이학 박사 학위논문

Mathematical Approach to Monochromatic Aberration Correction

(단색수차 보정에 대한 수학적 접근)

2020년 2월

서울대학교 대학원

수리과학부

정승원

Mathematical Approach to Monochromatic Aberration Correction

(단색수차 보정에 대한 수학적 접근)

지도교수 강명주

이 논문을 이학 박사 학위논문으로 제출함

2019년 10월

서울대학교 대학원

수리과학부

정승원

정승원의 이학 박사 학위논문을 인준함

2019년 12월

위 원 장	______________________	(인)
부 위 원 장	______________________	(인)
위 원	______________________	(인)
위 원	______________________	(인)
위 원	______________________	(인)

Mathematical Approach to Monochromatic Aberration Correction

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Seungwon Jeong

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

February 2020

Abstract

This thesis introduces efficient and effective methods for solving monochromatic aberration correction problems. The proposed methods are based on Forward-Backward proximal splitting method, which solves the optimization problem by iteratively solving two sub parts for each step: 1. gradient descent and 2. noise removal. Since the gradient descent part has high computational cost, we develop a low-cost implementation of computing aberration operator and its transpose. Then, we propose 6 different methods, which are based on 6 types of different regularization in the noise removal part. In this thesis, we perform experiments on the proposed image restoration methods. In the experiments, we use synthetic images generated by point spread functions (PSFs), which emulate the effects of monochromatic aberration in modern digital cameras.

Key words: Image processing, monochromatic aberrations, aberration correction, spatially variant deconvolution
Student Number: 2012-23030

Contents

Chapter 1

Introduction

In this thesis, we discuss efficient and effective methods to restore a clean image (or unknown) u from an observed image s given by spatially variant convolution (s.v.c.) with a given point spread function (PSF),

$$s(x) = (Lu)(x) + n(x) = \int_\Omega \mathrm{PSF}(y, x)u(y)\, dy + n(x), \tag{1.0.1}$$

where $\Omega \subset \mathbb{R}^2$ is the image domain and $x \in \Omega$, and n is an additive (Gaussian) noise. If we use the concept of mathematical kernels, we can rewrite (1.0.1) as

$$s(x) = \int_\Omega k(x, y)u(y)\, dy + n(x),$$

where $k(x, y) = \mathrm{PSF}(y, x)$ is the spatially variant convolution kernel. The ideal (aberration-free) kernel for an ideal optical system would be Dirac's delta distributions,

$$k(x, y) = \delta_x(y),$$

so that

$$s(x) = u(x) + n(x),$$

which is just a denoising problem. However, in real world, PSFs and kernels have supports of positive measure because of aberrations (Figure 1.1, 1.2, 1.3). Therefore, our restoration problem becomes a spatially variant deconvolution (s.v.d.) problem.

This thesis is organized as follows. Chapter 2 consists of three sections. In the first section, we present frequently used numerical approximation methods for (1.0.1). In the second section, we introduce basic Fourier Optics theory which is essential to create lens aberration PSFs in the real world. In the last section, we present mathematical preliminaries. The last section has three subsections:

1. Basic Properties of spatially variant convolution (s.v.c.) Operators
   This section includes basic results of operator analysis on s.v.c. operators.

2. Regularizations in Inverse Problems
   In this section, we present Bayesian approach to our problem and mathematical derivation of model problem. Also we introduce local and nonlocal regularizations, which are frequently used in image analysis.

3. Convex Optimization Theory
   In the last section, we introduce preliminaries of convex optimization. This section provides our main tool to investigate convergence of proposed algorithms.

In Chapter 3, we start with a brief explanation about the reason why we need low cost implementation. This chapter consists of two sections. The first part is about how to reduce required computing resources of s.v.c. operators. The second part includes our main algorithm, which is Forward Backward Splitting (FBS). After providing a simple proof of convergence, we present Split Bregman method, which can be a powerful tool to find a minimizer of l_1 regularization problems. In the last section, we exhibit detailed implementation methods for 5 different regularizers.

In Chapter 4, we show experimental results of proposed methods. This chapter consists of two sections. In the first section, we present experimental environment and implementation details including programming code for s.v.c. operator. The second section has two subsections. In the first subsection, we display test image set and generate synthetically blurred images. The second subsection, main part of this chapter, shows main results of this thesis. In the beginning of the subsection, we present individual results and parameter selection for proposed algorithms. Then we display the detailed comparison between the results of proposed methods.

In the last chapter, we provide conclusion of this thesis and future work.

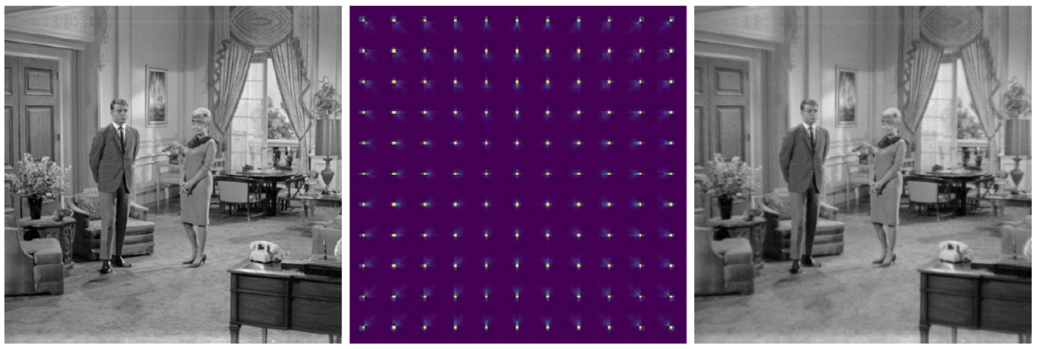


Figure 1.1: An example of monochromatic aberrations. The clean image (left), illustration of the aberration PSF (middle), and the aberrated image (right).

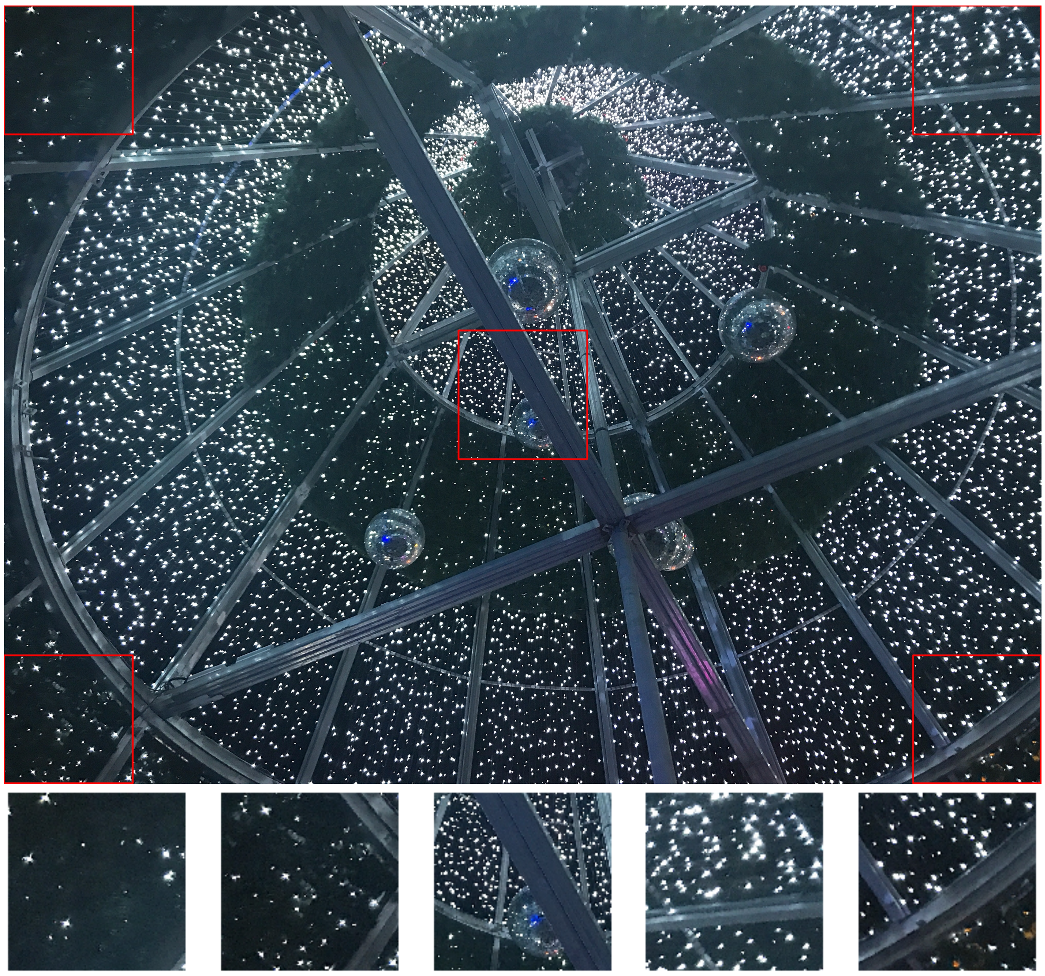
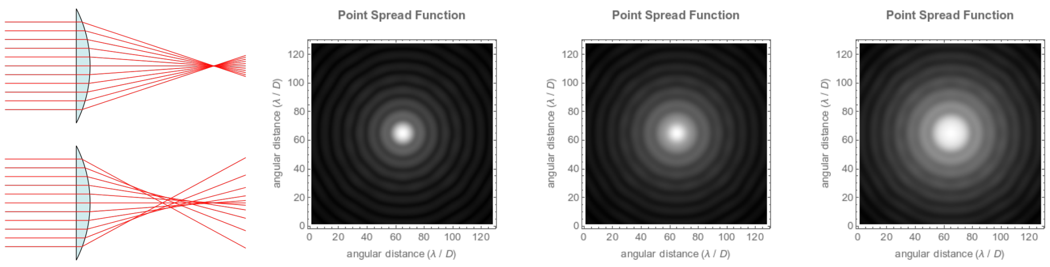
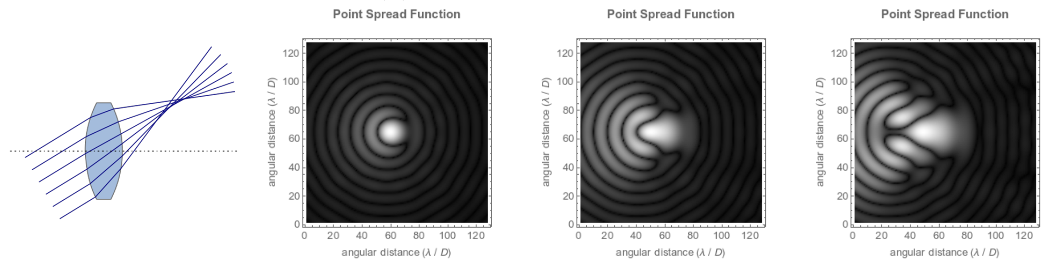


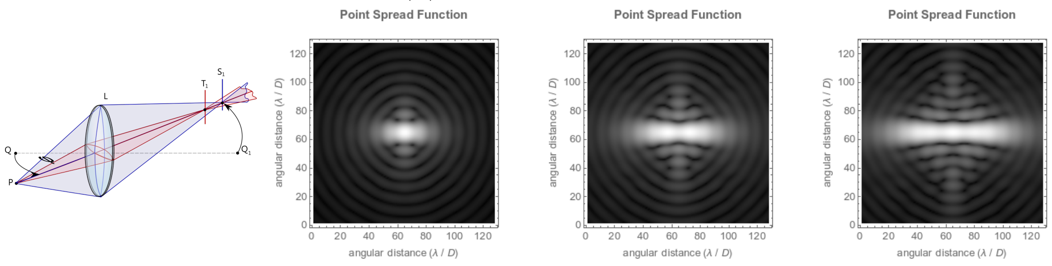
Figure 1.2: Another example of monochromatic aberrations. The photo was taken by the author in Busan, 2019.



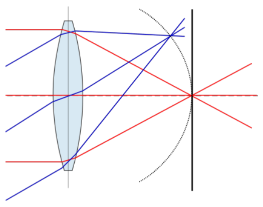
(a) Spherical aberration



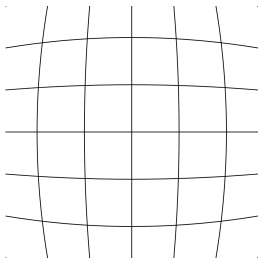
(b) Coma aberration



(c) Astigmatism



(d) Field curvature



(e) Distortion

Figure 1.3: Illustration of Seidel's five aberrations and corresponding PSFs. If we use PSFs, we can describe blurring and defocusing effects caused by monochromatic aberrations.

Chapter 2

Related Works

In this chapter, we review previous research on monochromatic aberrations. In Section 2.1, we present previous research on numerical methods that approximate (1.0.1) and explain the reason why we use direct computation as the approximation method in this thesis. In Section 2.2, we introduce basic theory of Fourier optics to explain how to estimate PSFs in a given optical system. In the last section, we cover mathematical preliminaries to be used in the rest part of this thesis.

2.1 Approximation Methods

In this section, we present numerical approximation methods for (1.0.1). Each method has its own pros and cons. In what follows, we follow the presentation given in Escande-Weiss [12].

2.1.1 Methods

Direct Discretization (Direct Computation)

Consider a square shaped image domain $\Omega = [0, N]^2$ with side length $N \in \mathbb{N}$. Then, it is natural to consider direct discretization which can be formulated as

$$(Lu)(x) = \sum_{y \in \Omega_N} k(x, y)u(y),$$

where $k(x, y) = \text{PSF}(y, x)$ is the s.v.c. kernel and $\Omega_N = \{1 \leq i \leq N\}^2$ is the discrete image domain grid. The strong feature of this method is that it approximates general s.v.d. problems accordingly. Angel-Jain [1] used this approximation with conjugate gradient method to solve s.v.d. problems. However, it has not been favored because of its heavy computational cost given by $\mathcal{O}(N^4)$.

Composition of Diffeomorphism and Convolution

Previous research such as Yue et al. [26] and Zhang et al. [28] used composition of diffeomorphism and convolution to describe (1.0.1). Another famous name of this method is *splitting and warping* technique. The technique uses coordinates transformation into polar coordinates system (warping by diffeomorphism), $u(x_1, x_2) \to f(r, \theta)$ and $k(x, y) \to g(r_x, \theta_x, r_y, \theta_y)$. If we utilize radial symmetry of PSFs in spatial coordinates, we have $g(r_x, \theta_x, r_y, \theta_y) = g(r_x, r_y, \theta_x - \theta_y)$. If we further assume g is slowly varying with respect to r_x, we approximate g by spatially invariant convolution kernels $g_i(r_x - r_y, \theta_x - \theta_y)$ on corresponding strips $r_{i-1} \leq r \leq r_i, i = 1 \ldots m$. This process is described in Figure 2.1. This method reduces computational cost of (1.0.1) drastically in the sense of memory usage and computing convolution operations. However, since *warping and splitting* technique requires radial symmetry on the optical system, we can't apply the technique to certain optical systems such as multi-lens array camera, because radial symmetry is broken. Moreover, warping process uses interpolation, which may ruin noise robustness of the recovery process. Therefore, the technique presents obvious weak points.

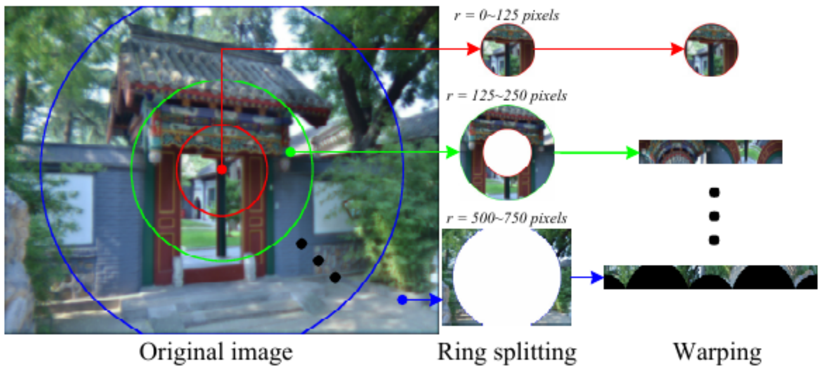


Figure 2.1: *Splitting and warping* technique from: Yue et al. [26], page 1686.

Approximation by Separable Kernels

We may also use approximation by separable kernels like in Angel [1]. A kernel k is separable if there exist k^1, k^2 such that

$$k(x, y) = k^1(x_1, y_1)k^2(x_2, y_2),$$

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. If k is approximated by separable kernels, computational cost of convolution operation reduces to $\mathcal{O}(N^3)$. However, there are also s.v.c. kernels which are not separable, for example, spatially variant motion blur kernels.

Piecewise Convolutions

Another approximation method was proposed in Heide et al. [17] and Bardsley et al. [3]. The method assumes PSFs are nearly invariant in small regions of the image (Figure 2.2). The method subdivides the image into (overlapping) small regions and takes spatially invariant convolutions by approximate kernels.

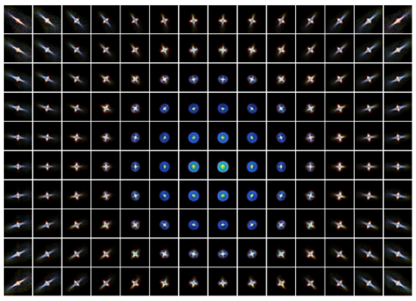


Figure 2.2: Patchwise estimation of PSFs from: Heide et al. [17].

However, the performance is poor when PSFs are not slowly varying with respect to positions on the image plane.

2.1.2 Methods Comparison and Conclusion

In Escande-Weiss [12], the authors compared restoration performance of various numerical methods which approximate spatially variant convolutions. Among the comparisons, they pointed out that the performance gap between direct computation and approximation methods, including *warping and splitting* and patch-wise approximations, ranges up to 6dB in peak signal-to-noise ratio (PSNR) values. For a reference image A and an observed (or restored) image I, the PSNR value is defined by

$$\mathrm{PSNR} = -10\log_{10}\left(\frac{MAX_I^2}{MSE}\right),$$

where $MAX_I(=255)$ is the maximum intensity value and MSE is the mean squared error of I from A,

$$MSE = \frac{1}{|\Omega|}\sum_{x\in\Omega}|I(x)-A(x)|^2.$$

As we mentioned in the previous subsection, direct computation has not been commonly used because it has heavy computational cost. However, when we deal with monochromatic aberrations, we may assume supports of the kernel and the PSF, given by $\operatorname{supp} k(x,\cdot)$ and $\operatorname{supp}\mathrm{PSF}(x,\cdot)$ on each image position $x\in\Omega_N$, are contained in a box around x, $x+B$ with $B=[-w,w]^2$ for some $w\in\mathbb{N}$. In this thesis, we use this assumption to develop an efficient way to perform direct computation (Chapter 3).

2.2 Basic Fourier Optics

We shall discuss optical theory on aberrations to compute PSFs. Hence we present basic Fourier Optics in this section. Key idea of Fourier Optics is considering a wave function as a composition of planar wave functions. If we use Fourier transform, we can find spectrum coefficients of corresponding planar wave functions. This section is based on Breckinridge-Voelz [6] and Goodman [15].

2.2.1 Wavefront Optical Path Difference, $W(x,y)$

In this section, we discuss wavefront optical path difference. Ideal optical system has spherical wavefronts converging to points on the image plane. However, because of the curvature and thickness of lenses in the optical system, wavefronts are aberrated and there would be an optical path difference (OPD) between spherical and aberrated wavefronts (Figure 2.3). Since lights are electromagnetic waves, the optical path difference causes phase difference. Then, the resulting phase difference raises interference pattern on image plane.

