255). The saturated parts in image make the process of deblurring difficult and may cause severe ringing effects.(see the figure below)

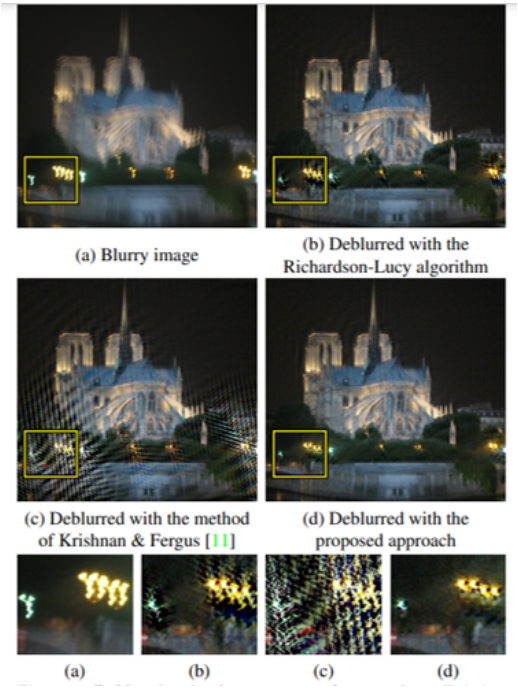

Figure 4.17: ringing effect due to saturation

Here the blur process is the same as previous assumptions

$$\boldsymbol{g} = \boldsymbol{A}\boldsymbol{f} \tag{119}$$

The matrix $\boldsymbol{A}$ can be further decomposed as $\boldsymbol{A} = \sum_k w_k \boldsymbol{T}_k$. A consists of several transformation matrix $\boldsymbol{T}$ simulating the translation of camera. w is proportional to time spent at view k. If we solve the problem directly by Richardson-Lucy method, the output of each iteration is simply

$$\boldsymbol{f}^{t+1} = \boldsymbol{f}^t \circ \boldsymbol{A}^T\left(\frac{\boldsymbol{g}}{\boldsymbol{A}\boldsymbol{f}^t}\right) \tag{120}$$

which is the same as in equation(80). However, this is not a proper model for the saturated images.

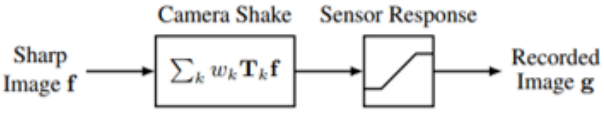

Figure 4.18: generation of saturated images

One can see from the above figure that Richardson-Lucy methods does not consider the saturation during blurring process. A straightforward way of solving this problem is to discard the saturated pixels.

$$\boldsymbol{f}^{t+1} = \boldsymbol{f}^t \circ \boldsymbol{A}^T\left(\frac{\boldsymbol{g} \circ \boldsymbol{z}}{\boldsymbol{A}\boldsymbol{f}^t} + \mathbf{1} - \boldsymbol{z}\right) \tag{121}$$

where $\boldsymbol{z}$ is a binary mask matrix. The elements in $\boldsymbol{z}$ is 1 if $g_i < T$ and 0 elsewhere. T is a user defined threshold to determined whether the pixel is saturated. However, this algorithm will result in a trade off between ringing effect and sharpness. Therefore, the authors proposed a *forward model for saturation*. The model introduce a function $R(x) = x - \frac{1}{a}\log(1 + e^{(a(x-1))})$. The modified Richardson-Lucy is thus

$$\boldsymbol{f}^{t+1} = \boldsymbol{f}^t \circ \boldsymbol{A}^T\left(\frac{\boldsymbol{g} \circ R'(\boldsymbol{A}\boldsymbol{f}^t)}{R(\boldsymbol{A}\boldsymbol{f}^t)} + \mathbf{1} - R'(\boldsymbol{A}\boldsymbol{f}^t)\right) \tag{122}$$

The design of $R(x)$ aims to simulate the response of saturation.

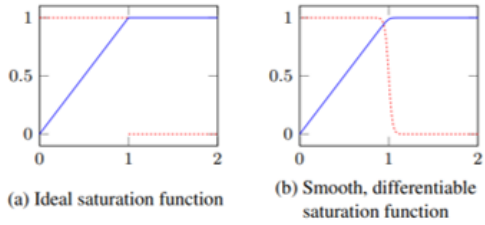

Figure 4.19: red line in (b): $R'(x)$, blue line in (b): $R(x)$

However, there is still a problem in the modified Richardson-Lucy algorithm. We know that the convolution with blur kernel can be seen as a process of *spreading* a pixel to its neighboring pixels. That is to say, if some(or all) its neighboring pixels are saturated and clipped, the information of that pixel become incomplete. To prevent this scenario, the authors separate the latent image to 2 subset $\boldsymbol{f} = \boldsymbol{f}_U + \boldsymbol{f}_S$. U denotes the set without saturation and the pixels in U are likely to be restored accurately. S, on the other hand, are pixels that are unlikely to be estimated accurately. The point is to keep the pixels in S from influencing the pixels in U. When performing Richardson-Lucy iteration, only the pixels in U are used for a *saturation-free* deblurring process. Hence, a binary mask v is defined; v corresponds to the set of point in V where $V = \cap_{k:w_k>0} U_{\boldsymbol{T}_k}$. Set V can be interpreted as: Intersection of points in U transformed by all $\boldsymbol{T}_k$. The condition of set U in each iteration is simply

$$U = \{j | f_j^t \leq \phi\} \tag{123}$$

The complete algorithm is to solve two Richardson-Lucy in each iteration

$$\begin{aligned}
&(a)\ \text{update}U, \quad U = \{j | f_j^t \leq \phi\} \\
&(b)\ \text{update}V, \quad V = \cap_{k:w_k>0} U_{\boldsymbol{T}_k} \\
&(c)\boldsymbol{f}_U^{t+1} = \boldsymbol{f}_U^t \circ \boldsymbol{A}^T(\frac{\boldsymbol{g} \circ R'(\boldsymbol{A}\boldsymbol{f}^t) \circ \boldsymbol{v}}{R(\boldsymbol{A}\boldsymbol{f}^t)} + \boldsymbol{1} - R'(\boldsymbol{A}\boldsymbol{f}^t) \circ \boldsymbol{v}) \\
&(d)\boldsymbol{f}_S^{t+1} = \boldsymbol{f}_S^t \circ \boldsymbol{A}^T(\frac{\boldsymbol{g} \circ R'(\boldsymbol{A}\boldsymbol{f}^t)}{R(\boldsymbol{A}\boldsymbol{f}^t)} + \boldsymbol{1} - R'(\boldsymbol{A}\boldsymbol{f}^t)) \\
&(e)\boldsymbol{f}^{t+1} = \boldsymbol{f}_U^{t+1} + \boldsymbol{f}_S^{t+1}
\end{aligned} \tag{124}$$

A sample result.

Figure 4.20: original blurred image

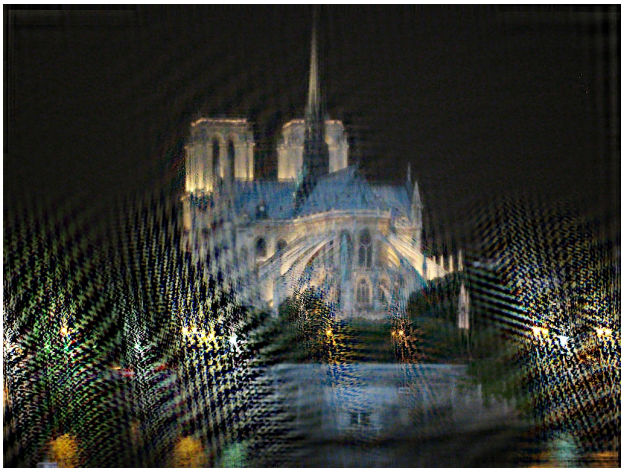

Figure 4.21: result of Hyper-Laplacian(4.1.1)

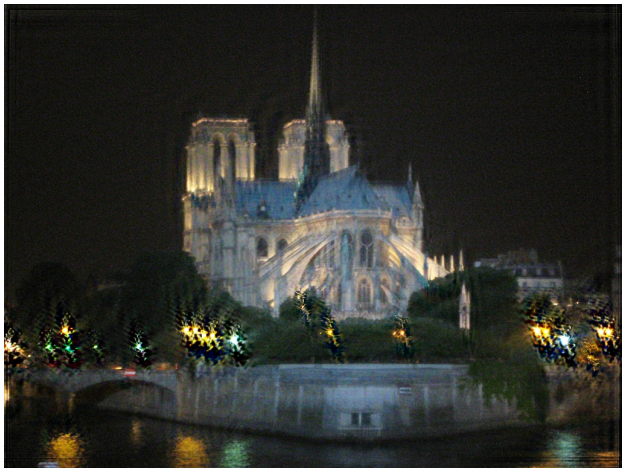

Figure 4.22: result of Richardson-Lucy algorithm

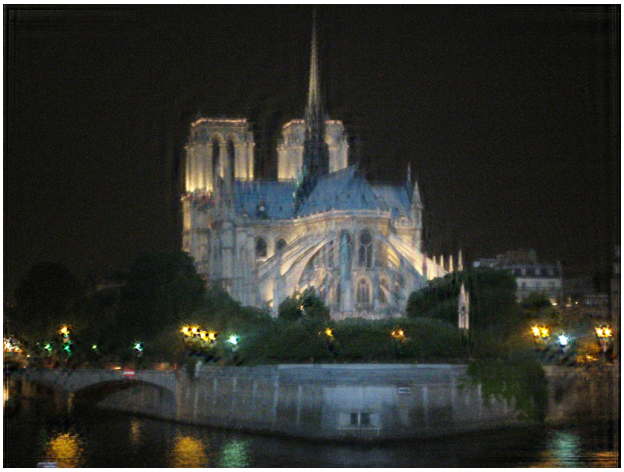

Figure 4.23: result of proposed method

4.2.4 Deblurring Text Images via L0-Regularized Intensity and Gradient Prior [11] (code path: Deblurring_Gather/text_deblurring_code_v4/demo_text_deblurring.m)

If we shrink the problem range from image deblurring to *blurrd text image deblurring*, more specific features are found to have regularize the restored image. The following figure shows the histogram of horizontal gradient of text image.