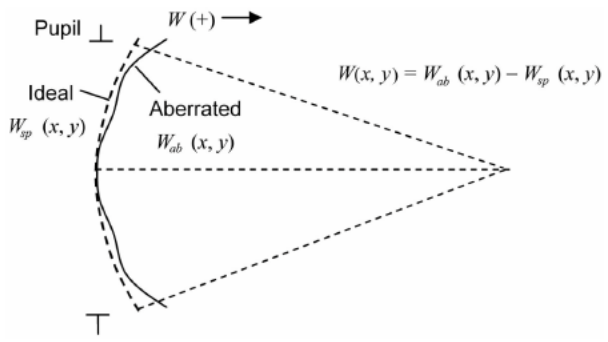


Figure 2.3: Ideal spherical (sp) and aberrated (ab) wavefronts from: Breckinridge et al. [6], page 142.

When the optical system has radial symmetry, Seidel aberration polynomials can be used to describe OPD, $W(x, y)$. Let $(\hat{u}, \hat{v})$ be the normalized image coordinates and $(\hat{x}, \hat{y})$ be the normalized exit pupil coordinates. Because of symmetry, W can be expressed in terms of rotationally invariant combinations of image coordinates and exit pupil coordinates. Those combinations are $\hat{x}^2 + \hat{y}^2$ (modulus of exit pupil coordinates), $\hat{x}\hat{u} + \hat{y}\hat{v}$ (inner product), $\hat{u}^2 + \hat{v}^2$ (modulus of image plane coordinates). We may rotate both exit pupil and image planes such that $(\hat{u}_0, 0) = (\hat{u}, \hat{v})$. Then we expand $W(\hat{u}_0; \hat{x}, \hat{y}) = W(\hat{x}^2 + \hat{y}^2, \hat{x}\hat{u}_0, \hat{u}_0^2)$ to get

$$\begin{aligned} W(\hat{u}_0; \hat{x}, \hat{y}) =& D_f(\hat{x}^2 + \hat{y}^2) + S(\hat{x}^2 + \hat{y}^2)^2 + C(\hat{x}^2 + \hat{y}^2)\hat{x}\hat{u}_0 \\ &+ A\hat{x}^2\hat{u}_0^2 + F(\hat{x}^2 + \hat{y}^2)\hat{u}_0^2 + D\hat{x}\hat{u}_0^3 \\ &+ \text{higher order terms}, \end{aligned} \tag{2.2.1}$$

which is called Seidel aberration polynomial. Note that lowest order terms are neglected since they represent tilt $(\hat{x}\hat{u}_0)$ and constant phase shift $(\hat{u}_0^2)$, which do not affect the shape of the wavefront. Meanings of coefficients in (2.2.1) are described in Table 2.1 and they (except for defocus) are called Seidel's five aberrations. Bociort-Kross [5], Simpkins [23] and Mikš [19] explain how to compute Seidel aberration coefficients.

Coefficient	Meaning	Corresponding term
D_f	defocus	$D_f(\hat{x}^2+\hat{y}^2)$
S	spherical aberration	$S(\hat{x}^2+\hat{y}^2)^2$
C	coma aberration	$C(\hat{x}^2+\hat{y}^2)\hat{x}\hat{u}_0$
A	astigmatism	$A\hat{x}^2\hat{u}_0^2$
F	field curvature	$F(\hat{x}^2+\hat{y}^2)\hat{u}_0^2$
D	distortion	$D\hat{x}\hat{u}_0^2$

Table 2.1: Optical meanings of Seidel polynomial coefficients.

If the optical system is not radially symmetric, we generally use Zernike aberration polynomials. Then the OPD $W(x,y)$ can be rewritten as

$$W(\hat{u}_0,\hat{v}_0;\rho,\theta)=\sum_{|m|\le n} C_n^m(\hat{u}_0,\hat{v}_0)Z_n^m(\rho,\theta),$$

where $(\hat{x},\hat{y})=(\rho\cos\theta,\rho\sin\theta)$. Zernike aberration polynomials are mutually orthogonal,

$$\langle Z_n^m, Z_{n'}^{m'}\rangle=\frac{\varepsilon_m\pi}{2n+2}\delta_{n,n'}\delta_{m,m'},$$

where ε_m, called Neumann factor, is defined as 2 if $m=0$ and 1 if $m\neq 0$, and $\delta_{i,j}$ is Kronecker's delta symbol. Zernike aberration polynomials are presented in Table 2.2. The relationship between Seidel and Zernike aberration coefficients are explained in Tyson [25] and Conforti [11].

	Meaning	Polynomial
Z_0^0	Piston	1
Z_1^{-1}	Tilt (vertical tilt)	$2\rho\sin\theta$
Z_1^1	Tip (horizontal tilt)	$2\rho\cos\theta$
Z_2^{-2}	Oblique astigmatism	$\sqrt{6}\rho^2\sin 2\theta$
Z_2^0	Defocus	$\sqrt{3}(2\rho^2-1)$
Z_2^2	Vertical astigmatism	$\sqrt{6}\rho^2\cos 2\theta$
Z_3^{-3}	Vertical trefoil	$\sqrt{8}\rho^3\sin 3\theta$
Z_3^{-1}	Vertical coma	$\sqrt{8}(3\rho^3-2\rho)\sin\theta$
Z_3^1	Horizontal coma	$\sqrt{8}(3\rho^3-2\rho)\cos\theta$
Z_3^3	Oblique trefoil	$\sqrt{8}\rho^3\cos 3\theta$
Z_4^{-4}	Oblique quadrafoil	$\sqrt{10}\rho^4\sin 4\theta$
Z_4^{-2}	Oblique secondary astigmatism	$\sqrt{10}(4\rho^4-3\rho^2)\sin 2\theta$
Z_4^0	Primary spherical	$\sqrt{5}(6\rho^4-6\rho^2+1)$
Z_4^2	Vertical secondary astigmatism	$\sqrt{10}(4\rho^4-3\rho^2)\cos 2\theta$
Z_4^4	Vertical quadrafoil	$\sqrt{10}\rho^4\cos 4\theta$

Table 2.2: Zernike aberration polynomials.

2.2.2 Pupil and Amplitude Transfer Functions

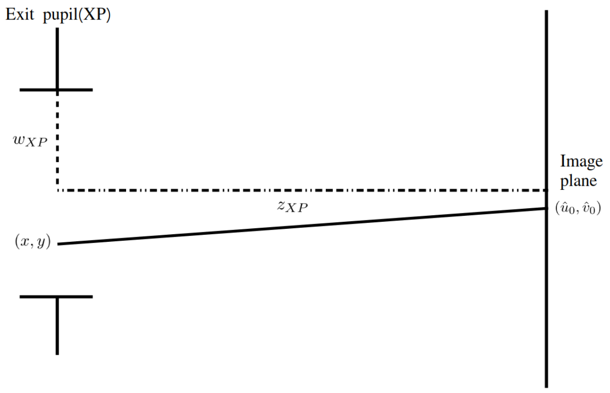


Figure 2.4: Definitions of dimension parameters.

In this section, we introduce the concept of pupil and amplitude transfer functions. Let $(\hat{u}_0, \hat{v}_0)$ be the normalized image coordinates and $W(\hat{u}_0, \hat{v}_0; \hat{x}, \hat{y})$ be the OPD function of the optical system. It holds that $\hat{x} = x/w_{XP}, \hat{y} = y/w_{XP}$, where w_{XP} is the width of circular exit pupil. Then pupil function $P(\hat{u}_0, \hat{v}_0; x, y)$ is defined by

$$P(\hat{u}_0, \hat{v}_0; x, y) = \operatorname{circ}\left(\frac{\sqrt{x^2+y^2}}{w_{XP}}\right) \exp\left[-ikW\left(\hat{u}_0, \hat{v}_0; \frac{x}{w_{XP}}, \frac{y}{w_{XP}}\right)\right],$$

where k is the wave number of the light. Pupil function describes the phase shift and the existence of waves from pupil position (x, y). For an aberration free optical system, Fraunhofer diffraction pattern of the exit pupil is given by

$$\begin{aligned} h(u, v; \xi, \eta) &= h(u-\xi, v-\eta) \\ &= \frac{A}{\lambda z_{XP}} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} P(x, y) \exp\left\{-i\frac{2\pi}{\lambda z_{XP}}[(u-\xi)x + (v-\eta)y]\right\} dxdy, \end{aligned}$$

where z_{XP} is the exit pupil distance, λ is the wavelength of the light, (u, v), (ξ, η) are the normalized image coordinates. The formula can be easily derived from paraxial approximation,

$$|f| \approx |f_Z|,$$

and the geometric relations between (u, v) and (ξ, η). We define $U_g(\xi, \eta)$ to be the ideal image wave function for the perfect imaging system (clear image), we get the formula for the image wave function $U_i(u, v)$,

$$U_i(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(u-\xi, v-\eta)U_g(\xi,\eta)\, d\xi d\eta,$$

that is convolution of U_g under the kernel h. If we use Fourier transform, we see that

$$G_i(f_X, f_Y) = H(f_X, f_Y)G_g(f_X, f_Y),$$

where G_i, G_g are Fourier transforms of U_i, U_g respectively and H is Fourier transform of h. Hence it holds that

$$\begin{aligned} H(f_X, f_Y) &= \mathcal{F}\left\{\frac{A}{\lambda z_{XP}}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} P(x,y)\exp\left\{-i\frac{2\pi}{\lambda z_{XP}}(ux+vy)\right\} dxdy\right\} \\ &= (A\lambda z_{XP})P(-\lambda z_{XP} f_X, -\lambda z_{XP} f_Y) \\ &= P(-\lambda z_{XP}, -\lambda z_{XP}), \end{aligned}$$

since we can put $A\lambda z_{XP} = 1$ for a mathematical convenience. If we consider U_i as the sum of planar waves (Huygens' principle), H carries amplitudes information of planar wave spectrum in frequency domain. In this reason, H is called amplitude transfer function. Similar argument with fixed image coordinates $(\hat{u}_0, \hat{v}_0)$, for an aberration present optical system, can be done by considering spatially variant convolution as the sum of spatially invariant convolutions multiplied by Dirac's delta distribution, in other words,

$$U_i(u,v) = (h_{u,v} * U_g)(u,v),$$

where $h_{u,v}$ is the convolution kernel at (u, v). Therefore, we have

$$H(\hat{u}_0, \hat{v}_0, f_X, f_Y) = P(\hat{u}_0, \hat{v}_0; -\lambda z_{XP} f_X, -\lambda z_{XP} f_Y).$$

Note that the amplitude transfer function H depends also on $(\hat{u}_0, \hat{v}_0)$, not only on (x, y). This results spatial variety of PSFs.

2.2.3 Point Spread Functions

We now explain how to generate PSFs with given apertures. Wave intensity (or energy) is proportional to the square of the wave amplitude. Therefore, PSFs can be calculated using amplitude transfer function H,

$$\mathrm{PSF}(\hat{u}_0, \hat{v}_0; x, y) = \left|\mathcal{F}^{-1}(H(\hat{u}_0, \hat{v}_0, f_X, f_Y))\right|^2$$

$$= \left|\mathcal{F}^{-1}(P(\hat{u}_0, \hat{v}_0; -\lambda z_{XP} f_X, -\lambda z_{XP} f_Y))\right|^2.$$

Note that PSF's are determined by wavelength λ, exit pupil radius W_{XP}, exit pupil distance Z_{XP}, and the normalized position on image plane $(\hat{u}_0, \hat{v}_0)$. In most cases, Z_{XP} is chosen to be focal length of lens aperture,

$$Z_{XP} = f,$$

and we write exit pupil diameter respect to focal length to represent the aperture. For example,

$$\begin{aligned} f/1.8 &\implies W_{XP} = f/(1.8 \cdot 2), \\ f/2.2 &\implies W_{XP} = f/(2.2 \cdot 2), \\ f/3 &\implies W_{XP} = f/(3 \cdot 2). \end{aligned}$$

For the experiments, we assumed $f/3$ aperture with $f = 15mm$, CCD sensor pixel pitch: $2\mu m$, sensor pixels(image size): 512×512. Sensor pixels are chosen in small sizes in order to reduce computational cost. We generated two types of PSFs: big and small. For the big PSF, we use coefficients given by $(D_f, S, C, A, F, D) = (0, 0.5, 1.3, 1.0, 0, 0)$. On the other hand, for the small PSF, we use coefficients given by $(D_f, S, C, A, F, D) = (0, 0.1, 1.0, 0.3, 0, 0)$. In order to reduce computational time for PSF generation, PSFs are generated only on the first quadrant of the image plane and reflected onto other three quadrants. Seidel aberration coefficients are chosen to mimic modern digital camera. PSFs' shapes are illustrated in Figure 2.5.

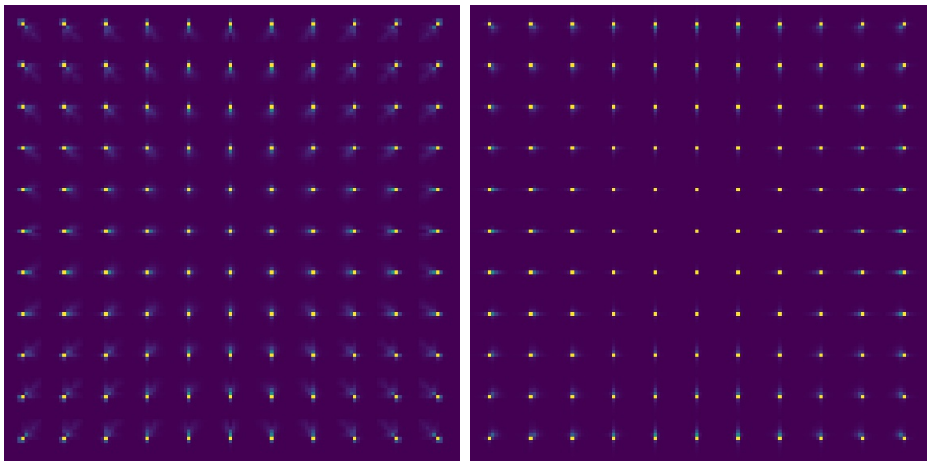


Figure 2.5: Illustration of big (left) and small (right) PSFs.

2.3 Mathematical Preliminaries

In this section, we introduce mathematical preliminaries used in this thesis.

2.3.1 Basic Properties of s.v.c. Operators

We start with basic mathematical properties of s.v.c. operators. Let $L : L^2(\Omega) \to L^2(\Omega)$ be the s.v.c. blur operator associated with the kernel $k \geq 0$,

$$Lu(x) = \int_\Omega k(x,y)u(y)\,dy.$$

Equipped with the standard inner product $\langle u, v\rangle_\Omega = \int_\Omega u(x)v(x)\,dx$, L has the following property.

Lemma 2.3.1. *The transpose L^t of a blur operator L is given by*

$$L^t u(x) = \int_\Omega k(y,x)u(y)\,dy = \int_\Omega \mathrm{PSF}(x,y)u(y)\,dy.$$

Proof. By definition,

$$\begin{aligned}
\langle L^t u, v\rangle_\Omega &= \langle u, Lv\rangle_\Omega \\
&= \int_\Omega u(x)\left(\int_\Omega k(x,y)v(y)\,dy\right)dx \\
&= \int_\Omega\int_\Omega k(x,y)u(x)v(y)\,dydx \\
&= \int_\Omega\left(\int_\Omega k(x,y)u(x)\,dx\right)v(y)\,dy,
\end{aligned}$$

for all $v \in \Omega$ by Fubini's theorem. $\square$

Let us investigate operator 2-norm of L. Simple composition gives

$$L^t Lu(x) = \int_\Omega k(y,x)Lu(y)\,dy = \int_\Omega k(y,x)\int_\Omega k(y,z)u(z)\,dz\,dy,$$

and hence,

$$L^t Lu(x) = \int_\Omega\left(\int_\Omega k(y,x)k(y,z)\,dy\right)u(z)\,dz. \tag{2.3.2}$$

So the s.v.c. kernel k_{L^tL} of L^tL is given by

$$k_{L^tL}(x,z) = \int_\Omega k(y,x)k(y,z)\,dy.$$

For the spatially invariant convolution kernels satisfying $k(x,y) = f(x-y)$,

$$k_{L^tL}(x,z) = \int_\Omega f(y-x)f(y-z)\,dy = \int_{\Omega+z} f(t+z-x)f(t)\,dt = (\bar{f}*f)(x-z),$$

where $\bar{f}$ is the reflection of f, $\bar{f}(x) = f(-x)$. Now we compute the operator norm of L.

Lemma 2.3.2. *Let L be the blur operator defined as above. Then it holds that*

$$\|L\|_2 \le \max_y \sqrt{\int_\Omega k_{L^tL}(z,y)\,dz}.$$

Therefore,

$$\|L^tL\|_2 \le \max_y \int_\Omega k_{L^tL}(z,y)\,dz. \tag{2.3.3}$$

Proof. First we see that

$$\|Lu\|_2^2 = \int_\Omega (Lu(x))^2\,dx = \int_\Omega \left(\int_\Omega k(x,y)u(y)\,dy\right)^2 dx. \tag{2.3.4}$$

Let us define $C(x) = \int_\Omega k(x,y)\,dy$. Then by Jensen's inequality,

$$\begin{aligned}
\int_\Omega \left(\int_\Omega k(x,y)u(y)\,dy\right)^2 dx &= \int_\Omega \left(C(x)\int_\Omega \frac{k(x,y)}{C(x)}u(y)\,dy\right)^2 dx \\
&\le \int_\Omega C^2(x)\int_\Omega \frac{k(x,y)}{C(x)}u^2(y)\,dydx \\
&= \int_\Omega C(x)\int_\Omega k(x,y)u^2(y)\,dydx \\
&= \int_\Omega \left(\int_\Omega C(x)k(x,y)\,dx\right)u^2(y)\,dy.
\end{aligned}$$

Therefore, we have

$$\|Lu\|_2 \le \left(\max_y \sqrt{\int_\Omega C(x)k(x,y)\,dx}\right)\|u\|_2.$$

In addition,

$$\begin{aligned}\int_\Omega C(x)k(x,y)\,dx &= \int_\Omega \left(\int_\Omega k(x,z)\,dz\right) k(x,y)\,dx \\ &= \int_\Omega \left(\int_\Omega k(x,z)k(x,y)\,dx\right) dz \\ &= \int_\Omega k_{L^tL}(z,y)\,dz.\end{aligned}$$

□

Above lemma can be used to determine Lipschitz constant of L^tL.

2.3.2 Regularizations in Inverse Problems

In this section, we introduce regularizations in inverse problems. We start with the derivation of regularizations. Suppose that we want to find a least square solution u to the discrete version of (1.0.1),

$$\hat{u} = \underset{u \in l_2(\Omega_N)}{\operatorname{argmin}} \|s - Lu\|^2. \tag{2.3.5}$$

Since the noise $n = s - Lu$ follows Gaussian distribution, least square method it maximizes log-likelihood probability $\log p(s|u)$,

$$\begin{aligned}\log p(s|u) &= \log\left[\Pi_{x\in\Omega_N} A \exp(-|s(x) - Lu(x)|^2/(2\sigma^2))\right] \\ &= |\Omega_N| \log A - \frac{1}{2\sigma^2} \sum_{x\in\Omega_N} |s(x) - Lu(x)|^2,\end{aligned}$$

where $\sigma > 0$ is the standard deviation of Gaussian distribution and $A > 0$ is the normalization constant. In this reason, least square method is a maximum likelihood approach. On the other hand, the optimal condition for (2.3.5) is

$$L^tLu = L^ts,$$

and it may be an ill-posed problem depending on s.v.c. kernel k associated to L. In order to overcome the ill-posedness, we use maximum a posteriori (MAP) approach. According to Bayes' rule, a posteriori probability satisfies

$$p(u|s) \propto p(s|u)p(u).$$

Therefore, MAP approach to our problem is equivalent to

$$\hat{u} = \underset{u \in l_2(\Omega_N)}{\operatorname{argmax}}\, p(s|u)p(u). \tag{2.3.6}$$

In the equation above, $p(u)$ is called a prior probability of u, in other words, it is the probability of u to be in a class of clean images. If we take logarithm on objective functional (probability) of (2.3.6), we get

$$\hat{u} = \underset{u \in l_2(\Omega_N)}{\operatorname{argmin}} \left(-\log p(u) + \frac{1}{2\sigma^2} \sum_{x \in \Omega_N} |s(x) - Lu(x)|^2 \right).$$

In continuous version, it reads

$$\hat{u} = \underset{u \in L^2(\Omega)}{\operatorname{argmin}} \left(-\sigma^2 \log p(u) + \frac{1}{2} \int_\Omega (s(x) - Lu(x))^2 \, dx \right).$$

There are various types of priors such as

- Gaussian prior
$$p(u(x)) = A \exp(-|\nabla u(x)|^2/(2a^2)),$$
- Laplace prior
$$p(u(x)) = \exp(-\lambda |\nabla u(x)|),$$
- l_1 prior
$$p(u(x)) = \exp(-\lambda |u(x)|),$$

and so on. If we take (negative) logarithm to priors, we get regularizers

- H^1 regularizer
$$J(u) = \frac{1}{2} \int_\Omega |\nabla u(x)|^2 \, dx,$$
- TV (Rudin-Osher-Fatemi [22]) regularizer
$$J(u) = \int_\Omega |\nabla u(x)| \, dx,$$

  where $\int_\Omega |\nabla u(x)| \, dx$ is defined in distribution sense,
$$\int_\Omega |\nabla u(x)| \, dx := \sup \left\{ \int_\Omega u \operatorname{div} \phi \, dx : \phi \in C_0^1(\Omega)^2, |\phi|_{L^\infty(\Omega)} \le 1 \right\},$$

- L^1 regularizer
$$J(u) = \int_\Omega |u(x)|\,dx,$$
- L^2 (Tikhonov [24]) regularizer
$$J(u) = \int_\Omega |u(x)|^2\,dx$$

correspond to each prior. If we use notion of regularizers, we may rewrite MAP approach to the problem as our model equation:

$$\hat{u} = \operatorname*{argmin}_{u\in X}\left(\mu J(u) + \frac{1}{2}\int_\Omega (s(x) - Lu(x))^2\,dx\right), \tag{2.3.7}$$

for some solution space X and regularization parameter $\mu > 0$. The optimal condition of (2.3.7) is given by

$$0 \in \mu\partial J(u) + L^t(Lu - s),$$

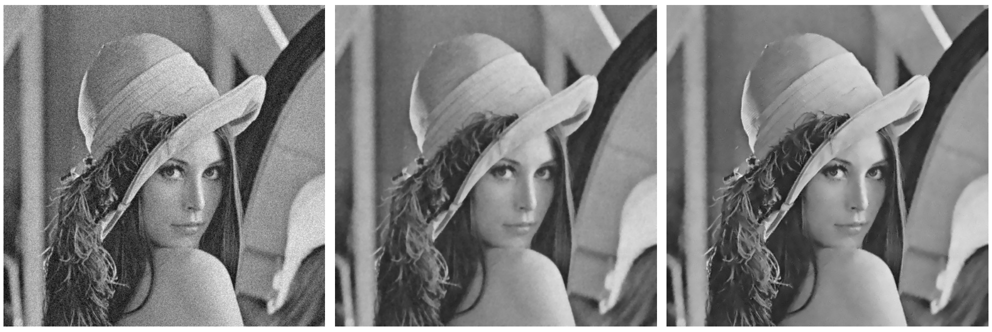
where the subdifferential ∂J of J will be introduced in Section 2.3.3. We present more regularizers in the rest of this section.

Local Regularizations

Local regularizers assume sparsity on the local derivative ∇u or the value u of the solution. Tikhonov [24] suggested regularization methods for ill-posed problems. For the historical backgrounds, refer to Groetsch [16]. H^1 regularizer is the most famous local regularizer. For H^1 regularizer, the (sub)differential of $J(u) = \frac{1}{2}\int_\Omega |\nabla u(x)|^2\,dx$ is $-\Delta u$, which can be written by

$$-(\Delta u)_{i,j} = 4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1},$$

in discrete version. In the case of L^2 regularizer which is given by $J(u) = \frac{1}{2}\int_\Omega u(x)^2\,dx$, the (sub)differential of J is given by u itself. Therefore such local regularizations have diagonally dominant differentials, which are close to the identity operator. If we add those diagonally dominant operators, which are close to the identity operator, we may relax ill-posedness of our problem. However, local regularization methods suffer from smoothing effects (Figure 2.6).



(a) Noisy image (b) Restored with TV (c) Restored with NLH^1

Figure 2.6: Denoising results with TV (local) and NLH^1 (non-local) regularizations.

Non-local Regularizations

In this subsection, we introduce non-local regularizers. Non-local regularizers have texture enhancing properties and is frequently used in deblurring problems. History of non-local regularizers starts with non-local means filter (NLM filter) for denoising problems proposed in Buades et al. [7],

$$NL[u](x) = \frac{1}{C(x)} \int_{\Omega} w(x,y)u(y)\,dy,$$

where $w(x,y) = \exp\left(\frac{(G_a * |u(x+\cdot) - u(y+\cdot)|^2)(0)}{h^2}\right)$ is the non-local weight of y with respect to x, and $C(x) = \int_{\Omega} w(x,z)\,dz$ is a normalizing constant, and G_a is a Gaussian kernel with standard deviation a, and h is the filtering parameter. Figure 2.7 illustrates computation of non-local weights. It appears that non-local weight map captures textures in the image accordingly.

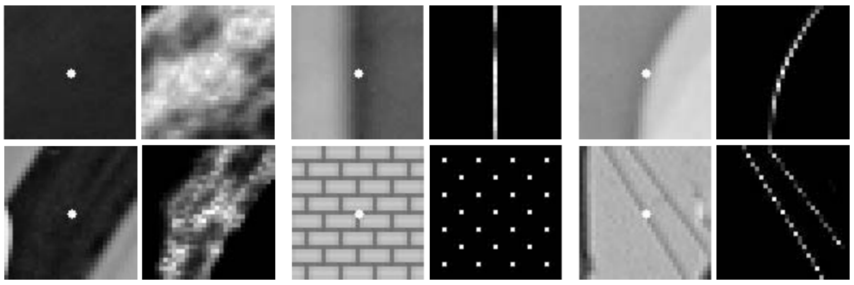


Figure 2.7: Illustration of non-local weights from Buades [7]. Each pair consists of original image (left) and weight map (right) with respect to the central point (white point) of the image.

Gilboa-Osher [13] generalized NLM filter by defining non-local operators. The non-local derivative $\nabla_w u : \Omega \times \Omega \to \mathbb{R}$ of a function u with respect to the weight w is defined by

$$\nabla_w u(x,y) = (\nabla_w u(x))(y) = \partial_y u(x) = (u(y) - u(x))\sqrt{w(x,y)}, \quad x, y \in \Omega.$$

Let $p : \Omega \times \Omega \to \mathbb{R}$ be a vector. We define non-local divergence div_w with respect to the weight w using the following relation

$$\langle u, \mathrm{div}_w p\rangle_\Omega = -\langle \nabla_w u, p\rangle_{\Omega\times\Omega}.$$

Note that

$$\begin{aligned}
\langle \nabla_w u, p\rangle_{\Omega\times\Omega} &= \int_\Omega \int_\Omega \nabla_w u(x,y)p(x,y)\,dxdy \\
&= \int_\Omega \int_\Omega (u(y) - u(x))\sqrt{w(x,y)}p(x,y)\,dx\,dy \\
&= \int_\Omega u(y) \int_\Omega p(x,y)\sqrt{w(x,y)}\,dx\,dy \\
&\quad - \int_\Omega u(x) \int_\Omega p(x,y)\sqrt{w(x,y)}dy\,dx \\
&= \int_\Omega u(x) \int_\Omega (p(y,x) - p(x,y))\sqrt{w(x,y)}\,dy\,dx.
\end{aligned}$$

Therefore, $\mathrm{div}_w p(x)$ is equal to

$$\mathrm{div}_w p(x) = \int_\Omega (p(x,y) - p(y,x))\sqrt{w(x,y)}\,dy.$$

Hence non-local Laplacian $\Delta_w u(x)$ is given by

$$\begin{aligned}
(\Delta_w u)(x) = (\mathrm{div}_w(\nabla_w u))(x) &= \int_\Omega (\nabla_w u(x)(y) - \nabla_w u(y)(x))\sqrt{w(x,y)}\,dy \\
&= 2\int_\Omega (u(y) - u(x))w(x,y)\,dy \\
&= 2\int_\Omega u(y)w(x,y)\,dy - 2u(x)\int_\Omega w(x,y)\,dy.
\end{aligned}$$

By non-local derivatives, we define NLH^1 and $NLTV$ regularizers as follows.

- NLH^1 regularizer:

$$J(u) = \frac{1}{2}\int_\Omega |\nabla_w u(x)|^2\,dx,$$

- $NLTV$ regularizer:

$$J(u) = \int_\Omega |\nabla_w u(x)|\, dx.$$

Now we consider NLH^1 denoising problem

$$\underset{u}{\operatorname{argmin}} \left(\mu \frac{1}{2} \int_\Omega |\nabla_w u(x)|^2\, dx + \frac{1}{2} \int_\Omega (u(x) - f(x))^2\, dx \right).$$

Optimal condition for u is given by

$$u = f + \mu \Delta_w u.$$

If we use the formula for $\Delta_w u$, we have

$$u(x) = f(x) + 2\mu \left(\int_\Omega u(y) w(x,y)\, dy - u(x) \int_\Omega w(x,y)\, dy \right).$$

If we rearrange above equation, we have

$$u(x) = \frac{f(x) + 2\mu \int_\Omega u(y) w(x,y)\, dy}{1 + 2\mu \int_\Omega w(x,y)\, dy}.$$

A single iteration of fixed point method with $u_0 = f$ gives approximate solution to NLH^1 denoising problem

$$u_1(x) = \frac{f(x) + 2\mu \int_\Omega f(y) w(x,y)\, dy}{1 + 2\mu \int_\Omega w(x,y)\, dy}.$$

Note that we get the same formula as NLM filter when $\mu \to \infty$.

Denoising result of NLH^1 regularizer is presented in Figure 2.6.

2.3.3 Convex Optimization Theory

We now introduce the results from convex optimization theory which are essential for developing algorithms for our problems. This section is based on [10] and [27]. If we use regularization to our modeling, our objective functional $E(u)$ is written in the form of

$$E(u) = \mu J(u) + \frac{1}{2} \int_\Omega (s(x) - Lu(x))^2\, dx.$$

Let $\mathcal{X}$ be a Hilbert space with the inner product $\langle \cdot, \cdot \rangle$. For a extended real valued function $\varphi : \mathcal{X} \to [-\infty, +\infty]$, it is called *proper* if φ is not identically $+\infty$, and it is called *lower semi-continuous* if

$$\lim_{x_n \to x} \varphi(x_n) \geq \varphi(x),$$

for all sequences x_n which converges to $x \in \mathcal{X}$, and it is called *convex* if

$$\varphi(\lambda x + (1-\lambda)y) \leq \lambda\varphi(x) + (1-\lambda)\varphi(y),$$

for all $\lambda \in [0,1]$ and for all $x, y \in \mathcal{X}$. In particular if we have

$$\varphi(\lambda x + (1-\lambda)y) < \lambda\varphi(x) + (1-\lambda)\varphi(y),$$

for all $\lambda \in (0,1)$ and $x, y \in \mathcal{X}, x \neq y$, we say φ is *strictly convex.* For an extended real valued function $\varphi : \mathcal{X} \to [-\infty, +\infty]$, we define its domain $\operatorname{dom}\varphi$ by

$$\operatorname{dom}\varphi = \{x \in \mathcal{X} : -\infty < \varphi(x) < +\infty\},$$

and its epigraph $\operatorname{epi}\varphi$ by

$$\operatorname{epi}\varphi = \{(x, \eta) \in \mathcal{X} \times \mathbb{R} : \varphi(x) \leq \eta\}.$$

We define $\Gamma_0(\mathcal{X})$ as the class of all lower semi-continuous convex functions from $\mathcal{X}$ to $(-\infty, +\infty]$. If we consider discrete version, our functional space becomes $\mathcal{X} = \mathbb{R}^{N^2}$ and our problem can be generalized to:

Problem 2.3.3. *Find $\hat{u} \in \mathcal{X}$ such that*

$$\hat{u} = \operatorname*{argmin}_{u \in \mathcal{X}} \left(f_1(u) + f_2(u) \right),$$

for some two proper, lower semi-continuous, convex functions f_1 and f_2. In addition, we assume f_2 is differentiable on $\mathcal{X}$ with a $1/\beta$-Lipschitz continuous gradient for some $\beta \in (0, \infty)$.

To discuss the problem above, we need additional concepts on convex analysis. We first introduce convex conjugate or *Fenchel-Legendre transform* of $\varphi \in \Gamma_0(\mathcal{X})$.

Definition 2.3.4 (Fenchel-Legendre transform)**.** *Let $\varphi \in \Gamma_0(\mathcal{X})$. The conjugate φ^* of φ is defined by*

$$\varphi^*(u) := \sup_{x \in \mathcal{X}} \langle x, u \rangle - \varphi(x).$$

Note that $\varphi^* \in \Gamma_0(\mathcal{X})$ automatically since it is defined by the supremum of affine functions. Moreover it can be shown that $\varphi^{**} = \varphi$ using basic theory of convex analysis as in Zalinescu [27]. The subdifferential operator $\partial\varphi : \mathcal{X} \to 2^{\mathcal{X}}$ is the set-valued operator defined by

$$\partial\varphi(x) = \{u \in \mathcal{X} : \langle y - x, u \rangle + \varphi(x) \leq \varphi(y) \text{ for all } y \in \mathcal{X}\}, \tag{2.3.8}$$

or equivalently,

$$\partial\varphi(x) = \{u \in \mathcal{X} : \varphi(x) + \varphi^*(u) = \langle x, u \rangle\}.$$

According to Fermat's rule,

$$\varphi(x) = \inf \varphi(X) \Leftrightarrow 0 \in \partial\varphi(x).$$

Moreover, if φ is (Gâteaux) differentiable at x with gradient $\nabla\varphi(x)$, we have $\partial\varphi(x) = \{\nabla\varphi(x)\}$. Convex conjugates and Subdifferential operators have the following properties (Zalinescu [27]).

Lemma 2.3.5. *Let $\varphi \in \Gamma_0(\mathcal{Y})$, $\psi \in \Gamma_0(\mathcal{X})$, and let $L : \mathcal{X} \to \mathcal{Y}$ be a bounded linear operator such that $0 \in \operatorname{int}(\operatorname{dom}\varphi - L(\operatorname{dom}\psi))$. Then*

1. $\partial(\varphi \circ L + \psi) = L^* \circ (\partial\varphi) \circ L + \partial\psi$,
2. $\inf_{x\in\mathcal{X}}(\varphi(Lx) + \psi(x)) = \sup_{v\in\mathcal{Y}} -(\varphi^*(v) + \psi^*(-L^*v))$ *(Fenchel-Rockafellar duality).*

Now we define the notion of firmly nonexpansive operators.

Definition 2.3.6. *An operator $T : \mathcal{X} \to \mathcal{X}$ is firmly nonexpansive if it satisfies*

$$\|Tx - Ty\|^2 \leq \langle Tx - Ty, x - y \rangle,$$

for all $x, y \in \mathcal{X}$.

The definition above has the following equivalent statement:

$$\|Tx - Ty\|^2 + \|(\operatorname{Id} - T)x - (\operatorname{Id} - T)y\|^2 \leq \|x - y\|^2,$$

for all $x, y \in \mathcal{X}$, which can be easily seen by expanding the second term in the left hand side. Note that firmly nonexpansive operators are nonexpansive, i.e.,

$$\|Tx - Ty\| \leq \|x - y\|,$$

for all $x, y \in \mathcal{X}$. Another importance notion of convex analysis is proximal operators. The proximal operator $\operatorname{Prox}_\varphi$ of an operator $\varphi \in \Gamma_0(\mathcal{X})$ is defined by

$$\operatorname{Prox}_\varphi(x) = \operatorname*{argmin}_{y \in \mathcal{X}} \left(\varphi(y) + \frac{1}{2}\|y - x\|^2 \right),$$

which is the unique point satisfying

$$x - \operatorname{Prox}_\varphi(x) \in \partial\varphi(\operatorname{Prox}_\varphi(x)). \tag{2.3.9}$$

Our first observation is that proximal operators are firmly nonexpansive.

Lemma 2.3.7. *Let $\varphi \in \Gamma_0(\mathcal{X})$. Then $\operatorname{Prox}_\varphi$ and $\operatorname{Id} - \operatorname{Prox}_\varphi$ are firmly nonexpansive.*

Proof. Let $x, y \in \mathcal{X}$. If we use (2.3.8) and (2.3.9), we have

$$\langle \operatorname{Prox}_\varphi y - \operatorname{Prox}_\varphi x, x - \operatorname{Prox}_\varphi x \rangle + \varphi(\operatorname{Prox}_\varphi x) \leq \varphi(\operatorname{Prox}_\varphi y),$$

and symmetrically,

$$\langle \operatorname{Prox}_\varphi x - \operatorname{Prox}_\varphi y, y - \operatorname{Prox}_\varphi y \rangle + \varphi(\operatorname{Prox}_\varphi y) \leq \varphi(\operatorname{Prox}_\varphi x).$$

If we add above two inequalities, we have

$$\|\operatorname{Prox}_\varphi x - \operatorname{Prox}_\varphi y\|^2 \leq \langle \operatorname{Prox}_\varphi x - \operatorname{Prox}_\varphi y, x - y \rangle,$$

which is the desired result. $\square$

The Moreau envelope ${}^\gamma\varphi$ of index $\gamma \in (0, +\infty)$ of a function $\varphi \in \Gamma_0(\mathcal{X})$ is the continuous convex function defined by

$${}^\gamma\varphi(x) = \inf_{y \in \mathcal{X}} \varphi(y) + \frac{1}{2\gamma}\|y - x\|^2.$$

If we use proximal operators, it can be rewritten as

$$^{\gamma}\varphi(x) = \varphi(\operatorname{Prox}_{\gamma\varphi}(x)) + \frac{1}{2\gamma}\|\operatorname{Prox}_{\gamma\varphi}(x) - x\|^2.$$

If we apply Fenchel-Rockafellar duality The following lemma is a key point of the proof of Baillon-Haddad Theorem (Corollaire 10 of Baillon-Haddad [2]).