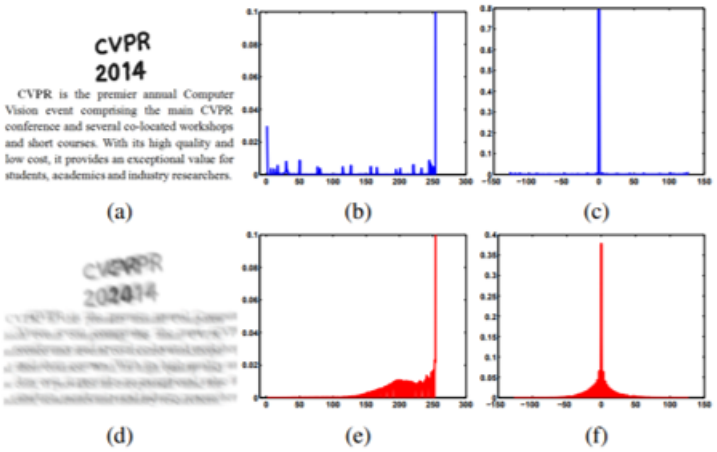

Figure 4.24: gradient values of clear and blurred text image

Most of the gradient values of a sharp text image are 0. On the other hand, the blurred one has considerable gradient values that are *close to 0*. Therefore, the goal is to *minimize* non-zero values in the gradient map. The resulting objective function is

$$\min_{x,k} \|x \star k - y\|_2^2 + \gamma \|k\|_2^2 + \lambda P(x) \tag{125}$$

where k is a unknown blur kernel, y is the given blurred text image, λ is a weighting parameter, and $P(x) = \sigma\|x\|_0 + \|\nabla x\|_0$. One may notice that not only the gradient but also the pixel values are taken into account; it will be further shown that both gradient and pixel values are necessary for regularization. Solving the objective function requires two step: latent image estimation.

$$\min_{x} \|x \star k - y\|_2^2 + \lambda P(x) \tag{126}$$

and kernel estimation

$$\min_{k} \|x \star k - y\|_2^2 + \gamma \|k\|_2^2 \tag{127}$$

In latent image estimation, two auxiliary variables u, g are used(just as in [7]).

$$\min_{x,u,g} \|x \star k - y\|_2^2 + \beta\|x - u\|_2^2 + \mu\|\nabla x - g\|_2^2 + \lambda(\sigma\|u\|_0 + \|g\|_0) \tag{128}$$

The algorithm shown below is used to restore latent image

Algorithm 1 Solving (6)

Input: Blur image y and blur kernel k.
$x \leftarrow y$, $\beta \leftarrow 2\lambda\sigma$.
repeat
  solve for u using (a)
  $\mu \leftarrow 2\lambda$.
  repeat
    solve for g using (b)
    solve for x using (c)
    $\mu \leftarrow 2\mu$.
  until $\mu > \mu_{\max}$
  $\beta \leftarrow 2\beta$.
until $\beta > \beta_{\max}$
Output: Intermediate latent image x.

Figure 4.25: latent image restoration

where

$$\begin{aligned}
&\text{(a) } u = \begin{cases} x, & |x|^2 \geq \frac{\lambda\sigma}{\beta} \\ 0, & \text{otherwise} \end{cases} \\
&\text{(b) } g = \begin{cases} \nabla x, & |\nabla x|^2 \geq \frac{\lambda}{\mu} \\ 0, & \text{otherwise} \end{cases} \\
&\text{(c) } x = \mathcal{F}^{-1}\left(\frac{\bar{\mathcal{F}}(k)\mathcal{F}(y) + \beta\mathcal{F}(u) + \mu F_G}{\bar{\mathcal{F}}(k)\mathcal{F}(k) + \beta + \mu\bar{\mathcal{F}}(\nabla)\mathcal{F}(\nabla)}\right)
\end{aligned} \tag{129}$$

Note that $\mathcal{F}(\cdot)$ denotes Fourier transform, and $\bar{\mathcal{F}}(\cdot)$ is the complex conjugate operator. $F_G = \mathcal{F}(\bar{\nabla}_h)\mathcal{F}(g_h) + \mathcal{F}(\bar{\nabla}_v)\mathcal{F}(g_v)$ where ∇_h and ∇_v denote the horizontal and vertical differential operators. After obtaining estimated latent image x, the blur kernel estimation is generated by the following algorithm

Algorithm 2 Blur kernel estimation algorithm

Input: Blur image y.
initialize k with the results from the coarser level.
for $i = 1 \to 5$ **do**
  solve for x using Algorithm 1.
  solve for k using (d)
  $\lambda \leftarrow \max\{\lambda/1.1, 1e^{-4}\}$.
end for
Output: Blur kernel k and intermediate latent image x.

Figure 4.26: complete algorithm

where (d) is the objective function with respect to kernel k

$$\min_k \|\nabla x \star k - \nabla y\|_2^2 + \gamma \|k\|_2^2 \tag{130}$$

The above optimization problem can be solved efficiently by fast Fourier transform(FFT). Note that initializing k with the result from the coarser level is commonly seen in many deblurring methods; the purpose is to reconstruct a unknown kernel from low resolution to high resolution.

We make a remark on the regularization term $P(x) = \sigma\|x\|_0 + \|\nabla x\|_0$. The following figure shows that both $\|x\|_0$ and $\|\nabla x\|_0$ are necessary. (h) is the intermediate result without the regularization of $\|x\|_0$. The restored image does not separate text and background properly. In comparison, if one didn't regularize $\|\nabla x\|_0$(i.e., (i)), the edges in restored image are vague.

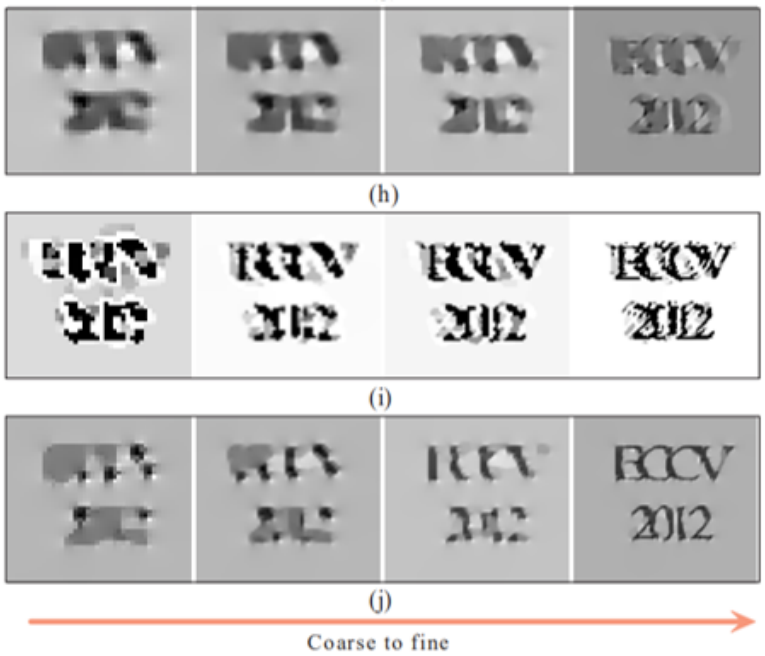

Figure 4.27: (h) intermediate salient edges using only $\|\nabla x\|_0$. (i) intermediate results using only $\|x\|_0$. (j) proposed intermediate salient edges

Finally, the result of sample code are shown below. The resolution of kernel increased at each coarser level.