Lemma 2.3.8. *Let $\varphi \in \Gamma_0(\mathcal{X})$ and $\gamma \in (0, +\infty)$. Then $^{\gamma}\varphi$ is Fréchet differentiable on $\mathcal{X}$ and $\nabla {}^{\gamma}\varphi = (\operatorname{Id} - \operatorname{Prox}_{\gamma\varphi})/\gamma$.*

Proof. Since $^{\gamma}\varphi = {}^{1}(\gamma\varphi)/\gamma$, it suffices to show

$$\nabla {}^{1}\varphi = \operatorname{Id} - \operatorname{Prox}_{\varphi}. \tag{2.3.10}$$

From the notion of Fréchet differentiability, (2.3.10) means that

$$\lim_{\|\varepsilon\|\to 0} \frac{|{}^{1}\varphi(x+\varepsilon) - {}^{1}\varphi(x) - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle|}{\|\varepsilon\|} = 0.$$

By definition of Moreau envelope and proximal operator, we have

$$\begin{aligned}
{}^{1}\varphi(x+\varepsilon) &- {}^{1}\varphi(x) - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle \\
&= \varphi(\operatorname{Prox}_{\varphi}(x+\varepsilon)) + \frac{1}{2}\|x+\varepsilon - \operatorname{Prox}_{\varphi}(x+\varepsilon)\|^2 \\
&\quad - \varphi(\operatorname{Prox}_{\varphi}(x)) - \frac{1}{2}\|x - \operatorname{Prox}_{\varphi}(x)\|^2 - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle \\
&= \varphi(\operatorname{Prox}_{\varphi}(x+\varepsilon)) - \varphi(\operatorname{Prox}_{\varphi}(x)) - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle \\
&\quad + \langle \varepsilon - \operatorname{Prox}_{\varphi}(x+\varepsilon) + \operatorname{Prox}_{\varphi}(x), x - \operatorname{Prox}_{\varphi}(x)\rangle \\
&\quad + \frac{1}{2}\|\varepsilon - \operatorname{Prox}_{\varphi}(x+\varepsilon) + \operatorname{Prox}_{\varphi}(x)\|^2 \\
&= \varphi(\operatorname{Prox}_{\varphi}(x+\varepsilon)) - \varphi(\operatorname{Prox}_{\varphi}(x)) \\
&\quad - \langle \operatorname{Prox}_{\varphi}(x+\varepsilon) - \operatorname{Prox}_{\varphi}(x), x - \operatorname{Prox}_{\varphi}(x)\rangle \\
&\quad + \frac{1}{2}\|(\operatorname{Id} - \operatorname{Prox}_{\varphi})(x+\varepsilon) - (\operatorname{Id} - \operatorname{Prox}_{\varphi})(x)\|^2.
\end{aligned}$$

If we use $x - \operatorname{Prox}_{\varphi}(x) \in \partial\varphi(\operatorname{Prox}_{\varphi}(x))$, we have

$${}^{1}\varphi(x+\varepsilon) - {}^{1}\varphi(x) - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle \geq 0.$$

On the other hand, since $x + \varepsilon - \operatorname{Prox}_{\varphi}(x+\varepsilon) \in \partial\varphi(\operatorname{Prox}_{\varphi}(x+\varepsilon))$, we have

$${}^{1}\varphi(x+\varepsilon) - {}^{1}\varphi(x) - \langle x - \operatorname{Prox}_{\varphi}(x), \varepsilon\rangle$$

$$\begin{aligned}
&= \varphi(\operatorname{Prox}_\varphi(x+\varepsilon)) - \varphi(\operatorname{Prox}_\varphi(x)) \\
&\quad + \langle \operatorname{Prox}_\varphi(x) - \operatorname{Prox}_\varphi(x+\varepsilon), x+\varepsilon - \operatorname{Prox}_\varphi(x+\varepsilon)\rangle \\
&\quad + \langle \operatorname{Prox}_\varphi(x+\varepsilon) - \operatorname{Prox}_\varphi(x), \varepsilon + \operatorname{Prox}_\varphi(x) - \operatorname{Prox}_\varphi(x+\varepsilon)\rangle \\
&\quad + \frac{1}{2}\|(\operatorname{Id} - \operatorname{Prox}_\varphi)(x+\varepsilon) - (\operatorname{Id} - \operatorname{Prox}_\varphi)(x)\|^2 \\
&\leq \frac{1}{2}\|\varepsilon\|^2 - \frac{1}{2}\|\operatorname{Prox}_\varphi(x+\varepsilon) - \operatorname{Prox}_\varphi(x)\|^2 \\
&\leq \frac{1}{2}\|\varepsilon\|^2.
\end{aligned}$$

Therefore, we have

$$\nabla^1\varphi = \operatorname{Id} - \operatorname{Prox}_\varphi .$$

□

Now we state Baillon-Haddad Theorem.

Theorem 2.3.9 (Baillon-Haddad, Corollaire 10 of Baillon-Haddad [2]). *Let $\varphi : \mathcal{X} \to \mathbb{R}$ be convex, Fréchet differentiable on $\mathcal{X}$, and such that $\nabla\varphi$ is β-Lipschitz continuous for some $\beta \in (0, +\infty)$. Then $\nabla\varphi/\beta$ is firmly nonexpansive.*

In the sequel, we follow the proof in [4].

Proof. Let $q(x) = \frac{1}{2}\|x\|^2$. Let $g = \beta q - \varphi$. Then by Cauchy-Schwartz,

$$\langle x-y, \nabla g(x) - \nabla g(y)\rangle = \beta\|x-y\|^2 - \langle x-y, \nabla\varphi(x) - \nabla\varphi(y)\rangle \geq 0,$$

and hence ∇g is monotone. Since ∇g is monotone, it follows that g is convex (Theorem 2.1.11 of Zalinescu [27]). Since $g \in \Gamma_0(\mathcal{X})$, it holds that $g = g^{**}$. Then, we have

$$\varphi = \beta q - g = \beta q - g^{**} = \beta q - \sup_{x\in\mathcal{X}}(\langle\cdot, x\rangle - g^*(x)) = \inf_{x\in\mathcal{X}}(\beta q - \langle\cdot, x\rangle + g^*(x)).$$

Therefore,

$$\begin{aligned}
\varphi^*(y) &= \sup_{z\in\mathcal{X}}\langle z, y\rangle - \varphi(z) = \sup_{z\in\mathcal{X}}\langle z, y\rangle - \inf_{x\in\mathcal{X}}(\beta q(z) - \langle z, x\rangle + g^*(x)) \\
&= \sup_{x,z\in\mathcal{X}} \langle z, x+y\rangle - \beta q(z) - g^*(x)
\end{aligned}$$

$$\begin{aligned} &= \sup_{x\in\mathcal{X}} (\beta q)^*(x+y) - g^*(x) = \sup_{x\in\mathcal{X}} q(x+y)/\beta - g^*(x) \\ &= q(y)/\beta + \sup_{x\in\mathcal{X}} (\langle x, y\rangle - (q/\beta - g^*)(x)). \end{aligned}$$

Since the second term in the last equation is the supremum of affine functions, $\varphi^* - q/\beta$ is convex. Now let $h = \varphi^* - q/\beta$. Then $h \in \Gamma_0(\mathcal{X})$ and $\varphi = \varphi^{**} = (h + q/\beta)^*$, and

$$\varphi(x) = \sup_{y\in\mathcal{X}} \langle x, y\rangle - h(y) - q(y)/\beta = \beta q(x) - \inf_{y\in\mathcal{X}} h(y) + q(y - \beta x)/\beta,$$

that is,

$$\varphi = \beta q - {}^{\beta}h(\beta\,\mathrm{Id}).$$

Therefore,

$$\nabla\varphi = \beta\,\mathrm{Id} - \beta(\mathrm{Id} - \mathrm{Prox}_{\beta h}) = \beta\,\mathrm{Prox}_{\beta h}\,.$$

Finally, $\nabla\varphi/\beta$ is firmly nonexpansive. □

In this setting, we have the following results on solutions of Problem 2.3.3.

Proposition 2.3.10. *If $\mathcal{X}$ is a real Hilbert space, then following statements hold.*

1. *Existence: Problem 2.3.3 has at least one solution if $f_1 + f_2$ is coercive, i.e.,*

$$\lim_{\|x\|\to+\infty} f_1(x) + f_2(x) = +\infty.$$

2. *Uniqueness: Problem 2.3.3 has at most one solution if $f_1 + f_2$ is strictly convex.*

3. *Characterization: If u is a solution of Problem 2.3.3, then it holds that:*

$$u = \mathrm{Prox}_{\gamma f_1}(u - \gamma\nabla f_2(u)),$$

for any $\gamma \in (0, +\infty)$.

Proof. 1. Let $(x_n)_{n\in\mathbb{N}} \subset \mathcal{X}$ be a minimizing sequence of $f_1 + f_2$. Then coercivity of $f_1 + f_2$ implies boundedness of $(x_n)_{n\in\mathbb{N}}$. By Banach-Alaoglu theorem,

$(x_n)_{n\in\mathbb{N}}$ has a subsequence weak-* converging to some $x \in \mathcal{X}$. Finally, lower semicontinuity of f_1, f_2 implies

$$\lim_{n\to+\infty} f_1(x_n) + f_2(x_n) \geq f_1(x) + f_2(x),$$

or equivalently, x is a minimizer of $f_1 + f_2$.

2. Let $x, y \in \mathcal{X}$ be minimizers of $f_1 + f_2$. If $x \neq y$, then

$$(f_1 + f_2)\left(\frac{x+y}{2}\right) < \frac{1}{2}((f_1 + f_2)(x) + (f_1 + f_2)(y)),$$

which is contradiction. Therefore $x = y$.

3. Let u be a solution. Then by Fermat's rule,

$$0 \in \partial f_1(u) + \nabla f_2(u).$$

It is equivalent to

$$0 \in \partial f_1(u) + 1/\gamma u - (1/\gamma u - \nabla f_2(u)).$$

Therefore

$$(u - \gamma\nabla f_2(u)) - u \in \partial(\gamma f_1(u)),$$

which means $u = \mathrm{Prox}_{\gamma f_1}(u - \gamma\nabla f_2(u))$. □

In Chapter 3, we introduce Forward-Backward Splitting Algorithm, which solves

$$u = \mathrm{Prox}_{\gamma f_1}(u - \gamma\nabla f_2(u))$$

by a convergent fixed point iteration method. In that chapter, we need following notions from Combettes [8].

Definition 2.3.11. *Let S be a nonempty closed and convex set in a real Hilbert space $\mathcal{X}$ with norm $\|\cdot\|$. A sequence $(x_n)_{n\in\mathbb{N}}$ of points in $\mathcal{X}$ is said to be S-Fejérian (or Fejér monotone with respect to S) if*

$$(\forall x \in S)(\forall n \in \mathbb{N}) \quad \|x_{n+1} - x\| \leq \|x_n - x\|.$$

Let $\mathfrak{W}$ and $\mathfrak{S}$ be the sets of weak and strong cluster points of $(x_n)_{n\in\mathbb{N}}$ respectively. Then Fejérian sequences have the following property.

Lemma 2.3.12. *Let $(x_n)_{n\in\mathbb{N}}$ be S-Fejérian. Then, the following assertions hold.*

1. $(x_n)_{n\in\mathbb{N}}$ *is bounded.*
2. $(\forall x \in S)$ $(\|x_n - x\|)_{n\in\mathbb{N}}$ *converges.*
3. $(d(x_n, S))_{n\in\mathbb{N}}$ *is nonincreasing.*
4. $(\forall x \in S) \quad x_n \to x \Leftrightarrow \underline{\lim}\|x_n - x\| = 0 \Leftrightarrow S \cap \mathfrak{S} \neq \emptyset$

Lemma 2.3.13 (Weak Convergence)**.** *If $(x_n)_{n\in\mathbb{N}}$ is S-Fejérian, it converges weakly to a point in S if and only if $\mathfrak{W} \subset S$.*

Lemma 2.3.14 (Strong Convergence)**.** *If $(x_n)_{n\in\mathbb{N}}$ is S-Fejérian, it converges strongly to a point in S if and only if $\underline{\lim} d(x_n, S) = 0$.*

At the end of this section, we introduce the notion of Bregman distance, which is essential to the theory of Bregman iterative algorithms.

Definition 2.3.15. *Let $J : \mathcal{X} \to \mathbb{R}$ be a convex functional on a Hilbert space $\mathcal{X}$. If $x, y \in \mathcal{X}$ and $p \in \partial J(y)$, we define Bregman distance $D_J^p(x, y)$ by*

$$D_J^p(x, y) = J(x) - J(y) - \langle p, x - y\rangle.$$

Note that this is not a mathematical metric because generally $D_J^p(x, y) \neq D_J^p(y, x)$. Nevertheless, it still measures the distance between x and y in the sense that $D_J^p(x, y) \geq 0$ and $D_J^p(x, y) \geq D_J^p(z, y)$ for all $z \in [x, y]$.

Chapter 3

Proposed Methods

In this chapter, we first propose an optimized way to compute s.v.c. operations with minimal resources. Then we discuss Forward Backward Splitting (FBS) algorithm. We also introduce Split Bregman technique to compute the denoising part of FBS algorithm for some l_1-type regularizations. In the last section, we propose various image recovery algorithms based on variety of regularization methods.

Recall our recovery model (2.3.7),

$$u = \operatorname*{argmin}_{u \in X} \mu J(u) + \frac{1}{2} \int_{\Omega} \left(Lu(x) - s(x) \right)^2 \, dx.$$

If J is convex, optimal condition of (2.3.7) is given by

$$0 \in \mu \partial J(u) + L^t L u - L^t s. \tag{3.0.1}$$

Let us consider the simpliest case such that $J(u) = \frac{1}{2}|\nabla u|^2$. Then (3.0.1) becomes

$$0 = -\mu \Delta u + L^t L u - L^t s,$$

that is,

$$u = (L^t L - \mu \Delta)^{-1} s. \tag{3.0.2}$$

In the case of spatially invariant convolution operators, the above can be directly solved using discrete Fourier transform (DFT):

$$u = \mathcal{F}^{-1} \left(\frac{\mathcal{F}(s)}{|\mathcal{F}(k)|^2 - \mu \mathcal{F}(d)} \right),$$

where k is the (spatially invariant) convolution kernel of L and d is the convolution kernel of Laplacian operator

$$d = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

However, for the case of s.v.c. kernels, (3.0.2) becomes difficult to solve. If we try to solve (3.0.2) with the matrix representation of $L^tL - \mu\Delta$ using the standard basis $\{E_{i,j} : (i,j) \in \Omega\}$, computation of the inverse and matrix-vector multiplication requires high computational cost with respect to time and memory usage. Hence we present low cost implementation of s.v.c. problems and restoration algorithms in this chapter.

3.1 Low Cost Implementation Using Small Support Assumption

In this section, we discuss low cost implementation of s.v.c. using small support assumption on PSFs. When we consider monochromatic aberrations occurring in modern digital cameras, we may assume the following property.

Definition 3.1.1. *We say the convolution kernel $k(x,y)$ satisfies small support assumption, if there exists $w > 0$ such that*

$$\operatorname{supp} k(x,\cdot) = \overline{\{y \in \Omega : k(x,y) \neq 0\}} \subset (x + B) \cap \Omega,$$

where $B = [-w,w]^2$.

We may assume PSFs also satisfy the small support assumption as in Figure 2.5,

$$\mathrm{PSF}(x,\cdot) \subset (x + B) \cap \Omega.$$

The small support assumption reduces memory usage to store discrete version of PSFs information. We use 4-dimensional array A of size $512 \times 512 \times (2w + 1) \times (2w + 1)$ to store PSFs as follows:

$$A(x,y) = \mathrm{PSF}(x, x + y),$$

where $x \in \Omega_N, y \in B$. If we use A, $L^t u$ can be rewritten as

$$L^t u(x) = \int_B \mathrm{PSF}(x, x+y) u(x+y)\, dy = \int_B A(x,y) u(x+y)\, dy.$$

Recall that $k_{L^t L}(x,z) = \int_\Omega k(y,x)k(y,z)\, dy$ and so

$$k_{L^t L}(x,z) = \int_\Omega \mathrm{PSF}(x,y)\,\mathrm{PSF}(z,y)\, dy = \int_{(B+x)\cap(B+z)} \mathrm{PSF}(x,y)\,\mathrm{PSF}(z,y)\, dy.$$

Since $\mathrm{PSF}(x,y) = A(x, y-x)$, we get

$$k_{L^t L}(x,z) = \int_{B\cap(B-(x-z))} A(x,y)A(z, y+x-z)\, dy. \tag{3.1.3}$$

Therefore,

$$L^t L u(x) = \int_\Omega \int_{B\cap(B-(x-z))} A(x,y)A(z,y+x-z)\, dy\, u(z)\, dz.$$

Note that $B \cap (B-(x-z))$ is not empty if and only if $|x-z|_\infty := \max_n |x-z|_n \le 2w$. It also means that $z \in x + 2B$, hence

$$L^t L u(x) = \int_{x+2B} \int_{B\cap(B-(x-z))} A(x,y)A(z,y+x-z)\, dy\, u(z)\, dz.$$

If we use change of variables,

$$L^t L u(x) = \int_{2B} \int_{B\cap(B+z)} A(x,y)A(x+z, y-z)\, dy\, u(x+z)\, dz. \tag{3.1.4}$$

Hence computational cost of $L^t L u$ is $\mathcal{O}((2w+1)^4|\Omega|)$. This cost is quite small compare to $\mathcal{O}(|\Omega|^3)$, which is the cost of non-optimized direct computation of (2.3.2). In the implementation, we pre-compute $D(x,z)$:

$$D(x,z) = \int_{B\cap(B+z)} A(x,y)A(x+z,y-z)\, dy.$$

$D(x,z)$ is 4-dimensional array of size $(4w+1)^2|\Omega|$. If we utilize pre-computed $D(x,z)$, computational cost becomes $\mathcal{O}((4w+1)^2|\Omega|)$, while the cost of non-optimized computation is $\mathcal{O}(|\Omega|^2)$. If we use this optimized computation, we can easily solve iterative solving part of (3.2.5). Also note that the computation can be easily parallelized.

Further optimization can be done by defining $C(x,y) = A(x+y,-y)$. Then it holds that

$$C(x,y) = \mathrm{PSF}(x+y,y) = k(x,x+y).$$

If we use arrays A and C, we can compute L^tLu by two-step convolution algorithm:

$$\begin{aligned} L^tLu(x) &= \int_B A(x,y)Lu(x+y)\,dy \\ &= \int_B A(x,y)\int_B C(x+y,z)u(x+y+z)\,dz. \end{aligned}$$

Computational cost for two-step convolution algorithm is $\mathcal{O}(2(2w+1)^2|\Omega|)$, which is smaller than the cost of using the array D.