Figure 4.28

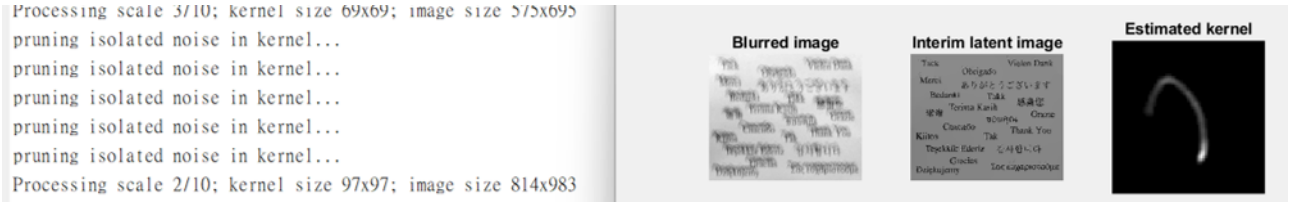

Figure 4.29

4.2.5 Blind Image Deblurring Using Dark Channel Prior [12] (code path: Deblurring_Gather/cvpr16_deblurring_code_v1/demo_deblurring.m)

In this paper, a novel feature the differentiate blurred images and sharp images is proposed—Dark channel prior. The dark channel of an image is defined as

$$D(I)(x) = \min_{y \in N(x)} (\min_{c \in \{r,g,b\}} I^c(y)) \tag{131}$$

In other words, given a three-channel image(i.e., r,g,b), the dark channel of pixel x is the smallest value in all three channel in the window $N(x)$ center at pixel x. The following figure shows the dark channel of a color image.

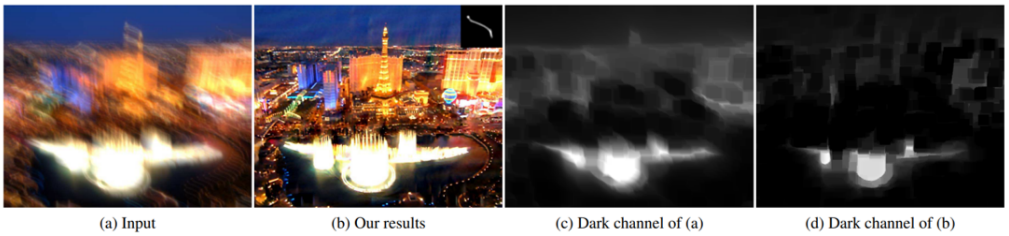

Figure 4.30: dark channel of blurred image and restored image

It may seem like (c) and (d) have no big differences when observed by naked eyes. However, the statistical property of dark channel is quite different in blurred image and sharp image.

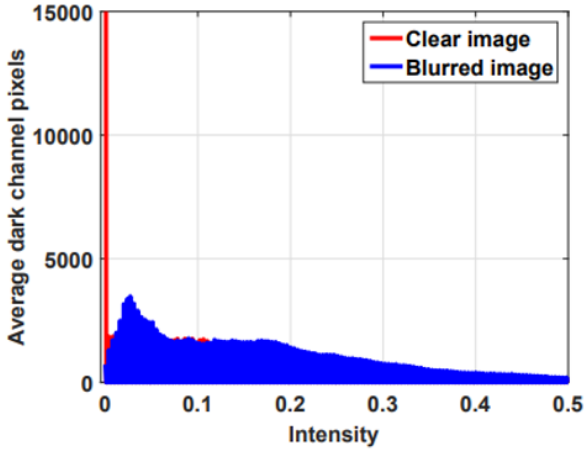

Figure 4.31: histogram of dark channel value

As one can see, The dark channel of sharp image has most of its values equals 0 whereas the dark channel values in blurred image has a much smoother distribution. It has been proved in the paper that

Let Ω *denote the domain of an image* I*. If there exist some pixels* $x \in \Omega$ *such that* $I(x) = 0$*, we have :*

$$\|D(B)(x)\|_0 > \|D(I)(x)\|_0 \tag{132}$$

where B is the blurred image, I is the clear image, and the L_0 norm $\|X\|_0$ is counting the number of non-zero values in X. The authors utilize this characteristic to prosed a regularization term $\|D(I)\|_0$. The point is to minimize non-zero pixel in the dark channel in the restored image. The objective function is thus

$$\min_{I,k} \|I \star k - B\|_2^2 + \gamma\|k\|_2^2 + \mu\|\nabla I\|_0 + \lambda\|D(I)\|_0 \tag{133}$$

γ, μ and λ are user defined weight parameters. Additionally, both the gradient and dark channel of latent image I are taken into account. Minimizing the objective function requires two steps: solve for latent image I:

$$\min_{I} \|I \star k - B\|_2^2 + \mu\|\nabla I\|_0 + \lambda\|D(I)\|_0 \tag{134}$$

and solve for blur kernel k:

$$\min_{k} \|I \star k - B\|_2^2 + \gamma\|k\|_2^2 \tag{135}$$

The function in step one has the similar form in section 4.1.1. Two auxiliary variables are introduced.

$$\min_{I,u,g} \|I \star k - B\|_2^2 + \alpha\|\nabla I - g\|_2^2 + \beta\|D(I) - u\|_2^2 + \mu\|g\|_0 + \lambda\|u\|_0 \tag{136}$$

The above equation can be further divide into three sub-steps

- (fix u, g) $\boldsymbol{I} = \min_{\boldsymbol{I}} \|\boldsymbol{T}_k\boldsymbol{I} - \boldsymbol{B}\|_2^2 + \alpha\|\nabla\boldsymbol{I} - \boldsymbol{g}\|_2^2 + \beta\|\boldsymbol{M}\boldsymbol{I} - \boldsymbol{u}\|_2^2$
  Where $\boldsymbol{T}_k$ is the convolution matrix of k. $\boldsymbol{M}\boldsymbol{I} = \boldsymbol{D}(\boldsymbol{I})$. M represents the operator of selecting dark channel value of image I.

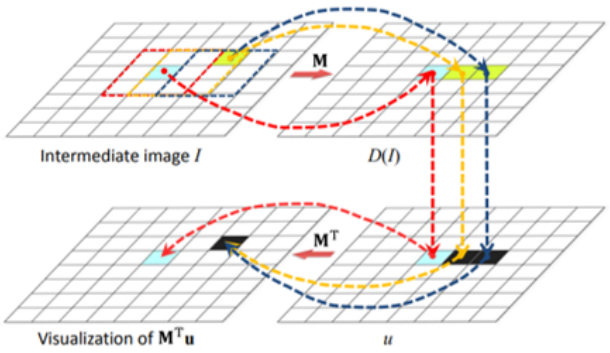

Figure 4.32: schematic diagram of mapping matrix M

Since all three terms are squared and $\boldsymbol{T}_k$ is Toeplitz matrix, $\boldsymbol{I}$ can be solve by direct fast Fourier transformation.

- (fix x, g) $u = \min_u \beta\|D(I) - u\|_2^2 + \lambda\|u\|_0$
  The result is given by

$$u = \begin{cases} D(I) & |D(I)|^2 \geq \frac{\lambda}{\beta} \\ 0 & \text{otherwise} \end{cases} \tag{137}$$

- (fix x, u) $g = \min_g \alpha\|\nabla I - g\|_2^2 + \mu\|g\|_0$
  The result is given by

$$g = \begin{cases} \nabla I & |\nabla I|^2 \geq \frac{\mu}{\alpha} \\ 0 & \text{otherwise} \end{cases} \tag{138}$$

After completing these three steps, one can obtain the latent image I. The second step is estimating blur kernel k i.e.,

$$k = \min_k \|I \star k - B\|_2^2 + \gamma\|k\|_2^2 \tag{139}$$

The objective function of k can also be solved directly by fast Fourier transform as the first sup-step in estimating I(section 4.1.1).

Here we make a final remark on the algorithm. The kernel estimation step is solved in a coarse-to-fine manner. That is to say, the kernel is estimated from low resolution to high resolution. A sample result is shown as follows

Figure 4.33: deblurring result of the first level

Figure 4.34: deblurring result of the fifth level

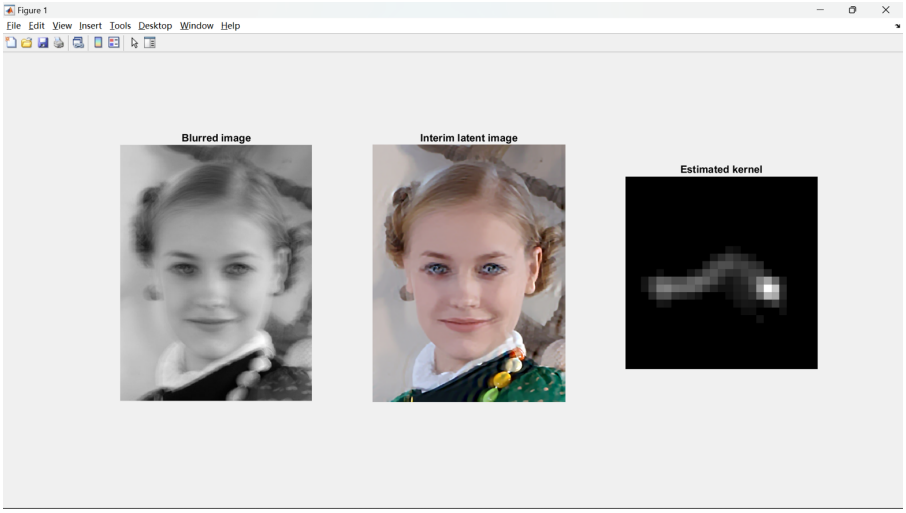

Figure 4.35: final result of latent image and kernel

Non-uniform Deblurring

In previous chapters, we assume that the blur kernel is space invariant. Yet in real world, an image is very likely to be blurred non-uniformly. Affected by various blur kernels, this type of problem can be rather complex.

5.1 Image and Depth from a Conventional Camera with a Coded Aperture [9] (code path: Deblurring_Gather/DeconvolutionCode-LevinEtAl07)

When taking a picture from a 3 dimensional world, objects from different depth are projected to the camera sensor.

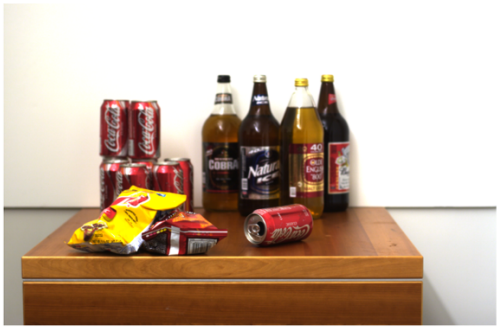

Figure 5.1: objects from different depth

Due to the principle of optical camera, objects from different depth experience different scales of blur. The schematic diagram is as follows

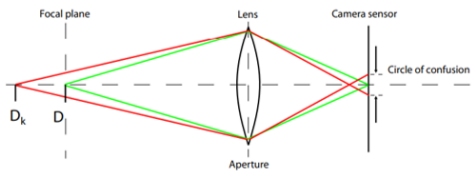

Figure 5.2: objects from different depth

The projected image(i.e., the blurred image) y can be expressed as

$$y = f_k \star x \tag{140}$$

where f_k is the blur kernel at depth k and x is the latent image. Now, the problem is to find a filter that can distinguish objects from different depths. Consider a toy example below

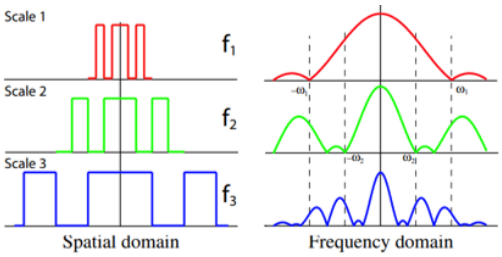

Figure 5.3: a simple 1D filter

The Fourier transform of a filter of different scales have zeros at different frequencies. By examining where the zeros of Fourier transform of y (i.e.,Y) locates, one can determine the depth of the blurred object and performed deblurring process accordingly. Yet there are two main challenges: the design of filter and the possible existence of noise. First, the filter must follow these constrains.

1. Consider figure 5.3, the targeted filter must be easily invert so that $y = f_k \star x$ can be solve.
2. The zeros in frequency domain should not overlap.
3. The filter is binary.
4. No floating parts in the center of coded aperture since it's implementation impossible.
5. Avoid excessive radial distortion by not using the full aperture.

Combining above conditions, the proposed coded aperture is shown below(top right)

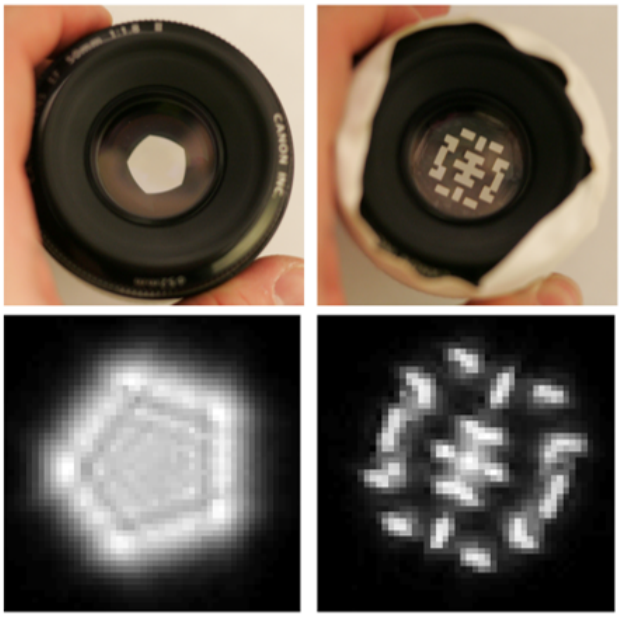

Figure 5.4: conventional aperture and coded aperture

After one obtains the image captured by the coded aperture, the next step is to remove possible noise. Given the object conditional probability model $P(x|y)$,

$$P(x|y) \propto e^{-(\frac{1}{\eta^2}|C_{f_k}x-y|^2+\alpha|C_{g_x}x|^2+\alpha|C_{g_y}x|^2)} \tag{141}$$

where η^2 is the variance of additive Gaussian noise $N(0, \eta^2 I)$, $C_{f_k}x = f_k \star x$, and C_{g_x}, C_{g_y} corresponds to the horizontal derivative matrix and vertical derivative matrix. The object function is thus

$$x_{est} = \arg\min_x \frac{1}{\eta^2}|C_{f_k}x - y|^2 + \alpha|C_{g_x}x|^2 + \alpha|C_{g_y}x|^2 \tag{142}$$

The answer is rather simple since the above equation can be seen as a set of linear equation $Ax = b$ with $A = \frac{1}{\eta^2}C_{f_k}^T C_{f_k} + \alpha C_{g_x}^T C_{g_x} + \alpha C_{g_y}^T C_{g_y}$ and $b = \frac{1}{\eta^2}C_{f_k}^T y$. Another objective function is

$$x_{est} = \arg\min_x \frac{1}{\eta^2}|C_{f_k}x - y|^2 + \alpha|C_{g_x}x|^{0.8} + \alpha|C_{g_y}x|^{0.8} \tag{143}$$

which is the exact same objective function as in section 4.1.1. In this case, a longer run time are required for a sharper restored image. Optimizing the objective function gives us the restored image blurred by kernel f_k. k images are generated as *candidates*. As one can see, a depth map is need so that one can determine the depth at different part in the image. The depth is determined as follows. First define reconstruction error at depth k, $e_k = y - f_k \star x_k$. The averaging error

$\hat{E}_k(y(i)) \approx \sum_{j \in window} e_k(j)^2$. Chose the depth k that minimize $\lambda_k \hat{E}_k(y(i))$, i.e.,

$$d(i) = \arg\min_k \lambda_k \hat{E}_k(y(i)) \tag{144}$$

where λ_k is learnt to minimize the scale misclassification error on a set of training images having a known depth profile. The resulting depth map is as follows

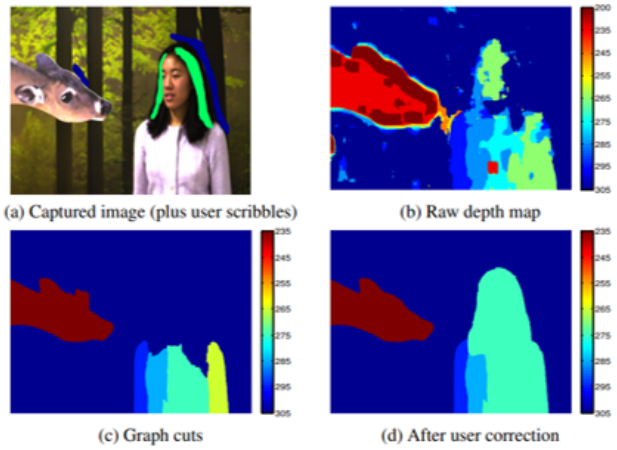

Figure 5.5: depth map

The complete algorithm can be summarized as

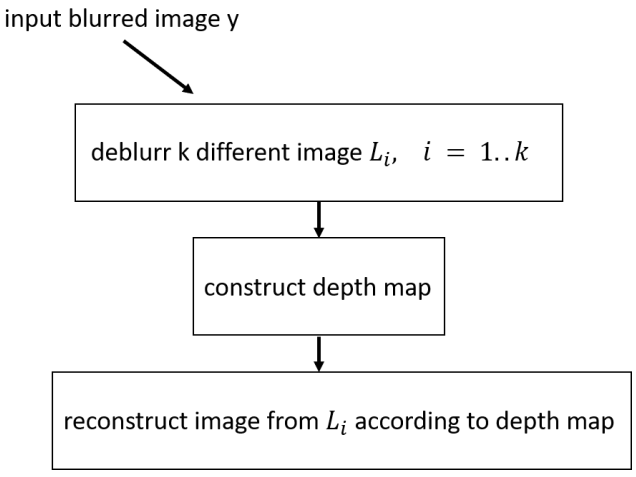

Figure 5.6: summarization

For more details of λ_k and smoothing of depth map, please refer to the paper. Note that the MATLAB code of this paper is just an implementation of 4.1.1 since the core deblurring algorithms are the same.