3.1.1 Vectorization Techniques

In this section, we discuss vectorization techniques for further acceleration of s.v.c. implementation. Algorithm running time can be drastically reduced if we use the following SIMD (Single Instruction, Multiple Data) optimized vectorization techniques. Suppose we want to compute

$$s(x) = \int_\Omega k(x,y)u(y)\,dy.$$

Traditional way to do this is:

Algorithm 1 Non-vectorized Convolution

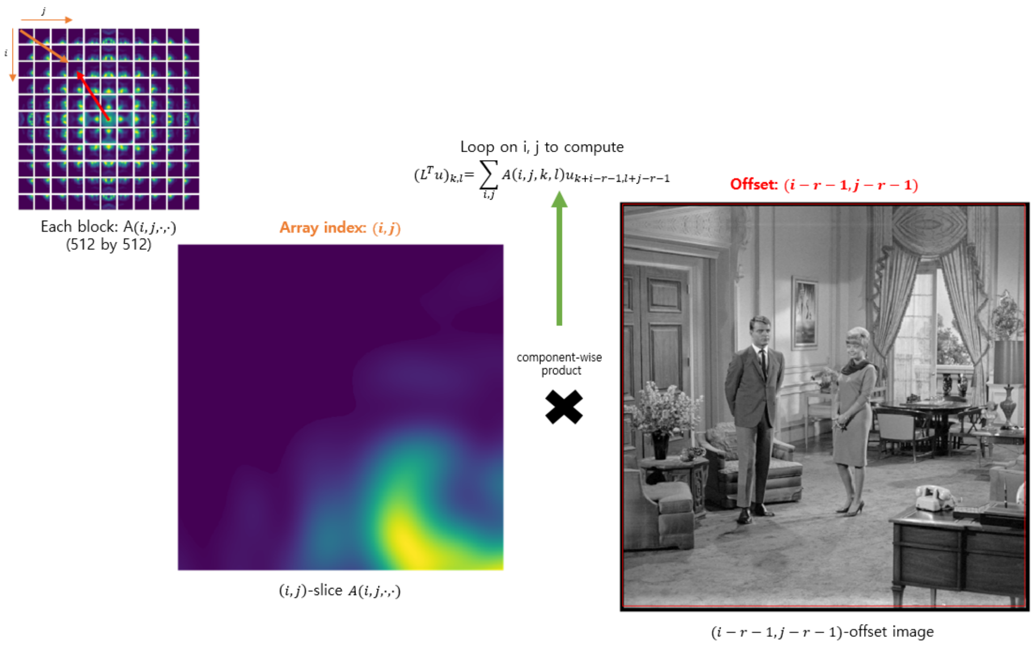
```
Initialize: $s = 0$
for $x$ in $\Omega$ do
    for $y$ in $B$ do
        $s(x) \leftarrow s(x) + k(x,x+y)u(x+y)$
    end for
end for
Output: $s$
```

Instead, we use vectorized version:

Algorithm 2 Vectorized Convolution

```
Initialize: $s = 0$
for $y$ in $B$ do
    $s(\cdot) \leftarrow s(\cdot) + k(\cdot, \cdot + y)u(\cdot + y)$
end for
Output: $s$
```

Implementations can be done with SIMD (single instruction multiple data) instructions or GPGPU programming. Non-local denoisers can be boosted with this technique as well. Our vectorization technique is illustrated in Figure 3.1.



j
i
Each block: A$(i,j,\cdot,\cdot)$ (512 by 512)
Array index: (i,j)
(i,j)-slice $A(i,j,\cdot,\cdot)$
Loop on i, j to compute
$(L^T u)_{k,l} = \sum_{i,j} A(i,j,k,l) u_{k+i-r-1, l+j-r-1}$
component-wise product
Offset: $(i-r-1, j-r-1)$
$(i-r-1, j-r-1)$-offset image

Figure 3.1: Illustration of vectorization technique. 4-dimensional array A stores PSF values. Number of loop variables reduces to the size of the support box B.

3.2 Proposed Algorithm

In this section, we introduce optimized numerical algorithms to solve (2.3.7).

3.2.1 Forward Backward Splitting Algorithm

This section covers Forward Backward Splitting Algorithm. Suppose that we have the following optimization problem:

$$\hat{u} = \operatorname*{argmin}_u \mu J(u) + H(u),$$

where J is convex and H is differentiable. As shown in Section 2.3.3, optimal condition for a solution u is given by

$$u = \mathrm{Prox}_{\gamma f_1}(u - \gamma \nabla f_2(u)),$$

for any $\gamma \in (0, +\infty)$, where $f_1 = \mu J$, $f_2 = H$. Therefore, finding a solution to Problem 2.3.3 is equivalent to finding a fixed point of $\mathrm{Prox}_{\gamma f_1}(\cdot - \gamma \nabla f_2(\cdot))$. It is natural to consider fixed point iteration,

$$u^{k+1} = \mathrm{Prox}_{\gamma f_1}(u^k - \gamma \nabla f_2(u^k)),$$

or equivalently,

$$\begin{cases} v^{k+1} = & u^k - \gamma \nabla f_2(u^k), \\ u^{k+1} = & \mathrm{Prox}_{\gamma f_1}(v^{k+1}). \end{cases}$$

This famous method is called Forward-Backward Operator Splitting (FBS), since it iteratively computes forward and backward differences of f_2 and f_1. It is proven in Combettes-Wajs [10] that above iteration converges when $0 < \gamma < \frac{2}{L}$, where $L > 0$ is Lipschitz constant of ∇f_2. To prove this, let G be the set of the solutions of Problem 2.3.3. The following theorem is a special case of Theorem 3.1 in Combettes [9].

Theorem 3.2.1 (Forward-Backward Splitting). *Suppose that $G \neq \emptyset$. Let $(\gamma_n)_{n\in\mathbb{N}}$ be a sequence in $(0, \infty)$ such that $0 < \inf_{n\in\mathbb{N}} \gamma_n \leq \sup_{n\in\mathbb{N}} \gamma_n < 2/L$, and let $(\lambda_n)_{n\in\mathbb{N}}$ be a sequence in $(0, 1]$ such that $\inf_{n\in\mathbb{N}} \lambda_n > 0$. Fix $u^0 \in \mathcal{X}$ and, for every $n \in \mathbb{N}$, set*

$$u^{n+1} = (1 - \lambda_n)u^n + \lambda_n(\mathrm{Prox}_{\gamma_n f_1}(u^n - \gamma_n \nabla f_2(u^n))).$$

Then, the following assertions hold.

1. *$(u^n)_{n\in\mathbb{N}}$ converges weakly to a point $u \in G$.*

2. $\sum_{n\in\mathbb{N}} \|\nabla f_2(u^n) - \nabla f_2(u)\|^2 < +\infty$.
3. $\sum_{n\in\mathbb{N}} \|\operatorname{Prox}_{\gamma_n f_1}(u^n - \gamma_n \nabla f_2(u^n)) - u^n\|^2 < +\infty$.
4. $(u^n)_{n\in\mathbb{N}}$ *converges strongly to* u *if and only if* $\underline{\lim} d(u^n, G) = 0$, *where* $d(x, G) = \inf_{y\in G} \|x - y\|$ *is the distance function from* G.

Proof. We first show condition 2 and condition 3. Note that $T_{1,n} = \operatorname{Prox}_{\gamma_n f_1}$ and $T_{2,n} = \operatorname{Id} - \gamma_n \nabla f_2$ are firmly nonexpansive and $u \in G$ means $T_{1,n}T_{2,n}u = u$. Then,

$$\begin{aligned}\|u^{n+1} - u\| &= \|(1-\lambda_n)u^n + \lambda_n T_{1,n}T_{2,n}u^n - u\|\\ &\le (1-\lambda_n)\|u^n - u\| + \lambda_n \|T_{1,n}T_{2,n}(u^n - u)\|\\ &\le (1-\lambda_n)\|u^n - u\| + \lambda_n\|u^n - u\| = \|u^n - u\|,\end{aligned}$$

for all $u \in G$, in other words, u^n is G-Fejérian. Now we show weak convergence. If we set $y^n = T_{1,n}T_{2,n}u^n$, it holds that

$$\begin{aligned}\|u^{n+1} - u\|^2 &= \|(1-\lambda_n)(u^n - u) + \lambda_n y^n - u\|^2\\ &= (1-\lambda_n)\|u^n - u\|^2 + \lambda_n\|y^n - u\|^2\\ &\quad - \lambda_n(1-\lambda_n)\|u^n - y^n\|^2.\end{aligned}$$

On the other hand, since $T_{1,n}, T_{2,n}$ are firmly nonexpansive,

$$\begin{aligned}\|y^n - u\|^2 &= \|T_{1,n}T_{2,n}u^n - T_{1,n}T_{2,n}u\|^2\\ &\le \|T_{2,n}u^n - T_{2,n}u\|^2 - \|(\operatorname{Id} - T_{1,n})T_{2,n}u^n - (\operatorname{Id} - T_{1,n})T_{2,n}u\|^2\\ &\le \|u^n - u\|^2 - \|(\operatorname{Id} - T_{1,n})T_{2,n}u^n - (\operatorname{Id} - T_{1,n})T_{2,n}u\|^2\\ &\quad - \|(\operatorname{Id} - T_{2,n})u^n - (\operatorname{Id} - T_{2,n})u\|^2.\end{aligned}$$

Therefore,

$$\begin{aligned}&\lambda_n\|(\operatorname{Id} - T_{1,n})T_{2,n}u^n - (\operatorname{Id} - T_{1,n})T_{2,n}u\|^2\\ &\quad + \lambda_n\|(\operatorname{Id} - T_{2,n})u^n - (\operatorname{Id} - T_{2,n})u\|^2\\ &\quad + \lambda_n(1-\lambda_n)\|u^n - y^n\|^2 \le \|u^n - u\|^2 - \|u^{n+1} - u\|^2.\end{aligned}$$

If we add both sides for $n = 0, \ldots, N-1$, we get

$$\sum_{n=0}^{N-1} \lambda_n \|(\operatorname{Id} - T_{1,n})T_{2,n}u^n - (\operatorname{Id} - T_{1,n})T_{2,n}u\|^2$$

$$+\sum_{n=0}^{N-1}\lambda_n\|(\mathrm{Id}-T_{2,n})u^n-(\mathrm{Id}-T_{2,n})u\|^2$$
$$+\sum_{n=0}^{N-1}\lambda_n(1-\lambda_n)\|u^n-y^n\|^2\le\|u^0-u\|^2-\|u^N-u\|^2.$$

Therefore if we let $N\to+\infty$, we have condition 2 and 3. In addition, $u^n-y^n\to 0$. We now prove condition 1 using condition 2 and 3. We first assume $y\in\mathfrak{W}$ and $u^{n_k}\rightharpoonup y$ for some $(n_k)_{k\in\mathbb{N}}$. Let $t^n=\frac{u^n-y^n}{\gamma_n}-\nabla f_2(u^n)$. Then by condition 2 and condition 3, $t_n\to-\nabla f_2(u)$. We want to show $y\in G$. Since $u^{n_k}\rightharpoonup y$ and $\nabla f_2(u^n)\to\nabla f_2(u)$, $\nabla f_2(y)=\nabla f_2(u)$. Thus $y^{n_k}\rightharpoonup y$, $t^{n_k}\to-\nabla f_2(y)$, and moreover $y^{n_k}=\mathrm{Prox}_{\gamma_{n_k}f_1}(\gamma_{n_k}t^{n_k})$, in other words $(y^{n_k},t^{n_k})\in\mathrm{gr}\,\partial f_1$. Since ∂f_1 is maximally1 monotone, its graph is sequentially weakly-strongly closed in $\mathcal{X}\times\mathcal{X}$. Hence $-\nabla f_2(y)\in\partial f_1(y)$, i.e. $y\in G$.

Condition 4 for Strong convergence is equivalent to Lemma 2.3.14. □

Liang et al. [18] proved sequential convergence rate is given by $\|u^k-u^{k+1}\|=o(1/\sqrt{k})$ and Molinari et al. [20] proved objective functional convergence rate is given by $E(u^k)-\min_u E(u)=o(1/k)$. Thus we have a convergence guaranteed algorithm:

Algorithm 3 Original FBS

```
**Initialize:** $k=0, u^0=0, 0<\delta<\frac{2}{L}$
**while** not converges **do**
    $u^{k+1}\leftarrow\mathrm{Prox}_{\delta\mu J}(u^k-\delta\nabla H(u^k))$
    $k\leftarrow k+1$
**end while**
**Output:** $u^k$
```

In case of image recovery model, we have $H(u)=\frac{1}{2}\|Lu-s\|^2$. Therefore if we substitute $\nabla H(u)=L^tLu-L^ts$ and define an auxiliary variable $v^{k+1}=u^k-\delta(L^tLu^k-L^ts)$, we obtain the following 2-step algorithm:

$$\begin{cases} v^{k+1} & =u^k-\delta(L^tLu^k-L^ts),\\ u^{k+1} & =\mathrm{argmin}_u\,\delta\mu J(u)+\frac{1}{2}\|u-v^{k+1}\|^2.\end{cases}\tag{3.2.5}$$

FBS algorithm splits optimization problem into two parts:

1. Iterative solving part (gradient descent) with step parameter $\delta > 0$.
2. Denoising part using the regularizer J.

Algorithm 4 FBS image recovery (2-step algorithm)

```
**Initialize:** $k = 0, u^0 = 0, v^0 = 0, \delta < \frac{1}{\|L^t L\|}$
**while** not converges **do**
    $\begin{cases} v^{k+1} & \leftarrow u^k - \delta(L^t L u^k - L^t s), \\ u^{k+1} & \leftarrow \operatorname{argmin}_u \delta\mu J(u) + \frac{1}{2}\|u - v^{k+1}\|^2. \end{cases}$
    $k \leftarrow k + 1$
**end while**
**Output:** $u^k$
```

3.2.2 Split Bregman Method

Now we introduce a method to solve denoising part of Algorithm 4. This section is based on Goldstein-Osher [14]. Suppose that we have a denoising problem

$$u = \underset{u}{\operatorname{argmin}} \left(\mu J(u) + \frac{1}{2}\|u - v\|^2 \right). \tag{3.2.6}$$

When J is given by l_1 type regularizer on $\Phi(u)$, in other words $J(u) = \int_\Omega |\Phi(u)(x)|\, dx$ for some linear operator Φ, directly solving (3.2.6) is nearly impossible. Among the many methods to solve (3.2.6), Split Bregman method suggested in Goldstein-Osher [14] is known to be one of the fastest way to do it. The basic idea is splitting the problem into two problems: a problem with respect to u and a problem with repect to $\Phi(u)$. For example, if we have Rudin-Osher-Fatemi denoising model [22], $\Phi(u) = \nabla u$. Our denoising problem is

$$u = \underset{u}{\operatorname{argmin}} \left(\mu \int_\Omega |\Phi(u)(x)|\, dx + \frac{1}{2}\|u - v\|^2 \right).$$

We use an auxiliary variable d and rewrite the equation as a constrained optimization problem,

$$(d, u) = \underset{d,u}{\operatorname{argmin}} \left[\mu \int_\Omega |d(x)|\, dx + \frac{1}{2}\|u - v\|^2 \right] \quad \text{subject to} \quad d = \Phi(u). \tag{3.2.7}$$

This is the key idea of Split Bregman splitting method. In order to solve the equation above, we need the following algorithms.

Bregman Iteration for Constrained l_1-Minimization Problems

Let $E : \mathbb{R}^n \to \mathbb{R}$ be a convex operator. We wish to solve

$$\operatorname*{argmin}_{u\in\mathbb{R}^n} E(u) \quad \text{subject to} \quad F(u) = 0,$$

for some convex functional $F : \mathbb{R}^d \to \mathbb{R}$. Bregman iteration is an iterative algorithm that solves the above problem using Bregman distance and unconstrained approximations,

$$\begin{aligned} &u^{k+1} \leftarrow \operatorname{argmin}_u D_E^{p^k}(u, u^k) + \lambda F(u), \\ &k \leftarrow k+1, \end{aligned} \tag{3.2.8}$$

where $p^k \in \partial E(u^k)$, $\lambda > 0$. When F is differentiable, we have

$$0 \in \partial E(u^{k+1}) - p^k + \lambda \nabla F(u^{k+1}).$$

Therefore, updating Bregman vector p^k is simply

$$p^{k+1} \leftarrow p^k - \nabla \lambda F(u^{k+1}).$$

Osher et al. [21] proved the following convergence results of the above algorithm:

Theorem 3.2.2. *Assume that E and F are convex functionals, and that F is differentiable. If solutions to* (3.2.8) *exist, then we have*

1. *monotone decrease in F: $F(u^{k+1}) \leq F(u^k)$.*
2. *convergence to a minimizer of F: $F(u^k) \leq F(u^*) + E(u^*)/k$.*

When the constraint is given by $Au = b$ for some linear operator $A : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$, we may define $F(u) = \frac{1}{2}\|Au - b\|^2$. In this case, F is differentiable and $\nabla F(u) = A^t(Au - b)$. Then (3.2.8) becomes

$$\begin{aligned} &u^0 \leftarrow 0, p^0 \leftarrow 0, \\ &u^{k+1} \leftarrow \operatorname{argmin}_u D_E^{p^k}(u, u^k) + \tfrac{\lambda}{2}\|Au - b\|^2, \\ &p^{k+1} \leftarrow p^k - \lambda A^t(Au^{k+1} - b), \\ &k \leftarrow k+1. \end{aligned} \tag{3.2.9}$$

We consider another version of (3.2.9):

$$\begin{aligned}
&b^0 \leftarrow 0, u^0 \leftarrow 0,\\
&b^{k+1} \leftarrow b + b^k - Au^k\\
&u^{k+1} \leftarrow \operatorname{argmin}_u E(u) + \tfrac{\lambda}{2}\|Au - b^{k+1}\|^2,\\
&k \leftarrow k+1.
\end{aligned} \tag{3.2.10}$$

Lemma 3.2.3. *The algorithms* (3.2.9) *and* (3.2.10) *are equivalent.*

Proof. We use mathematical induction on k. Let u^k and $\hat{u}^k$ denote the solutions to (3.2.9) and (3.2.10) respectively. For $k = 0$, the objective functionals for u^1 and $\hat{u}^1$ are equally (up to constant $E(0)$)

$$E(u) + \frac{\lambda}{2}\|Au - b\|^2.$$

Moreover, since $b^1 = b$, we have

$$p^1 = p^0 - \lambda A^t(Au^1 - b) = \lambda A^t(b - Au^1) = \lambda A^t(b - A\hat{u}^1) = \lambda A^t(b^1 - A\hat{u}^1).$$

We claim that $p^k = \lambda A^t(b^k - A\hat{u}^k)$ for all k. Assume $p^k = \lambda A^t(b^k - A\hat{u}^k)$ holds true for some k. Then, the objective function of (3.2.9) becomes

$$\begin{aligned}
D_E^{p^k}(u, u^k) + \frac{\lambda}{2}\|Au - b\|^2 &= E(u) - \langle p^k, u\rangle + \frac{\lambda}{2}\|Au - b\|^2 + C\\
&= E(u) - \lambda\langle b^k - A\hat{u}^k, Au\rangle + \frac{\lambda}{2}\|Au - b\|^2 + C\\
&= E(u) + \frac{\lambda}{2}\|Au - (b + (b^k - A\hat{u}^k))\|^2 + C\\
&= E(u) + \frac{\lambda}{2}\|Au - b^{k+1}\|^2 + C,
\end{aligned}$$

for some generic constant C. Therefore, the objective functions of u^{k+1} and $\hat{u}^{k+1}$ are equal. Consequently, u^{k+1} and $\hat{u}^{k+1}$ solve the same objective function and therefore,

$$\begin{aligned}
p^{k+1} &= p^k - \lambda A^t(Au^{k+1} - b) = \lambda A^t(b^k - A\hat{u}^k) - \lambda A^t(Au^{k+1} - b)\\
&= \lambda A^t(b + b^k - A\hat{u}^k - Au^{k+1}) = \lambda A^t(b^{k+1} - Au^{k+1})\\
&= \lambda A^t(b^{k+1} - A\hat{u}^{k+1}).
\end{aligned}$$

□

The following theorem is the conclusion of this section:

Theorem 3.2.4. *Let $E : \mathbb{R}^n \to \mathbb{R}$ is convex and $A : \mathbb{R}^n \to \mathbb{R}^m$ is linear. Suppose that some iteration, u^* of* (3.2.10)*, satisfies $Au^* = b$. Then u^* is a solution to the original constrained problem,*

$$u^* = \operatorname*{argmin}_u E(u) \qquad \textit{subject to} \qquad Au = b. \tag{3.2.11}$$

Proof. Let u^*, b^* be the iteration satisfying $Au^* = b$ and

$$u^* = \operatorname*{argmin}_u E(u) + \frac{\lambda}{2}\|Au - b^*\|^2.$$

Let $\hat{u}$ be a true solution to (3.2.11). Then $Au^* = b = A\hat{u}$. Therefore we have

$$\|Au^* - b^*\|^2 = \|A\hat{u} - b^*\|^2.$$

By definition, u^* satisfies

$$E(u^*) + \frac{\lambda}{2}\|Au^* - b^*\|^2 \leq E(\hat{u}) + \frac{\lambda}{2}\|A\hat{u} - b^*\|^2.$$

Consequently, $E(u^*) \leq E(\hat{u})$, which means that u^* is indeed a solution to (3.2.11). $\square$

Split Bregman Algorithm

We apply Bregman iterations to (3.2.7) with $E(d,u) = \mu \int_\Omega |d(x)|\, dx + \frac{1}{2}\|u - v\|^2$ and $A(d,u) = d - \Phi(u)$, $b = 0$ and $\lambda = \gamma > 0$. Then we have

$$\begin{cases} (d^{k+1}, u^{k+1}) & = \operatorname{argmin}_{d,u} \left(\mu \int_\Omega |d(x)|\, dx + \frac{1}{2}\|u - v\|^2 + \frac{\gamma}{2}\|d - \Phi(u) - b^k\|^2\right), \\ \quad b^{k+1} & = b^k - d^{k+1} - \Phi(u^{k+1}), \end{cases}$$

which can be approximately solved by following alternating minimization algorithm:

$$\begin{cases} d^{k+1} & = \operatorname{argmin}_d \mu \int_\Omega |d(x)|\, dx + \frac{\gamma}{2}\|d - \Phi(u^k) - b^k\|^2, \\ u^{k+1} & = \operatorname{argmin}_u \frac{1}{2}\|u - v\|^2 + \frac{\gamma}{2}\|d^{k+1} - \Phi(u) - b^k\|^2, \\ b^{k+1} & = b^k - d^{k+1} + \Phi(u^{k+1}). \end{cases}$$

Note that we can directly solve d-subproblem using soft shrinkage function defined by

$$\operatorname{shrink}(x, \mu) = \frac{x}{|x|}\max\{|x| - \mu, 0\}, \tag{3.2.12}$$

such that

$$d^{k+1} = \text{shrink}(\Phi(u^k) - b^k, \frac{\mu}{\gamma}).$$

Therefore, we have Algorithm 5.

Algorithm 5 Split Bregman Method for Denoising Problems

```
**Initialize:** $k = 0, u^0 = 0, d^0 = 0, b^0 = 0$ and $tol = 10^{-3}$
**while** $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol_{in}$ **do**
    $d^{k+1} \leftarrow \text{shrink}(\Phi(u^k) - b^k, \frac{\mu}{\gamma})$
    $u^{k+1} \leftarrow \text{argmin}_u \frac{1}{2}\|u - v\|^2 + \frac{\gamma}{2}\|d^{k+1} - \Phi(u) - b^k\|^2$
    $b^{k+1} \leftarrow b^k - d^{k+1} + \Phi(u^{k+1})$
    $k \leftarrow k + 1$
**end while**
**Output:** $u^k$
```

3.2.3 Algorithms

Here we suggest various algorithms based on FBS. At first, we need to determine Lipschitz constant $L > 0$ of L^tL. It can be computed using (3.1.3) and (2.3.3). When we use differentiable regularizers such as $J(u) = \frac{1}{2}\int_\Omega |\nabla u|^2$, also called H^1 norm, optimal condition of denoising part becomes

$$0 = \delta\mu\partial J(u^{k+1}) + u^{k+1} - v^{k+1}, \tag{3.2.13}$$

and this can be easily solved using fast methods like Jacobi, Gauss-Seidel iterative methods or simply using FFTs. For the simpliest case, when J is equal to H^1 semi-norm,

$$u^{k+1} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(v^{k+1})}{1 - \mu\delta\mathcal{F}(d)}\right),$$

where d is the convolution kernel of Laplacian operator. Hence FBS would be literally a simple 2-step algorithm:

$$\begin{cases} v^{k+1} &= u^k - \delta(L^tLu^k - L^ts), \\ u^{k+1} &= \mathcal{F}^{-1}\left(\frac{\mathcal{F}(v^{k+1})}{1-\mu\delta\mathcal{F}(d)}\right). \end{cases} \tag{3.2.14}$$

Therefore we get the simpliest image recovery algorithm:

Algorithm 6 FBS-H^1 based image recovery

```
**Initialize:** $k = 0, u^0 = 0, v^0 = 0, \delta < \frac{1}{L}$ and $tol = 10^{-3}$
**while** $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol$ **do**
    Solve (3.2.14),
    $\begin{cases} v^{k+1} & \leftarrow u^k - \delta(L^t L u^k - L^t s), \\ u^{k+1} & \leftarrow \mathcal{F}^{-1}\left(\frac{\mathcal{F}(v^{k+1})}{1-\mu\delta\mathcal{F}(d)}\right). \end{cases}$
    $k \leftarrow k + 1$
**end while**
**Output:** $u^k$
```

For optical systems having PSFs of small supports, above simple and low-cost implementation should be enough to recover true image. However when the noise level and PSF supports get bigger, recovery process requires a more complex procedure. In that case, we may require more sparsity on $|\nabla u|$ using l_1 regularizer like Total Variation(TV) model,

$$J_{TV}(u) = \int_\Omega |\nabla u(x)|\, dx,$$

or use texture-preserving denoising prior like NLH^1,

$$J_{NLH^1}(u) = \frac{1}{2}\int_\Omega |\nabla_w u(x)|^2\, dx,$$

or require both sparsity on ∇u and texture-preserving property using $NLTV$,

$$J_{NLTV}(u) = \int_\Omega |\nabla_w u(x)|\, dx.$$

In the case of NLH^1, $(1 + \delta\mu\partial J_{NLH^1}) = (1 - \delta\mu\Delta_w)$ is diagonally dominant, therefore we may use Jacobi's iterative method to (3.2.13):

$$u_0^{k+1} = v^{k+1}, \quad u_{l+1}^{k+1}(x) = \frac{v^{k+1} + 2\delta\mu\sum_{y\neq x} u_l^{k+1}(y)w(x,y)}{1 + 2\delta\mu\sum_{y\neq x} w(x,y)} \quad \text{for} \quad l = 0, 1, 2, \ldots.$$

Note that for non-local regularizers, we need to refer to the result of other algorithms in order to initialize the non-local weight w, which holds a finer structure than that of given blurry image. Hence we have the following algorithm:

Algorithm 7 FBS-NLH^1 based image recovery

```
Initialize: $k = 0, u^0 = 0, v^0 = 0, \delta < \frac{1}{L}$ and $tol_{in} = 10^{-3}, tol_{out} = 10^{-2}$
Initialize: $w$ (weight) using the result of $H^1$ based image recovery
while $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol_{in}$ do
    $v^{k+1} \leftarrow u^k - \delta(L^t L u^k - L^t s)$,
    $u_0^{k+1} \leftarrow v^{k+1}$
    $l = 0$
    while $\|u_l^{k+1} - u_{l-1}^{k+1}\|/\|u_{l-1}^{k+1}\| > tol_{out}$ do
        $u_{l+1}^{k+1}(x) \leftarrow \dfrac{v^{k+1} + 2\delta\mu \sum_{y \neq x} u_l^{k+1}(y) w(x,y)}{1 + 2\delta\mu \sum_{y \neq x} w(x,y)}$
        $l \leftarrow l + 1$
    end while
    $k \leftarrow k + 1$
end while
Output: $u^k$
```

Another good assumption on clean image is the sparsity of the light intensity. A convolution operator makes sharp signals, which have sparse support, into blurry signals and breaks the sparsity. The l_1 regularizer is the most commonly used to ensure the sparsity on recovered images. Our optimal condition for u can be solved by soft shrinkage operator (3.2.12). The updating step of u becomes

$$u^{k+1} = \text{shrink}(v^{k+1}, \delta\mu).$$

Therefore, we have the following algorithm:

Algorithm 8 FBS-l_1 based image recovery

```
Initialize: $k = 0, u^0 = 0, v^0 = 0, \delta < \frac{1}{L}$ and $tol = 10^{-3}$
while $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol$ do
    Solve (3.2.14),
    $\begin{cases} v^{k+1} & \leftarrow u^k - \delta(L^t L u^k - L^t s), \\ u^{k+1} & \leftarrow \text{shrink}(v^{k+1}, \delta\mu). \end{cases}$
    $k \leftarrow k + 1$
end while
Output: $u^k$
```

When we use J_{TV} or J_{NLTV} for our regularizer, we use Split-Bregman algorithm introduced in Algorithm 5 for the denoising problems.

Algorithm 9 FBS-$TV/NLTV$ based image recovery

```
Initialize: $k = 0, u^0 = 0, v^0 = 0, \delta < \frac{1}{L}$ and $tol = 10^{-3}$
Initialize: $w$ if $J = J_{NLTV}$ using the result of $H^1$ based image recovery
while $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol$ do
    Solve (3.2.14),
    $\begin{cases} v^{k+1} & \leftarrow u^k - \delta(L^t L u^k - L^t s), \\ u^{k+1} & \leftarrow \mathrm{Prox}_J(v^{k+1}) \quad \text{using Algorithm 5.} \end{cases}$
    $k \leftarrow k + 1$
end while
Output: $u^k$
```

Here, we present Split-Bregman implementation of $TV/NLTV$ denoiser:

Algorithm 10 Split-Bregman TV denoiser

```
Initialize: $k = 0, u^0 = 0, d^0 = 0, b^0 = 0$ and $tol_{in} = 10^{-3}$
while $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol_{in}$ do
    $d^{k+1} \leftarrow \mathrm{shrink}(\nabla u^k - b^k, \frac{\mu}{\gamma})$
    $u^{k+1} \leftarrow \mathrm{argmin}_u \frac{1}{2}\|u - v\|^2 + \frac{\gamma}{2}\|d^{k+1} - \nabla u - b^k\|^2$ by Gauss-Seidel
    $b^{k+1} \leftarrow b^k - d^{k+1} + \nabla u^{k+1}$
    $k \leftarrow k + 1$
end while
Output: $u^k$
```

Algorithm 11 Split-Bregman $NLTV$ denoiser

```
Initialize: $k = 0, u^0 = 0, d^0 = 0, b^0 = 0$, weight $w$ and $tol_{in} = 10^{-3}$
while $\|u^k - u^{k-1}\|/\|u^{k-1}\| > tol_{in}$ do
    $d^{k+1} \leftarrow \mathrm{shrink}(\nabla_w u^k - b^k, \frac{\mu}{\gamma})$
    $u^{k+1} \leftarrow \mathrm{argmin}_u \frac{1}{2}\|u - v\|^2 + \frac{\gamma}{2}\|d^{k+1} - \nabla_w u - b^k\|^2$ by Jacobi
    $b^{k+1} \leftarrow b^k - d^{k+1} + \nabla_w u^{k+1}$
    $k \leftarrow k + 1$
end while
Output: $u^k$
```

Note that we use different solvers to the u-subproblem. This difference is due to the parallelization of NL operators. For the case of $NLTV$, note that the optimal condition for the u-subproblem is equivalent to

$$u^{k+1} = \mathrm{Prox}_{\gamma J_{NLH^1}}(v - \gamma \mathrm{div}_w(d^{k+1} - b^k)),$$

which can be solved in the same manner as in the NLH^1 case.

Chapter 4

Experiments

4.1 Implementation Details

All of the experiments were done on CPU: *Intel 5th generation Core i7-6800K (3.4Ghz-3.6Ghz, 6 cores 12 threads)*. For the parallelization on non-local algorithms, we use GPGPU: *NVIDIA Geforce GTX1080ti with 11GB GDDR5 RAM*. Vectorized algorithms using CPUs are written in *MATLAB* (PSF generation) and *python with numpy* (recovery process), while algorithms using GPGPU are written in *OpenCL* and *pyOpenCL* (python wrapper of *OpenCL*).

4.1.1 Generation of synthetic blurry images

To simulate lens aberration, the first thing to do is computing the convolution kernel. This can be done using the notion of array C in Section 2.2. Note that we use $11 \times 11 \times 512 \times 512$ sized array for A and C since python uses C-contiguous array. We use

$$C(\cdot, y) = A(\cdot + y, -y),$$

for vectorization. The following code is an example of a vectorized implementation of operator transpose in python code:

```
def op_transpose(mat4d):
    kernel = np.zeros(mat4d.shape).astype(np.float32)
    half = int(mat4d.shape[0]/2.)
    for dx in np.arange(-half,half+1):
        zmin=max(0,dx)
        zmin1=max(0,-dx)
        zmax=min(mat4d.shape[2],mat4d.shape[2]+dx)
        for dy in np.arange(-half,half+1):
            wmin=max(0,dy)
            wmin1=max(0,-dy)
            wmax=min(mat4d.shape[3],mat4d.shape[3]+dy)
            kernel[half+dx,half+dy,:,:] = \
                np.pad(mat4d[half-dx,half-dy,zmin:zmax,wmin:wmax],
                 ((zmin1,zmin),(wmin1,wmin)),'edge')
    return kernel
```

The python function “op_transpose” gets a 4-d array “mat4d”, which is the array A as an input and returns the 4-d array “kernel”, which is the array C. Note that the algorithm is fully vectorized on the image domain and repeats “for loops” only in 11×11 times.

As in the computation of C from A, aberration simulation can be done in a vectorized way:

1. Initialize a temporary image tmp of size 512×512 filled with 0s.

2. For vectors y inside the box $[-w, w]^2$, add $C(\cdot, y)I(\cdot + y)$ to tmp, where I is the image.

Implementation of convolution is as follows:

```
def convolve(img,kernel):
    assert img.shape[:] == kernel.shape[2:]
    img = img.astype(np.float32)
    tmp = np.zeros(img.shape).astype(np.float32)
    half = int(kernel.shape[0]/2)
    for dx in np.arange(-half, half+1):
        zmin=max(0,dx)
        zmin1=max(0,-dx)
        zmax=min(kernel.shape[2],kernel.shape[2]+dx)
        for dy in np.arange(-half, half+1):
            wmin=max(0,dy)
            wmin1=max(0,-dy)
            wmax=min(kernel.shape[3],kernel.shape[3]+dy)
            tmp+= kernel[half+dx,half+dy,:,:]*\
                np.pad(img[zmin:zmax,wmin:wmax],
                ((zmin1,zmin),(wmin1,wmin)), 'edge')
    return tmp
```

The python function "convolve" gets 2-d array "img", which is the given image and 4-d array "kernel", which is the convolution kernel as inputs and returns 2-d array "tmp", which is the blurry image. The returned array "tmp" becomes the synthetically aberrated image s_0. Then we add Gaussian noise n to s_0 to get s. We use standard test images with size 512×512.

4.2 Numerical Results

In the experiments, we tested our methods on standard test images. Figure 4.1 lists standard test images.

(a) woman_blonde.tif

(b) woman_darkhair.tif

(c) walkbridge.tif

(d) livingroom.tif

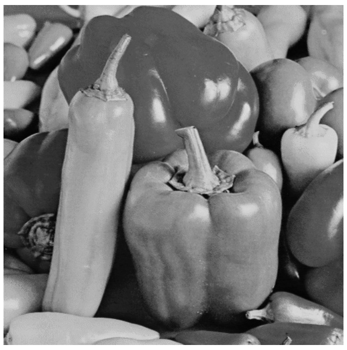
(e) pirate.tif

(f) mandrill_color.tif

(g) peppers_color.tif

(h) lena_color_512.tif

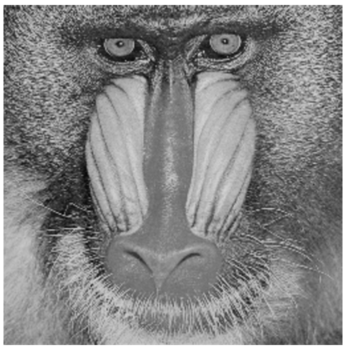
(i) lena_gray_512.tif


(j) peppers_gray.tif

(k) lake.tif

(l) cameraman.tif

(m) jetplane.tif


(n) mandrill_gray.tif

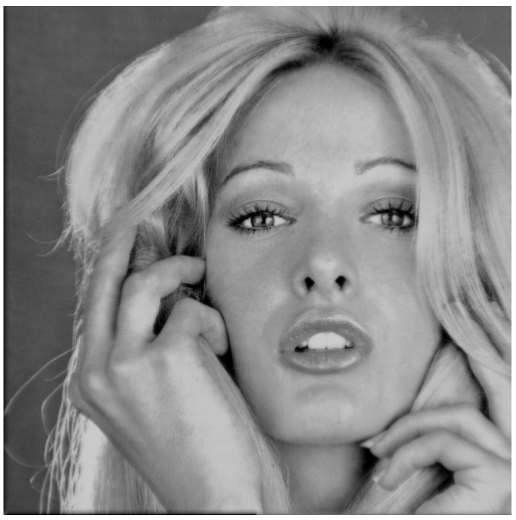
(o) house.tif

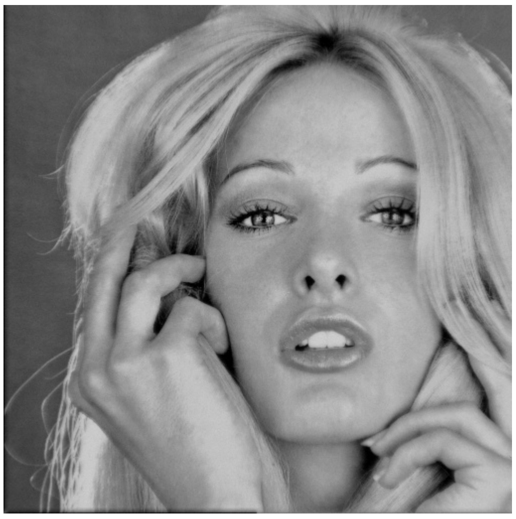
Figure 4.1: Standard test images.

4.2.1 Synthetically Blurred Images

Noise levels of given Gaussian noise, which are equal to standard deviations, are chosen such that $\sigma_{\text{noise}} \in (1, 2, 4)$. See Figure 4.2 for an example. As you can see, detailed textures are getting smoother as it goes further to the edges. The PSNR values of synthetically blurred images are in Table 4.1.