5.2 Spatially-Varying Out-of-focus Image Deblurring with L1-2 Optimization and A Guided Blur Map [14] (code path: Deblurring_Gather/ICASSP2012/main.m)

In this paper, a method is proposed to solve similar problem as in the previous section: spatially-varying out-of-focus blurring process. Since objects of different distances from the camera lens will experience different degrees of blur, the authors proposed a way to distinguish the blur kernels in the blurred and then deblur each part accordingly. The flow chart of proposed method is as follows

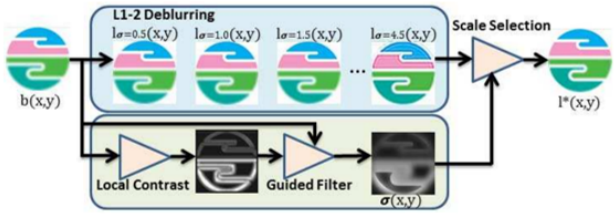

Figure 5.7: flow chart of proposed method

The idea here is to build a *blur map* which indicates the degree of blur a pixel suffer from(i.e., Brighter pixel is convolved by stronger blur kernel). Additionally, the blurred image is deblurred using different blur kernel. Pixels in final result are patched up according to its degree of blur according to the blur map.

The blur kernels $g(x, \sigma)$ are assumed to be Gaussian functions(i.e., $g(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma}}$) in this paper. The σ value of the kernel that a pixel convolved with can be determined by

$$\sigma(x, y) = \frac{1}{\sqrt{2\pi} LC(x, y)} = \frac{0.3989}{LC(x, y)} \tag{145}$$

$LC(x, y)$ is the local contrast prior $LC(x, y)$ given by

$$LC(x, y) = \frac{\max |\nabla b(x', y')|}{\max b(x', y') - \min b(x', y')} \tag{146}$$

where b is the input defocused image, $|\nabla b(x, y)| = \sqrt{\nabla b_x^2 + \nabla b_y^2}$, representing the magnitude of 2 dimensional gradient, and (x', y') is the neighboring points of (x, y) within a user-defined local window. $\sigma(x, y)$ is then filtered by a guided filter using the original blurred image.

$$\sigma(x, y) =\leftarrow GF\{b(x, y), \sigma(x, y), r, \varepsilon\} \tag{147}$$

r, ε being user-defined parameter. The filtered result is the blur map of the blurred image. Meanwhile, the original image is deblurred by different blur kernel using $L1-2$ optimization. The objective function is

$$\min_{l_\sigma} \frac{\mu}{2}\|G_\sigma l_\sigma - b\|_2^2 + \alpha \sum_{i=1}^{n^2}\|D_i l_\sigma\| + (1-\alpha)\sum_{i=1}^{n^2}\|D_i l_\sigma\|_2^2 \tag{148}$$

As in section 4.1.1, the objective function can be solve iteratively by introducing auxiliary variables

$$\min_{l_\sigma, w} \frac{\mu}{2}\|G_\sigma l_\sigma - b\|_2^2 + \alpha(\sum_{i=1}^{n^2}\|w_i\| + \frac{\beta}{2}\sum_{i=1}^{n^2}\|w_i - D_i l_\sigma\|) + (1-\alpha)\sum_{i=1}^{n^2}\|D_i l_\sigma\|_2^2 \tag{149}$$

The deblurring process is thus

- **update w_i:**

$$w_i = \max\{\|D_i l_\sigma\| - \frac{1}{\beta}, 0\}\frac{D_i l_\sigma}{\|D_i l_\sigma\|} \tag{150}$$

- **update l_σ:**

$$l_\sigma = \mathcal{F}^{-1}\{\frac{(\frac{\alpha\beta}{2})\mathcal{F}(D_i)^* \circ \mathcal{F}(w_i) + (\frac{\mu}{2})\mathcal{F}(G_\sigma)\circ\mathcal{F}(b)}{(\frac{\alpha\beta}{2} + (1-\alpha))\mathcal{F}(D_i)^* \circ \mathcal{F}(D_i) + (\frac{\mu}{2})\mathcal{F}(G_\sigma)\circ\mathcal{F}(b)}\} \tag{151}$$

$\circ$ is the element-wise multiplication and $*$ denotes complex conjugation. Now, having blur map and several deblurred images $\{l_{\sigma 1}, l_{\sigma 2}, \cdots, l_{\sigma N}\}$, the all-in-focus result is given by

$$l^*(x,y) = \sum_{(x,y)} l_{\sigma^*(x,y)}(x,y) \tag{152}$$

While $\{l_{\sigma 1}, l_{\sigma 2}, \cdots, l_{\sigma k}, \cdots, l_{\sigma N}\}$ are results from blur kernel of quantized σ, $\sigma(x,y)$ in blur map is excepts to be consecutive. Thus, $\sigma^*(x,y)$ is chosen to be *the biggest blur scale close to* $l_{\sigma k}$. In short, the proposed method can be summarized into three parts: (a)blur map generation, (b) $L1-2$ deblurring, and (c) scale selecting.

A sample result

Figure 5.8: out-of-focus image

Figure 5.9: blur map

Figure 5.10: restored all-in-focus image

5.3 Unnatural L0 Sparse Representation for Natural Image Deblurring [20] (code path: Deblurring_Gather/Non_Uniform_Pcode/runNon_Uniform_

In this paper, a method for both uniform and non-uniform is proposed. The novel point of this paper is that it proposed a new sparsity function regularizing the estimated latent image. To begin with, consider the blur process

$$y = \sum_m k_m H_m + \varepsilon \tag{153}$$

which is the same as in section 4.2.2. H_m is the transformation matrix of the camera, k_m denotes the weight that is proportional to the time camera spent on pose m. The transformation matrix can be separate into two cases: rotation matrix R_m and translation matrix $M - m$. The blurring process can thus be rewritten as

$$\sum_m k_m R_m x = B_R x = A_R k \tag{154}$$

$$\sum_m k_m M_m x = B_M x = A_M k \tag{155}$$

The reason of separating H_m into two cases is that the translation matrix M_m is referred as uniform blur whereas R_m is sufficient to model non-uniform blur. With these assumption, the objective function in this paper is given as

$$\min_{\tilde{x},k} \quad \|\sum_m k_m H_m \tilde{x} - y\| + \lambda \sum_{\star \in \{h,v\}} \phi_0(\partial_\star \tilde{x}) + \gamma \|k\|^2 \tag{156}$$

where λ and γ are user defined parameters. $\phi_0(\partial_\star \tilde{x})$ is a novel sparsity function proposed in this paper as follows

$$\begin{aligned} \phi_0(\partial_\star z) &= \sum_i \phi(\partial_\star z_i) \\ \phi(\partial_\star z_i) &= \begin{cases} \frac{1}{\varepsilon^2}|\partial_\star z_i|^2 & \text{if } |\partial_\star z_i| \le \varepsilon \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{157}$$

The two main advantages of $\phi_0(\partial_\star z)$ is that $\phi_0(\partial_\star z)$ is similar to L_0 norm(see (a) in the following figure)

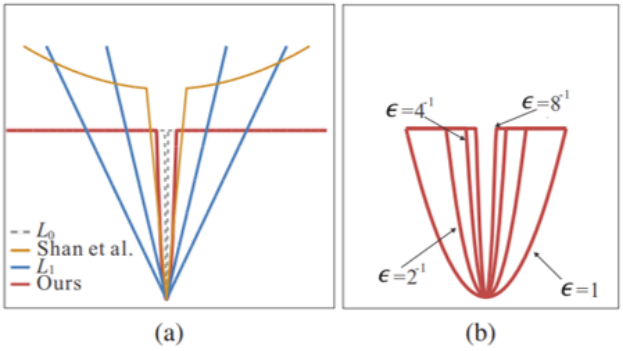

Figure 5.11: plots of different penalty functions

Additionally, since $\phi_0(a^t\partial_\star z) \simeq \phi_0(\partial_\star z)$, $\phi(\cdot)$ has an effect to remove fine structures and keep useful salient ones in the estimated latent image. In other words, this regularization term is *scale invariant*.

Now we move to the optimization part. In this tutorial we omit quite a lot mathematical details and derivation in the original for simplicity. To objective function is solved by alternatively computing

$$\begin{aligned} \tilde{x}^{t+1} &= \arg\min_{\tilde{x}}\{\|B^t\tilde{x} - y\|^2 + \lambda \sum_{\star\in\{h,v\}} \phi_0(\partial_\star \tilde{x})\} \\ k^{t+1} &= \arg\min_{k}\{\|A^{t+1}k - y\|^2 \ + \gamma\|k\|^2\} \end{aligned} \tag{158}$$

First, use half-quadratic L_0 solver to optimize $\tilde{x}^{t+1}$. The objective function is rewritten as:

$$\arg\min_{\tilde{x},l}\{\frac{1}{\lambda}\|B\tilde{x} - y\|^2 + \sum_{\star\in\{h,v\}}\sum_i \{|l_{\star i}|^0 + \frac{1}{\varepsilon^2}(\partial_\star \tilde{x}_i - l_{\star i})^2\}\} \tag{159}$$

Thus, solving $\tilde{x}$ needs to solve two sub-problems as well.