(a) Original


(b) Blurred with big PSF, noise $\sigma = 1$
PSNR: 29.39


(c) Blurred with big PSF, noise $\sigma = 2$
PSNR: 29.22

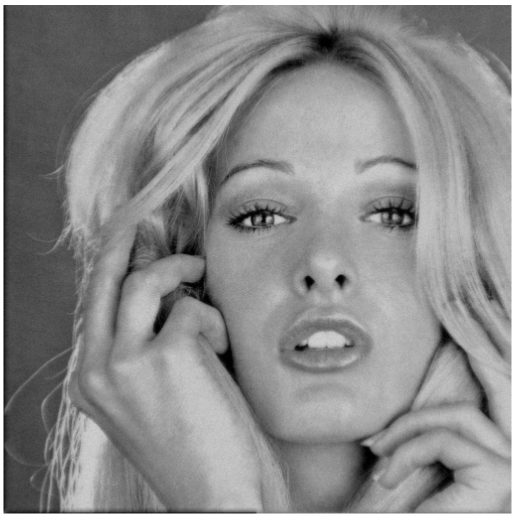

(d) Blurred with big PSF, noise $\sigma = 4$
PSNR: 28.59

Figure 4.2: Blurred “woman_blonde.tif” images with big PSF.

Image Name	Big PSF			Small PSF		
Noise σ	1	2	4	1	2	4
woman_blonde.tif	29.39	29.22	28.59	34.91	34.34	32.59
woman_darkhair.tif	35.46	34.8	32.88	41.4	39.25	35.21
walkbridge.tif	27.59	27.47	27.05	33.48	33.04	31.68
livingroom.tif	29.72	29.54	28.87	36.29	35.51	33.32
pirate.tif	30.28	30.07	29.33	36.84	35.99	33.59
mandrill_color.tif	25.02	24.95	24.71	30.66	30.43	29.62
peppers_color.tif	30.41	30.19	29.43	36.31	35.53	33.36
lena_color_512.tif	32.05	31.74	30.68	39.31	37.87	34.59
lena_gray_512.tif	32.14	31.82	30.76	39.64	38.11	34.69
peppers_gray.tif	31.14	30.89	30.01	37.27	36.32	33.79
lake.tif	28.44	28.3	27.8	34.61	34.06	32.38
cameraman.tif	32.77	32.4	31.2	40.39	38.6	34.93
jetplane.tif	31.41	31.14	30.21	38.47	37.25	34.29
mandrill_gray.tif	28.18	28.04	27.55	34.11	33.62	32.08
house.tif	34.37	33.84	32.24	41.9	39.55	35.29

Table 4.1: PSNR values of synthetically blurred images with big and small PSFs.

4.2.2 Image Restoration

We now present our major results on image restoration. The intervals of μ are taken such that PSNR values reach the peak. Other parameters are chosen empirically. We use "woman_blonde.tif" image to observe noise sensitivity/robustness of suggested methods and other 4 images "lena_gray_512.tif, mandrill_gray.tif, lake.tif, livingroom.tif" to compare results of suggested methods.

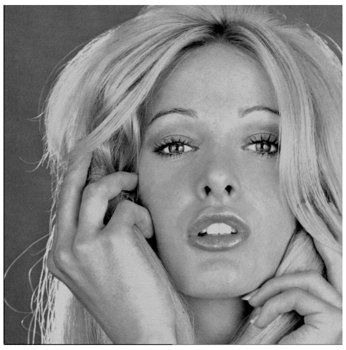
Van Cittert's iteration (Iterative Least Square Method)

If we let $J \equiv 0$ in Algorithm 3, we have Van Cittert(VC) type iterative solution to

$$L^t Lu = L^t s,$$

without any consideration on the noise n. Experiments performed with setting $\delta = 0.9$. Van Cittert's iteration restores blurred images as well as other regularized methods when small noise σ is given. However, in big noise σ, it enhances not only detailed textures, but also noise intensity (Figure 4.3).

(a) $\sigma = 1$, PSNR=38.80


(b) $\sigma = 2$, PSNR=34.26

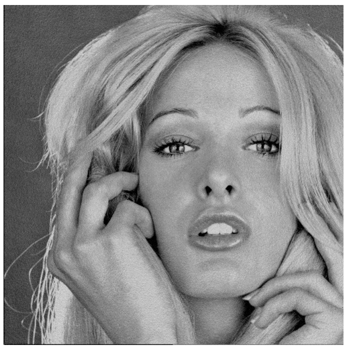
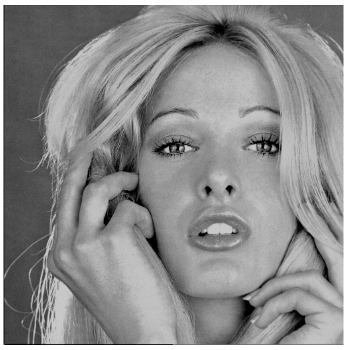
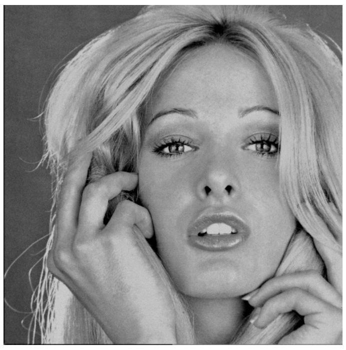

(c) $\sigma = 4$, PSNR=28.24

Figure 4.3: Results of Van Cittert's iteration.

FBS-H^1

We implemented Algorithm 6 with $\delta = 0.9$, $\mu \in (10^{-5}, 10^{-1})$. It shows better results than Van Cittert's iteration when σ is large.


(a) $\sigma = 1, \mu = 10^{-5}$, PSNR=38.80


(b) $\sigma = 2, \mu = 10^{-2}$, PSNR=34.56

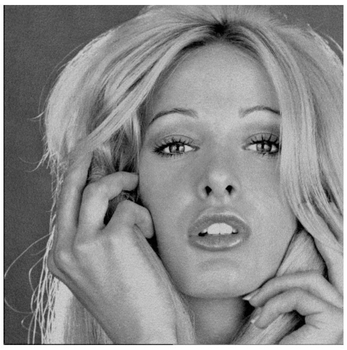
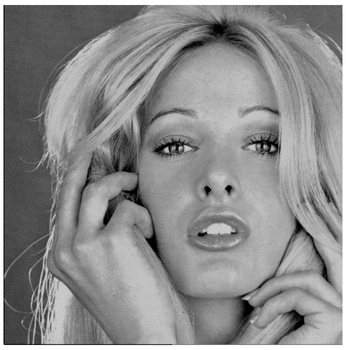

(c) $\sigma = 4, \mu = 10^{-2}$, PSNR=30.31

Figure 4.4: Results of FBS-H^1.

FBS-l_1

We implemented Algorithm 8 with the same parameters as in FBS-H^1. Restoration performance of the algorithm is superior to Van Cittert's iteration, but inferior to FBS-H^1.

(a) $\sigma = 1, \mu = 10^{-2}$, PSNR=38.80


(b) $\sigma = 2, \mu = 10^{-2}$, PSNR=34.26

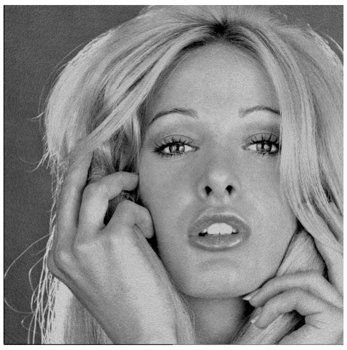
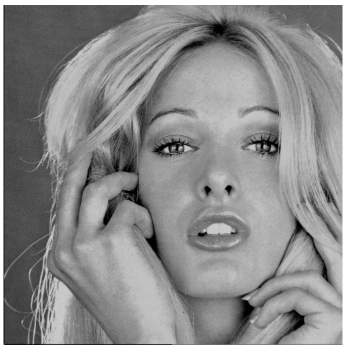

(c) $\sigma = 4, \mu = 10^{-1}$, PSNR=28.24

Figure 4.5: Results of FBS-l_1.

FBS-TV

We implemented the TV version of Algorithm 9 with $\delta = 0.9$, $\mu \in (10^{-5}, 10^{1})$. Bregman parameter $\gamma = 0.2$ was chosen. While it has a longer running time than that of FBS-H^1, FBS-TV shows better results in the cases of large noise levels.

(a) $\sigma = 1, \mu = 10^{-2}$, PSNR=38.87


(b) $\sigma = 2, \mu = 10^{-1}$, PSNR=35.32

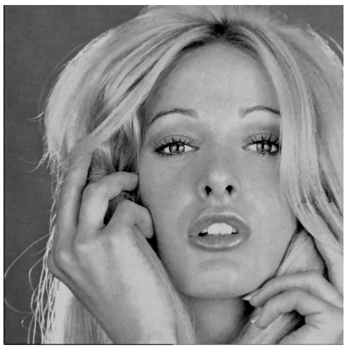
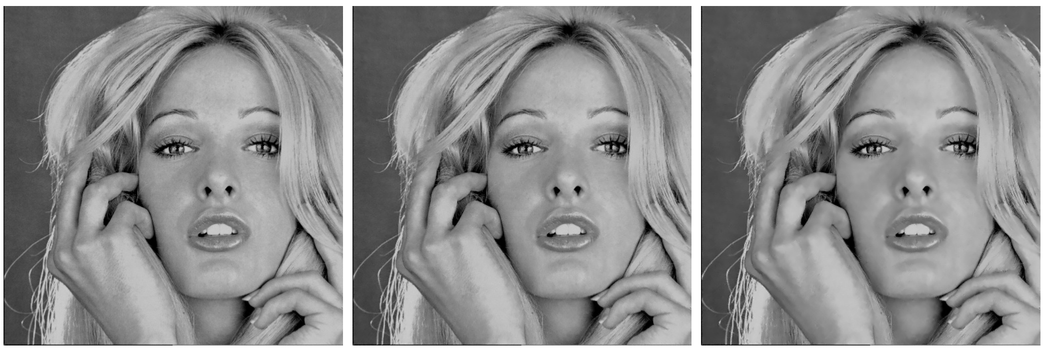

(c) $\sigma = 4, \mu = 1$, PSNR=32.43

Figure 4.6: Results of FBS-TV.

FBS-NLH^1

We implemented Algorithm 7 with $\delta = 0.9$, $\mu \in (0.01, 2.0)$. For non-local operators, we used *OpenCL* and *PyOpenCL* to accelerate convolution and non-local operations. Non-local patch size is 5×5 and the reference region size is 7×7. We put $h = 10$ and $a = 2$ to get the non-local weight robust to noise intensity.

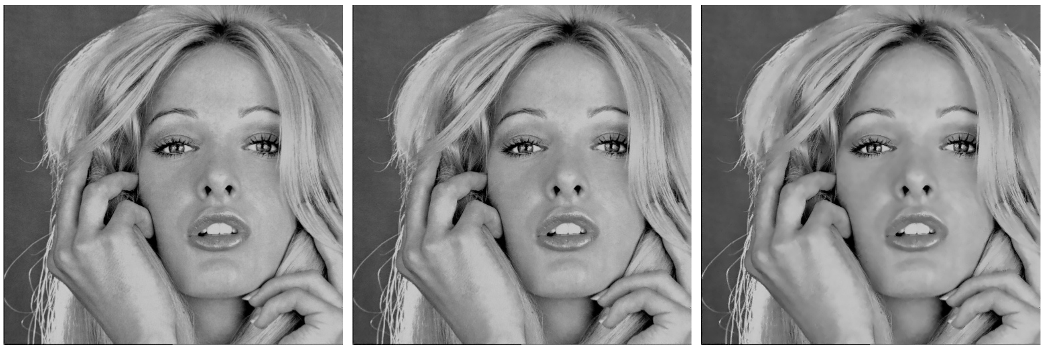


(a) $\sigma = 1, \mu = 0.02$, PSNR=39.01 (b) $\sigma = 2, \mu = 0.1$, PSNR=36.28 (c) $\sigma = 4, \mu = 1.2$, PSNR=33.41

Figure 4.7: Results of FBS-NLH^1.

FBS-$NLTV$

We implemented $NLTV$ version of Algorithm 9 with $\delta = 0.9$, $\mu \in (1, 80)$. Bregman parameter γ was chosen $\gamma = \mu/50$ so that the convergence of Bregmanized constraint $\nabla_w u = d$ is guaranteed. Non-local methods have the best results when large σ and detailed textures are given.



(a) $\sigma = 1, \mu = 1$ PSNR=39.01 (b) $\sigma = 2, \mu = 4$ PSNR=36.21 (c) $\sigma = 4, \mu = 64$ PSNR=33.45

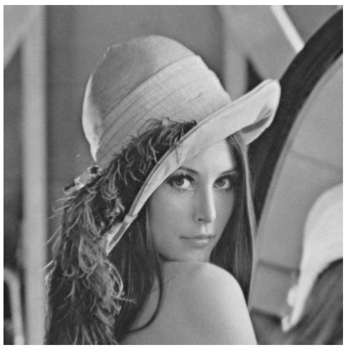
Figure 4.8: Results of FBS-$NLTV$.

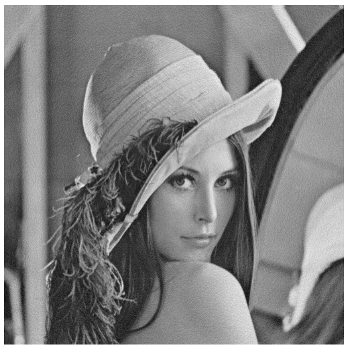
Results Comparison for big noise σ

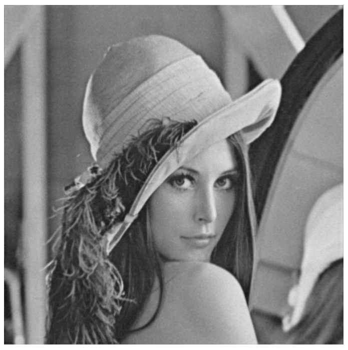
Here we present results on "lena_gray_512.tif, mandrill_gray.tif, lake.tif, livingroom.tif". For the small noise σ, all methods recover original images sharp and clearly. However, when $\sigma = 4$, the recovery PSNR value depends on the method. It is remarkable that the TV method has best PSNR values for some test images even though it has less computational complexity than that of non-local methods. While it has better PSNR values on some images, one can easily find that its results are still more blunt than the results of non-local algorithms. Texture enhanced results of non-local algorithms seem more clear and sharpening to human eyes. In running time comparison, the algorithms implemented with GPU, VC, NLH^1, $NLTV$ take 110ms, 152ms, 612ms (for each channel) in average respectively. For the algorithms implemented with CPU only, H^1, l_1, TV take 1.5s, 2.8s, 2.4s (for each channel) in average respectively.

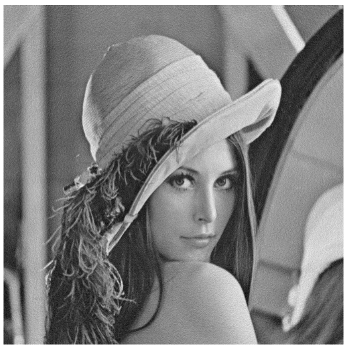
"Lena", one of the most famous test images, can be clearly partitioned into faces and lines. Consequently, non-local algorithms, which have a strong texture enhancing property, show better results than local algorithms. "Mandrill" test image (including colored version) contains so small detailed textures that cannot be distinguishable from noise. Hence, it has the worst PSNR values among test images when $\sigma = 4$. FBS-TV has the best PSNR value because non-local algorithms remove not only textures, but also noises. In the "lake" image, trees lying horizontally on the central region have blunt textures. For this reason, FBS-TV resulted as the best PSNR value. However, small noise patterns remained visible in the sky region.

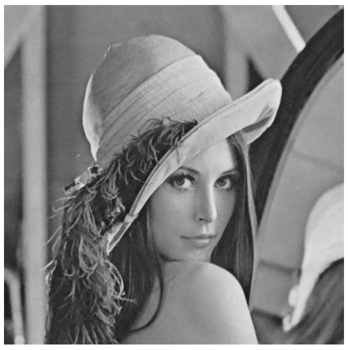
(a) Original

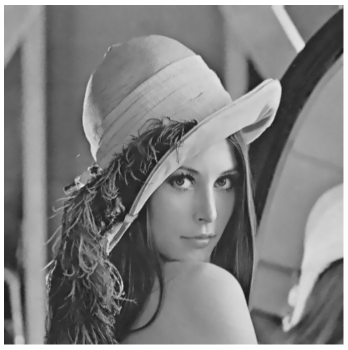

(b) Blurred, noise $\sigma = 4$
PSNR: 30.76


(c) Van Cittert's iteration
PSNR: 28.44


(d) FBS-H^1, $\mu = 0.1$
PSNR: 34.13


(e) FBS-l_1, $\mu = 0.001$
PSNR: 28.44


(f) FBS-TV, $\mu = 1.0$
PSNR: 35.48


(g) FBS-NLH^1, $\mu = 1.0$
PSNR: 35.62

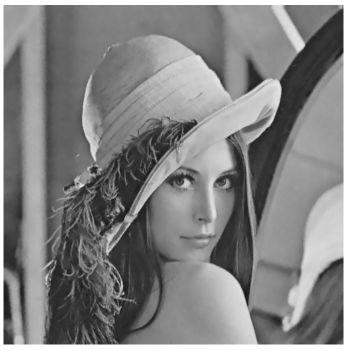
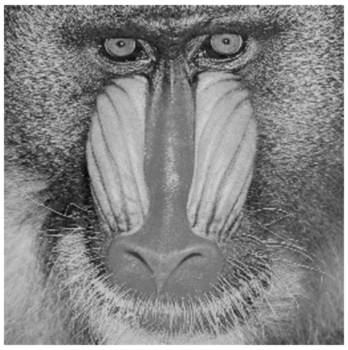
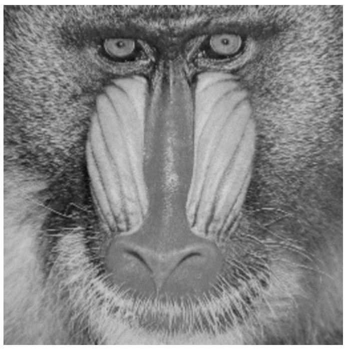
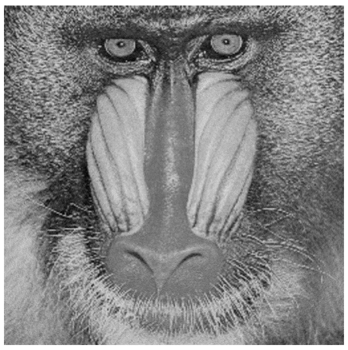
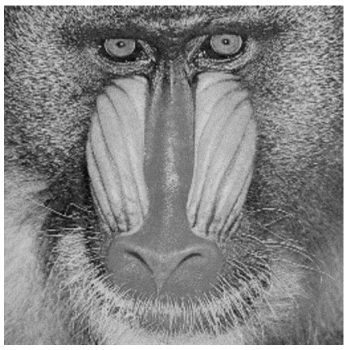

(h) FBS-$NLTV$, $\mu = 48$
PSNR: 35.62

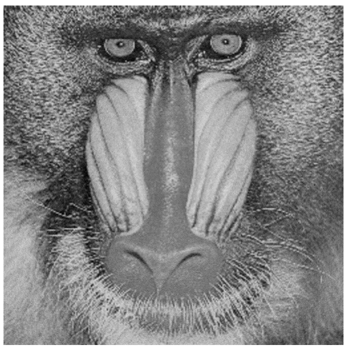
Figure 4.9: Restoration results of "lena_gray_512.tif" blurred with big PSF and noise $\sigma = 4$.