Update l:

$$\text{(a) } l_{hi(vi)} = \begin{cases} 0, & \text{if } |\partial_{h(v)}\tilde{x}_i| \le \varepsilon \\ \partial_{h(v)}\tilde{x}_i, & \text{otherwise} \end{cases} \tag{160}$$

Update $\tilde{x}$:

1. Uniform(b): $\mathcal{F}(\tilde{x}) = \frac{\overline{\mathcal{F}(B_M)}\cdot\mathcal{F}(y) + \frac{\lambda}{\varepsilon^2}(\overline{\mathcal{F}(\partial_h)}\cdot\mathcal{F}(l_h) + \overline{\mathcal{F}(\partial_v)}\cdot\mathcal{F}(l_v))}{\overline{\mathcal{F}(B_M)}\cdot\mathcal{F}(B_M) + \frac{\lambda}{\varepsilon^2}F_D^2}$, where $F_D^2 = |\mathcal{F}(\partial_h)|^2 + |\mathcal{F}(\partial_v)|^2$
2. Non-uniform(c): $\tilde{x} = \frac{1}{W}\sum_p C_p^{-1}\mathcal{F}^{-1}\frac{\overline{\mathcal{F}(C_k(A_{R\delta}k))} + \mathcal{F}(w\cdot C_p(y)) + \frac{\lambda}{\varepsilon^2}F_{Dl}^2}{F_k^2 + \frac{\lambda}{\varepsilon^2}F_D^2}$, where $F_k^2 = |\mathcal{F}(C_k(A_{R\delta}k))|^2$ and $F_{Dl} = (\overline{\mathcal{F}(\partial_h)}\cdot\mathcal{F}(C_p(l_h)) + \overline{\mathcal{F}(\partial_v)}\cdot\mathcal{F}(C_p(l_v)))$

The math equation in non-uniform case is derived by cutting blurred image into p patches. In this case, it can be assumed that these patches are *locally uniform*. C_p and C_p^{-1} means extracting patch p and pasting back patch p to the original image respectively. Additionally, $A_{R\delta}k = \sum_i k_i R_i \delta$ representing the blur kernel for patch p.
Now let's move on to the problem of solving k. We consider uniform and non-uniform cases as the previous step.

1. Uniform(d): $k^{t+1} = \mathcal{F}^{-1}(\frac{\overline{\mathcal{F}(A_M^{t+1})}\mathcal{F}(y)}{|\mathcal{F}(A_M^{t+1})|^2+\gamma})$
2. Non-uniform(e): $k^{t+1} = k^t \cdot (\frac{A_R^T y}{(A_R^T A_R+\gamma)k^t})^\alpha$

where α is the step size of iteration. The overall algorithm is given below

Algorithm 1: L_0 Deblurring in One Image Level

input : blurred image y
output: blur kernel k, deblurred image $\tilde{x}$

initialize k^1 from the coarser-scale kernel estimate
for $t = 1 : 5$ **do**
    // update image
    $\epsilon \leftarrow 1$
    for $i = 1 : 4$ **do**
        for $j = 1 : \epsilon^{-1}$ **do**
            solve for l using (a)
            solve for $\tilde{x}^{t+1}$ using (b) or (c)
        end
        $\epsilon \leftarrow \epsilon/2$ *// graduate non-convexity*
    end
    // update kernel
    solve for k^{t+1} using (d) or (e)
end

Figure 5.12: complete algorithm

A sample result:

Figure 5.13: blurred image

Figure 5.14: restored image

Deep Learning in Image Deblurring

with the burst of deep learning in recent years, some have found that deep learning can also be applied to solve ill-posed problems such as image deblurring. In this chapter, we categorize the methods into blind and non-blind deblurring like we've done in chapter 4. In this tutorial, however, only fundamental concepts are introduced. For a better understanding of deep learning in image deblurring, one can refer to [21].

6.1 Non-blind Deblurring

6.1.1 Deep Convolutional Neural Network for Image Deconvolution [19] (code path: Deblurring_Gather/dcnn_nips14/run_deblur_all_3chs_finetune.m)

When it comes to deblurring methods using deep learning, there are two indispensable elements: one or more cost functions to be minimized, and the proposed network architecture. In this paper, a convolution neural network(CNN) is used to reconstruct the blurred image. The loss function of the network is, not surprisingly,

$$\frac{1}{2|N|}\sum_{i\in N}\|f(\hat{y}_i)-\hat{x}_i\|^2 \tag{161}$$

where N is the size of training set, and $\{\hat{x}_i, \hat{y}_i\}$ is a pair of a clear image and a corrupted version of it. The architecture of the network proposed in this paper is as follows

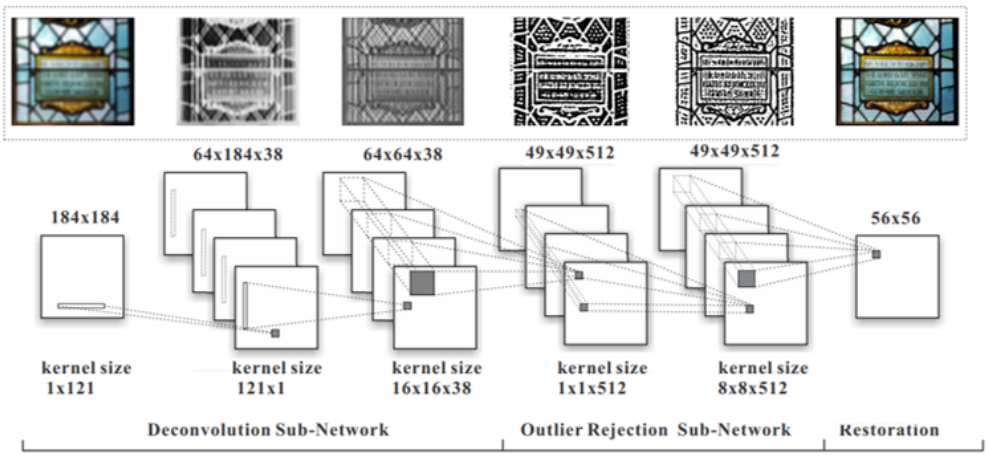

Figure 6.1: complete network architecture for deep deconvolution

The network consist of two part: image deconvolution CNN(DCNN) and outlier-rejection deconvolution CNN(ODCNN). The architecture of DCNN is given as

$$h_3 = W_3 * h_2; \quad h_l = \sigma(W_l * h_{l-1} + b_{l-1}), l \in \{1, 2\}; \quad h_0 = \hat{y} \tag{162}$$

where W_l is the weight mapping the $(l-1)^{th}$ layer to the l^{th} one and b_{l-1} is the vector value bias. $\sigma(\cdot)$ is the nonlinear function, which can be sigmoid or hyperbolic tangent. (The architecture of ODCNN will be discuss in next section). DCNN is trained by the natural images degraded by additive Gaussian noise and JPEG compression. However, the initialization of model parameters play an important part in training DCNN. The authors show that the kernel k' in Wiener deconvolution

$$x = \mathcal{F}^{-1}(\frac{1}{\mathcal{F}(k)}\{\frac{|\mathcal{F}(k)|^2}{|\mathcal{F}(k)|^2 + \frac{1}{SNR}}\}) * y = k' * y \tag{163}$$

can be used as initial values of model parameters to achieve better result.

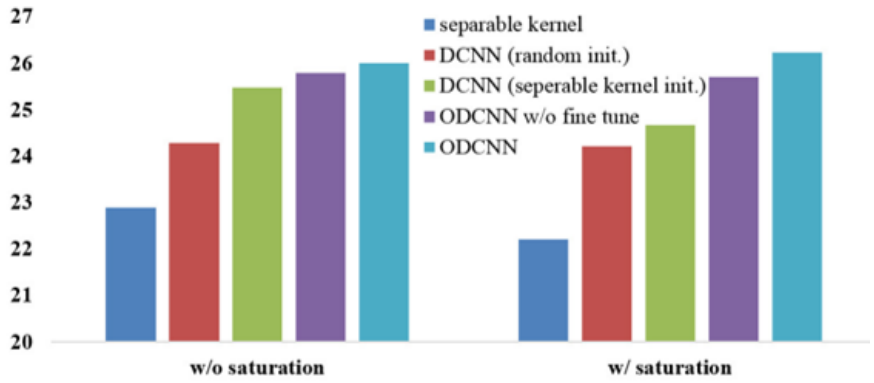

Figure 6.2: PSNRs produced in different stages CNN architecture

ODCNN, on the other hand, is trained by 2500 natural images from Flicker. The authors sample patches in these natural images and blur them as training data for ODCNN.

6.1.2 Restoring an Image Taken through a Window Covered with Dirt or Rain [4]

In this section, we introduce the ODCNN used in the previous method. The purpose of ODCNN is to remove small particles such as dirt and rain in an image. The figure below shows the input and output of ODCNN

Figure 6.3: (left): blurred image taken through glass that is covered with rain, (right): output of ODCNN

The loss function is given as

$$j(\theta) = \frac{1}{2|D|}\sum_{i\in D}\|F(x_i) - y_i^*\|^2 \tag{164}$$

where θ is the model parameters $\theta = (W_1, \cdots, W_L, b_1, \cdots, b_L)$. D is the data set containing image pairs (x_i, y_i^*) which corresponds to the noisy and clean image. The N-layer architecture is given as

$$\begin{aligned} F_0(x) &= x \\ F_l(x) &= tanh(W_l * F_{l-1}(x) + b_l), \quad l = 1, \cdots, L-1 \\ F(x) &= \frac{1}{m}(W_L * F_{L-1}(x) + b_L) \end{aligned} \tag{165}$$

The training data is mainly composed by images corrupted by dirt and water droplets. To simulation the effect of dirt, the artificial noisy image I' is generate as

$$I' = p\alpha D + (1-\alpha)I \tag{166}$$

where α is a transparency mask, D is the additive component of the dirt, and p is a random perturbation vector in RGB space. The water droplets, on the other hand, is simulated by spraying water on a pane of anti-reflective MgF2-coated glass, producing drops that closely resemble to real rain. In fact, the visualization of model weights at different layer shows how the model works.

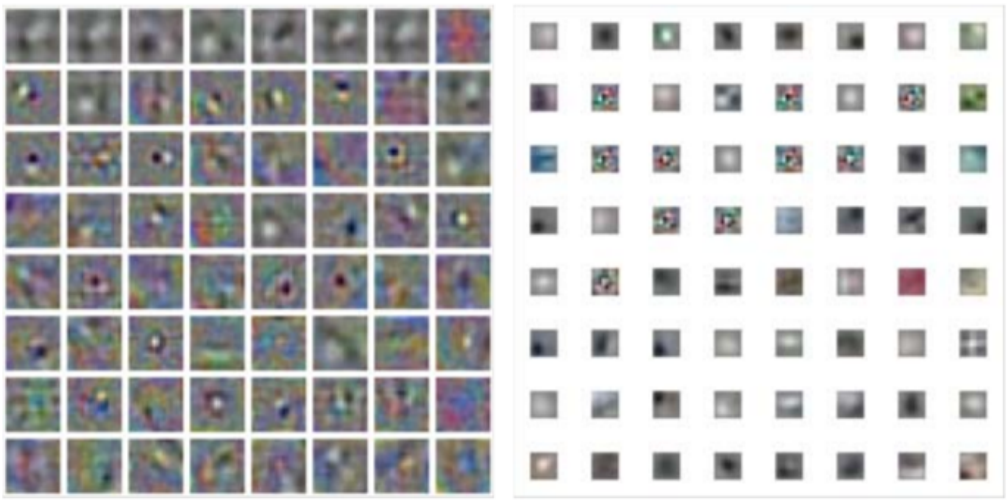

Figure 6.4: (left): first layer filters. (right): top layer filters used to reconstruct the clean patch

Since the proposed model is based on convolutional neural network(CNN), the filers in the first layer will *detect* and *extract* possible features of rain. And the filters in top layer are used to remove these distortion by convolution.

6.2 Blind Deblurring

6.2.1 Convolutional Neural Networks for Direct Text Deblurring [5]

In this paper, a convolutional neural network(CNN) is proposed for *text* deblurring. The loss function is straightforward,

$$\frac{1}{2|D|}\sum_{x_i,y_i\in D}\|F(y_i)-x_i\|_2^2+0.0005\|W\|_2^2 \tag{167}$$

where W and b are model parameters. D is the data set containing image pairs (x_i, y_i) which corresponds to the noisy and clean image. For a L-layer architecture, the model can be written as

$$\begin{aligned} F_0(y) &= y \\ F_l(y) &= max(0, W_l * F_{l-1}(y) + b_l), \quad l = 1, \cdots, L-1 \\ F(y) &= W_L * F_{L-1}(y) + b_L \end{aligned} \tag{168}$$

For a 15-Layer, the filter characteristic is shown in the following figure

Table 1: CNN architecture – filter size and number of channels for each layer.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L15	19×19	1×1	1×1	1×1	1×1	3×3	1×1	5×5	5×5	3×3	5×5	5×5	1×1	7×7	7×7
	128	320	320	320	128	128	512	128	128	128	128	128	256	64	3
L10	23×23	1×1	1×1	1×1	1×1	3×3	1×1	5×5	3×3	5×5					
S	128	320	320	320	128	128	512	48	96	3					
M	196	400	400	400	156	156	512	56	128	3					
L	220	512	512	512	196	196	512	64	196	3					

Figure 6.5: parameters of 10-layer and 15-layer model

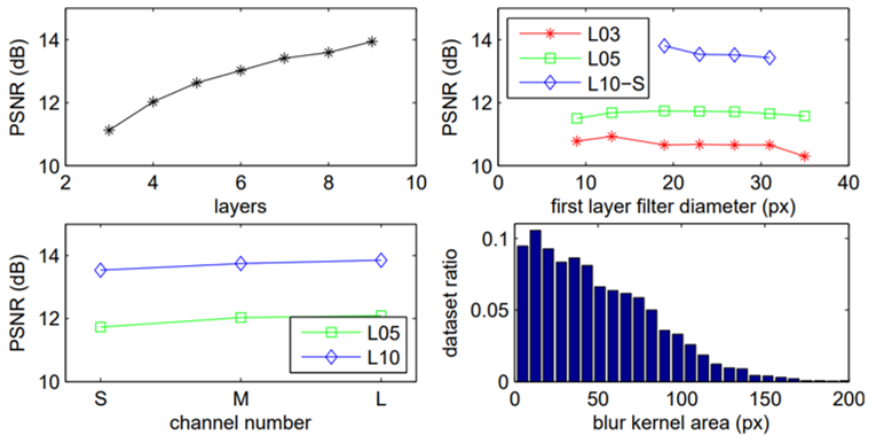

Figure 6.6: performances of different model size and channel number

The training data are sampled from scientific publication containing different content types. Additionally, in order to make the the sampled data more realistic, the authors apply small geometric transformations with bicubic interpolation to simulate camera rotation(since the real world photo is impossible to be exactly parallel to the content plane). Then, an uniform anti-aliased disc is used as kernel to simulate de-focus blur; and the motion blur was generated by a random walk. A sample result is shown below.

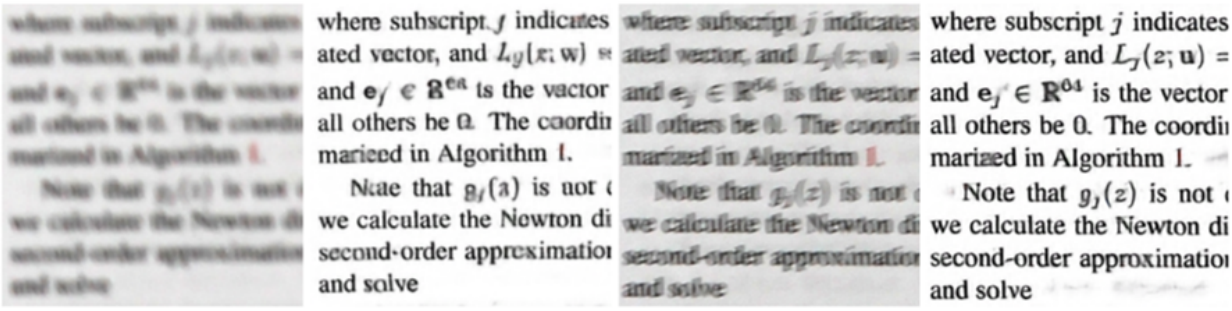

Figure 6.7: sample result

A Short Comment on This Tutorial

In this tutorial, we explored some methods of image deblurring. The conventional methods are extract from book [1], and the others methods are summarized from different research papers. However, the point of this tutorial is to introduce the spirits of each method(i.e., the image prior they utilize) and the core algorithm of the method. Hence, this tutorial *does* omits quite a lot mathematical details. It is highly recommended that one refer to the original paper for a better understanding. Last but not the least, there may be some typos and mistakes in this tutorial. Any comments and suggestion are welcomed. If you have any guidance or problems, please email `r11942060@ntu.edu.tw`. Thank you!

Appendices

Appendix A:

Issues of MATLAB Implementation

In this tutorial, the notation $\mathbf{A}_{op}f$ is frequently used. Generally speaking , f is a matrix and $\mathbf{A}_{op}$ is the *linear operator* on the matrix. A linear operator can be either convolution ($\mathbf{A}_{op}f = K * f$) or matrix multiplication ($\mathbf{A}_{op}f = Af$) of the matrix f. Although the above two representations are mathematically identical, they differ from each other when implementing. Here we address the relationship between them.
Consider a blurred image without noise

$$b = \mathbf{A}f \tag{169}$$

Suppose f is a sharp image of size $R \times K$ and the linear operator $\mathbf{A}$ defines the blurring process by a kernel a of size $r \times k$. The explicit dimension of (69) is

$$\begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,K} \\ \vdots & \ddots & \vdots & \\ b_{R,1} & b_{R,2} & \dots & b_{R,K} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k} \\ \vdots & \ddots & \vdots & \\ a_{r,1} & a_{r,2} & \dots & a_{r,K} \end{bmatrix} * \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,K} \\ \vdots & \ddots & \vdots & \\ b_{f,1} & f_{R,2} & \dots & f_{R,K} \end{bmatrix} \tag{170}$$

where $*$ denotes 2D convolution. In MATLAB, b can be obtained by

```
b = conv2(padarray(f, [r-1 k-1] , 'pre'), a, 'valid')
```

The padding ensures that the result after convolution will be the same size as f. Now, we try to modify the above convolution to matrix multiplication.(`a' = a(end:-1:1, end:-1:1)`)

$$\begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,K} \\ b_{2,1} \\ \vdots \\ b_{R,K} \end{bmatrix} = \begin{bmatrix} a'_{1,1} & \dots & a'_{1,k}, 0 & \dots 0, & a'_{2,1}, & \dots 0 & & \\ 0, & a'_{1,1} & \dots & a'_{1,k}, 0 & \dots 0, & a'_{2,1}, & \dots 0 & \\ 0, & 0, & a'_{1,1} & \dots & a'_{1,k}, 0 & \dots 0, & a'_{2,1}, & \dots 0 \\ \vdots & \ddots & & & & \vdots & & \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{1,2} \\ \vdots \\ f_{1,K-k+1} \\ f_{2,1} \\ \vdots \\ f_{R-r+1,K-k+1} \end{bmatrix} \tag{171}$$

Using a set of new notation $\mathbf{b}_{vec} = \mathbf{a}'_{expanded}\,\mathbf{f}_{vec}$, one can examine the outcome by expanding few terms in above equation. As a result, a 2D convolution can *theoretically* be formulated as matrix multiplication. The reason of doing so is that in some scenarios, the derivation of algorithm is based on such assumption. For example, the conjugate gradient method (54) (57) regards image blurring process as linear equations. $\mathbf{A}$ in equation (56) and (57) correspond to $\mathbf{a}'_{expanded}$ for its blur kernel.

However, consider a case of 2048×2048 sharp image and 20×20 blur kernel. The corresponding $\mathbf{b}_{vec}$, $\mathbf{a}'_{expanded}$, $\mathbf{f}_{vec}$ will be matrices with approximate size $2^{22} \times 1$, $2^{22} \times 2^{22}$, $2^{22} \times 1$ respectively. The size of matrices exceed the limitation in MATLAB. Therefore, the operation $\mathbf{A}f$ in implementation is usually performed as 2D convolution. What's more, the time complexity of 2D convolution is $\mathcal{O}(K^2HW)$. A faster way of doing 2D convolution is transform kernel K and sharp image f to Fourier domain and use point-wise multiplication since $K * f = \hat{K}\hat{f}$. The time complexity is reduced to two fast Fourier transform and a point-wise multiplication $\mathcal{O}(HW(\log H + \log W))$. In fact, MATLAB implementations of $\mathbf{A}f$ in this tutorial is done in either spatial domain or Fourier domain. Performing matrix multiplication in image deblurring is rather unrealistic.

Appendix B:

Iteratively Reweighted Least Squares(IRLS)

According to [10], IRLS is a method to find x that minimize costs of the form

$$\sum_j \rho(A_j x - b_j) \tag{172}$$

The IRLS algorithm is summarized as follows,

Algorithm 1 Iteratively Reweighted Least Squares(IRLS)

Initialization: $\psi_j^0 = 1$

while x^t does not convergence **do**

1. $\bar{A} = \sum_j A_j^T \psi_j^{t-1} A_j$ and $\bar{b} = \sum_j A_j^T \psi_j^{t-1} b_j$, x^t is the solution of $\bar{A}x = \bar{b}$
   $x^t = (A^T \Psi^t A)^{-1} A^T \Psi^t \boldsymbol{b}$

2. $u_j = A_j x - b_j$
   $\psi_j^t(u_j) = \frac{1}{u_j} \frac{d\rho(u_j)}{du}$

end while

For different $\rho(u_j)$, corresponding $\psi(u_j)$ can be calculate. For example, [15] showed that

Beaton and Tukey [4]	$\rho(u) = \begin{cases} \frac{a^2}{6}\left[1-\left(1-\left(\frac{u}{a}\right)^2\right)^3\right], & \lvert u\rvert \le a \\ \frac{a^2}{6}, & \lvert u\rvert > a \end{cases}$	$\psi(u) = \begin{cases} u\left[1-\left(\frac{u}{a}\right)^2\right]^2, & \lvert u\rvert \le a \\ 0, & \lvert u\rvert > a \end{cases}$
Cauchy [33]	$\rho(u) = \frac{b^2}{2}\log\left[1+\left(\frac{u}{b}\right)^2\right]$	$\psi(u) = \frac{u}{1+(u/b)^2}$
Huber [35, Chap. 7]	$\rho(u) = \begin{cases} \frac{1}{2}u^2, & \lvert u\rvert \le c \\ \frac{1}{2}c(2\lvert u\rvert - c), & c < \lvert u\rvert \end{cases}$	$\psi(u) = \begin{cases} 1, & \lvert u\rvert \le c \\ c\frac{u}{\lvert u\rvert}, & \lvert u\rvert > c \end{cases}$

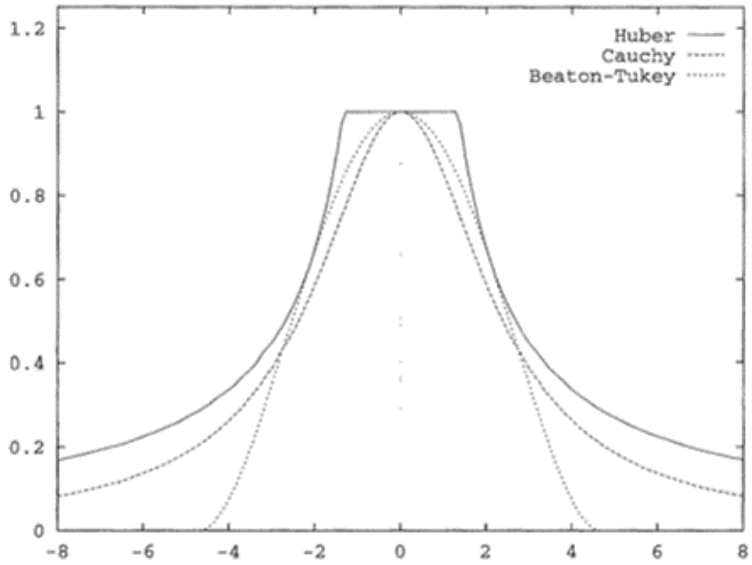

Figure B.1: (top)three different robust loss functions $rho(u)$ and associated ψ functions (bottom) $\psi(u)$ versus scale-normalized u

In the case of L^p norm linear regression, $\rho(u) = |u|^p$. The update after each iteration is

$$\psi^t(u) = |u|^{p-2} \tag{173}$$

The derivation can be found in Wikipedia [18] . Note that it has been shown in [7] that $p = 0.8$ is a proper parameter for natural images.

Reference

[1] M. Bertero, P. Boccacci, C. d. Mol, and M. Bertero. *Introduction to inverse problems in imaging*. CRC Press, 2021.

[2] S. Cho and S. Lee. Fast motion deblurring. *ACM Transactions on Graphics*, 28(5):1–8, Dec. 2009.

[3] J.-J. Ding, W.-D. Chang, Y. Chen, S.-W. Fu, C.-W. Chang, and C.-C. Chang. Image deblurring using a pyramid-based richardson-lucy algorithm. In *2014 19th International Conference on Digital Signal Processing*. IEEE, Aug. 2014.

[4] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *2013 IEEE International Conference on Computer Vision*. IEEE, Dec. 2013.

[5] M. Hradiš, J. Kotera, P. Zemčík, and F. Šroubek. Convolutional neural networks for direct text deblurring. In *Procedings of the British Machine Vision Conference 2015*. British Machine Vision Association, 2015.

[6] A. Kheradmand and P. Milanfar. A general framework for regularized, similarity-based image restoration. *IEEE Transactions on Image Processing*, 23(12):5136–5151, Dec. 2014.

[7] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[8] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR 2011*. IEEE, June 2011.

[9] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3):70, July 2007.

[10] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. In *Lecture Notes in Computer Science*, pages 602–613. Springer Berlin Heidelberg, 2004.

[11] J. Pan, Z. Hu, Z. Su, and M.-H. Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2014.

[12] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016.

[13] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur, 2014.

[14] C.-T. Shen, W.-L. Hwang, and S.-C. Pei. Spatially-varying out-of-focus image deblurring with l1-2 optimization and a guided blur map. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2012.

[15] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, Jan. 1999.

[16] C.-Y. Tseng, S.-J. Wang, C.-W. Chang, P.-C. Chen, C.-C. Chang, and Y.-A. Chen. Digital image restoration for phase-coded imaging systems. In P. Schelkens, T. Ebrahimi, G. Cristóbal, F. Truchetet, and P. Saarikko, editors, *SPIE Proceedings*. SPIE, Apr. 2010.

[17] O. Whyte, J. Sivic, and A. Zisserman. Deblurring shaken and partially saturated images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, Nov. 2011.

[18] Wikipedia. Iteratively reweighted least squares — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Iteratively%20reweighted%20least%20squares&oldid=1008856999, 2023. [Online; accessed 16-January-2023].

[19] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[20] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2013.

[21] K. Zhang, W. Ren, W. Luo, W.-S. Lai, B. Stenger, M.-H. Yang, and H. Li. Deep image deblurring: A survey. *International Journal of Computer Vision*, 130(9):2103–2130, June 2022.