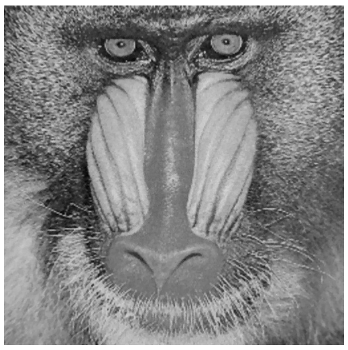

(a) Original

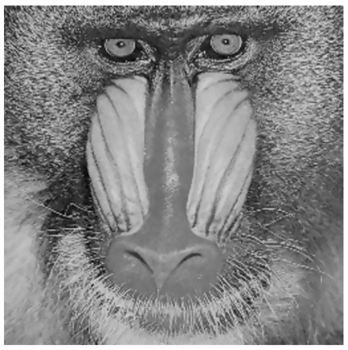

(b) Blurred, noise $\sigma = 4$
PSNR: 27.55


(c) Van Cittert's iteration
PSNR: 28.22


(d) FBS-H^1, $\mu = 0.01$
PSNR: 30.54


(e) FBS-l_1, $\mu = 0.01$
PSNR: 28.22


(f) FBS-TV, $\mu = 1.0$
PSNR: 31.91


(g) FBS-NLH^1, $\mu = 0.9$
PSNR: 31.24

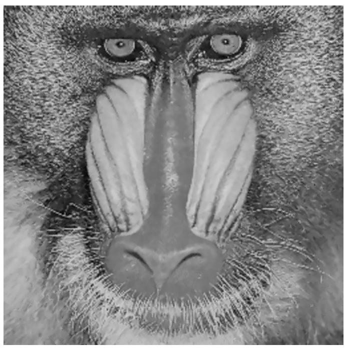
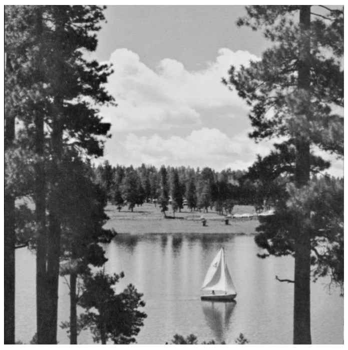
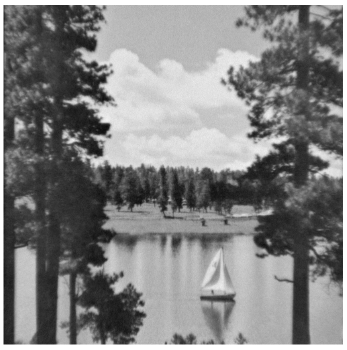
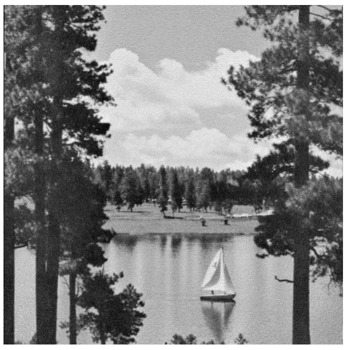
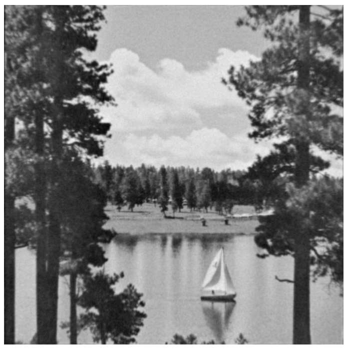

(h) FBS-$NLTV$, $\mu = 40$
PSNR: 31.20

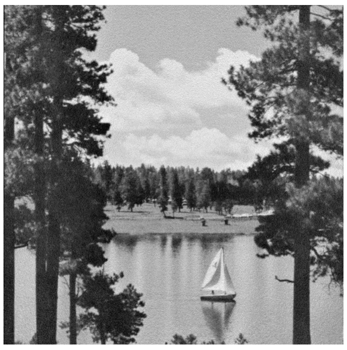
Figure 4.10: Restoration results of "mandrill_gray.tif" blurred with big PSF and noise $\sigma = 4$.

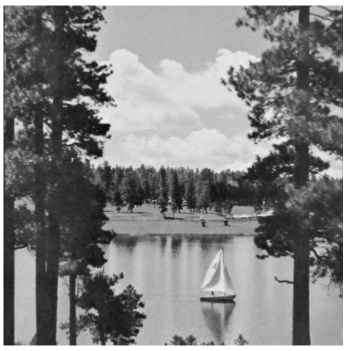

(a) Original

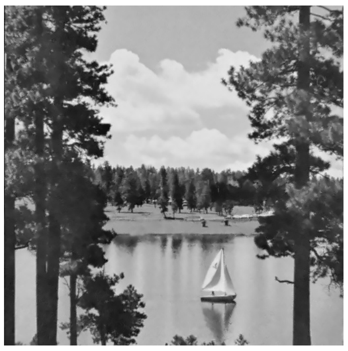

(b) Blurred, noise $\sigma = 4$
PSNR: 27.80


(c) Van Cittert's iteration
PSNR: 28.29


(d) FBS-H^1, $\mu = 0.1$
PSNR: 31.20


(e) FBS-l_1, $\mu = 0.01$
PSNR: 28.29


(f) FBS-TV, $\mu = 1.0$
PSNR: 32.95


(g) FBS-NLH^1, $\mu = 1.0$
PSNR: 32.62

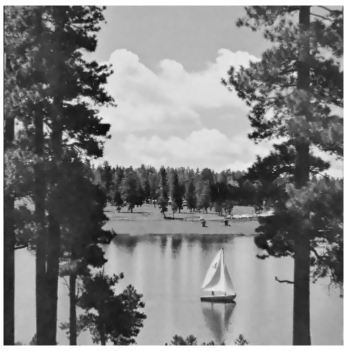
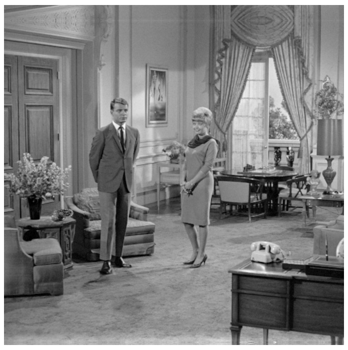
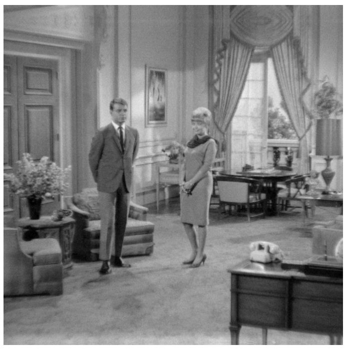
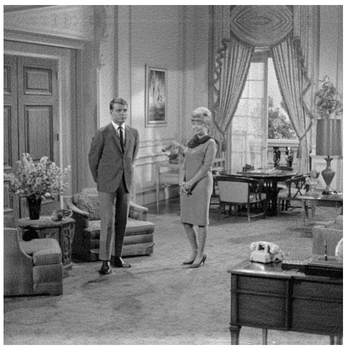
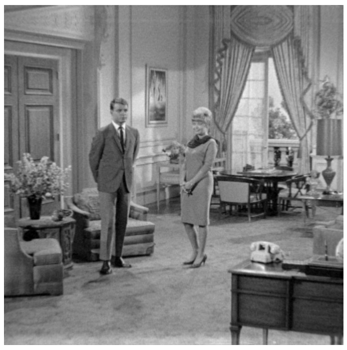

(h) FBS-$NLTV$, $\mu = 48$
PSNR: 32.63

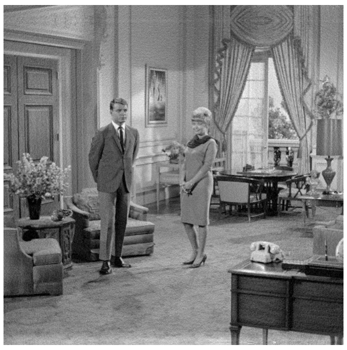
Figure 4.11: Restoration results of "lake.tif" blurred with big PSF and noise $\sigma = 4$.

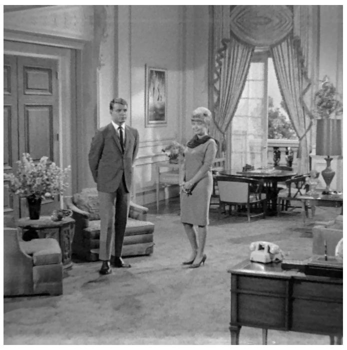

(a) Original

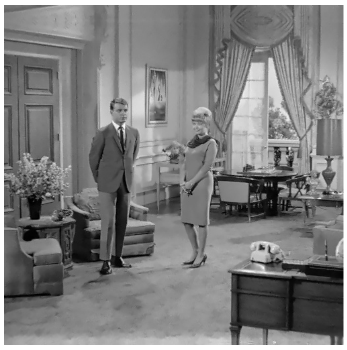

(b) Blurred, noise $\sigma = 4$
PSNR: 28.87


(c) Van Cittert's iteration
PSNR: 28.27


(d) FBS-H^1, $\mu = 0.1$
PSNR: 31.31


(e) FBS-l_1, $\mu = 0.1$
PSNR: 28.28


(f) FBS-TV, $\mu = 1.0$
PSNR: 32.99


(g) FBS-NLH^1, $\mu = 1.0$
PSNR: 33.13

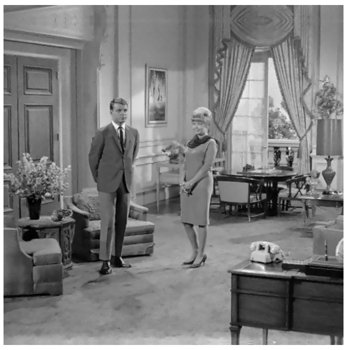

(h) FBS-$NLTV$, $\mu = 40$
PSNR: 33.14

Figure 4.12: Restoration results of "livingroom.tif" blurred with big PSF and noise $\sigma = 4$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	41.57	42.15	41.57	43.14	**43.15**	43.14
woman_blonde.tif	38.80	38.80	38.80	38.87	**39.01**	**39.01**
walkbridge.tif	38.86	38.86	38.86	38.87	**38.93**	38.87
pirate.tif	39.91	39.91	39.91	40.15	**40.48**	40.43
house.tif	43.09	44.12	43.09	**45.39**	45.31	45.22
woman_darkhair.tif	42.11	42.91	42.11	**43.63**	43.62	43.62
mandrill_gray.tif	39.65	39.65	39.65	39.70	**39.89**	**39.89**
lena_color_512.tif	40.72	40.76	40.72	40.81	**40.93**	40.92
lake.tif	39.21	39.21	39.21	39.21	**39.35**	**39.35**
peppers_color.tif	39.45	39.45	39.46	**39.47**	39.25	38.81
lena_gray_512.tif	41.01	41.11	41.01	41.55	**41.66**	**41.66**
peppers_gray.tif	40.10	40.10	40.10	40.12	**40.22**	40.10
jetplane.tif	41.10	41.14	41.10	**42.02**	41.99	41.92
mandrill_color.tif	**37.83**	37.82	**37.83**	**37.83**	37.44	36.96
livingroom.tif	39.92	39.92	39.92	40.00	**40.15**	**40.15**

Table 4.2: Restored PSNR values from images blurred by big PSF, $\sigma = 1$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	35.31	37.33	35.31	39.42	**40.17**	**40.17**
woman_blonde.tif	34.26	34.56	34.26	35.32	**36.27**	36.21
walkbridge.tif	34.03	34.20	34.03	34.68	**35.02**	35.01
pirate.tif	34.54	35.71	34.54	35.92	36.99	**37.04**
house.tif	35.99	40.54	35.99	**43.02**	42.31	42.31
woman_darkhair.tif	35.33	39.67	35.33	40.91	**41.05**	**41.05**
mandrill_gray.tif	34.46	35.48	34.46	35.33	**35.81**	35.78
lena_color_512.tif	35.04	36.51	35.04	36.58	**37.54**	37.50
lake.tif	34.44	35.21	34.44	35.48	**36.11**	36.08
peppers_color.tif	34.47	35.18	34.49	35.39	**35.63**	**35.63**
lena_gray_512.tif	35.10	36.80	35.10	37.25	**38.65**	38.63
peppers_gray.tif	34.76	35.99	34.76	36.04	**36.84**	36.82
jetplane.tif	35.75	37.08	35.75	38.40	**39.13**	39.05
mandrill_color.tif	33.49	33.51	33.49	33.58	**33.70**	33.68
livingroom.tif	34.58	35.66	34.58	35.82	**36.60**	36.56

Table 4.3: Restored PSNR values from images blurred by big PSF, $\sigma = 2$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	28.55	35.30	28.56	36.87	**36.90**	36.81
woman_blonde.tif	28.24	30.31	28.24	32.43	33.41	**33.45**
walkbridge.tif	28.09	30.05	28.09	30.58	**30.63**	**30.63**
pirate.tif	28.19	31.93	28.19	**33.56**	33.50	33.40
house.tif	28.63	37.01	28.63	38.91	39.23	**39.35**
woman_darkhair.tif	28.37	36.63	28.37	38.03	38.49	**38.53**
mandrill_gray.tif	28.22	30.54	28.22	**31.91**	31.27	31.20
lena_color_512.tif	28.41	33.47	28.41	34.60	**34.86**	34.62
lake.tif	28.29	31.20	28.29	**32.95**	32.62	32.63
peppers_color.tif	28.34	31.71	28.35	32.43	**32.65**	32.53
lena_gray_512.tif	28.44	34.13	28.44	35.47	**35.66**	35.62
peppers_gray.tif	28.41	33.22	28.41	**34.19**	34.15	34.12
jetplane.tif	28.85	34.30	28.85	**36.12**	36.04	36.01
mandrill_color.tif	27.89	**28.92**	27.89	28.53	28.90	28.91
livingroom.tif	28.27	31.31	28.28	32.99	33.13	**33.14**

Table 4.4: Restored PSNR values from images blurred by big PSF, $\sigma = 4$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	45.22	45.79	45.22	46.50	**46.65**	46.60
woman_blonde.tif	44.55	44.55	44.55	44.72	**44.84**	**44.84**
walkbridge.tif	44.50	44.50	44.50	44.52	**44.62**	44.57
pirate.tif	44.84	44.88	44.84	**45.28**	**45.28**	45.27
house.tif	45.84	46.37	45.84	**47.73**	47.50	47.56
woman_darkhair.tif	45.46	45.94	45.46	46.47	**46.68**	46.66
mandrill_gray.tif	44.81	44.85	44.81	**45.02**	**45.02**	**45.02**
lena_color_512.tif	45.10	45.14	45.10	**45.39**	45.35	45.35
lake.tif	44.66	44.66	44.66	44.69	**44.83**	**44.83**
peppers_color.tif	44.67	44.67	44.67	**44.71**	44.67	44.37
lena_gray_512.tif	45.20	45.45	45.20	45.76	**45.79**	**45.79**
peppers_gray.tif	44.77	44.78	44.77	**44.91**	**44.91**	**44.91**
jetplane.tif	45.18	45.37	45.18	**45.97**	45.96	45.94
mandrill_color.tif	**44.46**	**44.46**	**44.46**	**44.46**	44.21	43.61
livingroom.tif	44.77	44.80	44.77	45.01	**45.02**	**45.02**

Table 4.5: Restored PSNR values from images blurred by small PSF, $\sigma = 1$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	39.28	40.64	39.28	**42.93**	42.73	42.73
woman_blonde.tif	38.86	39.13	38.86	39.46	**40.26**	40.21
walkbridge.tif	38.77	38.89	38.77	39.09	**39.36**	39.33
pirate.tif	38.95	39.54	38.95	39.67	40.55	**40.62**
house.tif	39.52	42.32	39.52	**44.98**	43.79	43.79
woman_darkhair.tif	39.30	41.92	39.30	**43.11**	42.96	42.96
mandrill_gray.tif	38.97	39.50	38.97	39.35	**39.68**	**39.68**
lena_color_512.tif	39.17	39.77	39.17	39.83	**40.60**	40.56
lake.tif	38.88	39.27	38.88	39.35	**39.87**	39.86
peppers_color.tif	38.92	39.20	38.92	39.38	**39.50**	**39.50**
lena_gray_512.tif	39.27	39.89	39.27	40.87	**41.50**	41.47
peppers_gray.tif	38.97	39.51	38.97	39.58	**40.17**	40.16
jetplane.tif	39.25	40.00	39.25	**41.99**	41.91	41.89
mandrill_color.tif	38.71	38.74	38.71	**38.83**	38.81	38.77
livingroom.tif	38.95	39.44	38.95	39.48	**40.09**	40.08

Table 4.6: Restored PSNR values from images blurred by small PSF, $\sigma = 2$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
cameraman.tif	33.01	37.00	33.01	38.04	**38.95**	38.91
woman_blonde.tif	32.91	33.65	32.91	35.81	**36.16**	36.06
walkbridge.tif	32.82	33.51	32.82	**34.26**	34.16	33.99
pirate.tif	32.91	34.92	32.91	36.24	**36.27**	36.23
house.tif	32.99	37.63	32.99	38.90	**40.15**	39.99
woman_darkhair.tif	33.01	37.50	33.01	38.34	**39.73**	39.60
mandrill_gray.tif	32.87	34.06	32.87	**35.07**	34.48	34.39
lena_color_512.tif	32.97	35.90	32.97	36.64	**36.85**	36.52
lake.tif	32.89	34.26	32.89	**35.58**	35.41	35.21
peppers_color.tif	32.95	34.16	32.96	34.98	**35.08**	34.84
lena_gray_512.tif	32.99	36.38	32.99	37.20	**37.75**	37.64
peppers_gray.tif	32.90	35.40	32.90	36.01	**36.19**	36.08
jetplane.tif	33.10	36.49	33.10	37.65	**38.07**	38.06
mandrill_color.tif	32.75	33.17	32.75	33.00	**33.20**	**33.20**
livingroom.tif	32.89	34.48	32.89	**35.80**	35.76	35.67

Table 4.7: Restored PSNR values from images blurred by small PSF, $\sigma = 4$.

Method	VC	H^1	l_1	TV	NLH^1	$NLTV$
Iterations	14	12	15	12	11	13
$\mu(\sigma = 1)$	N	0.00295	0.00440	0.0927	0.0370	1.97
$\mu(\sigma = 2)$	N	0.0271	0.00915	0.430	0.156	7.53
$\mu(\sigma = 4)$	N	0.0790	0.0240	0.940	0.966	33.5

Table 4.8: Average iterations and optimal values of parameter μ.

Tables 4.2-4.7 show best PSNR values of each method for all σ's and PSFs. Emboldened PSNR values in each row has the best results on the test image. The tables show that, even when PSFs are widely spread, small amount of noises do not seem to be an obstacle when we restore the clear image without any regularization (Van Cittert's iteration). However, when the noise grows to $\sigma = 2$, the PSNR value gap between the methods starts to rise. It is up to 6dB ("house.tif"). When $\sigma = 4$, the PSNR value gap is far bigger than that of $\sigma = 2$ case. It has grown up to 10dB ("house.tif, woman_darkhair.tif"). As mentioned above, experiments on "mandrill" images show the worst recovery results. For narrowly supported PSFs, according to Table 4.5-4.7, the PSNR value gap occurs similarly to the case of big PSFs. Among all experiments, most of the best PSNR values are obtained with the FBS-NLH^1 method. Table 4.8 shows average iterations and optimal values of parameter μ for each restoration method.

Effect of Vectorization Techniques

In this section, we show the effect of vectorization techniques introduces in Section 3.1.1. The following table shows mean running time for the task:

$$\text{Output} = L[\text{Input}],$$

where L is the s.v.c. operator corresponding to the big PSF.

Method	CPU, non-SIMD	CPU, with SIMD	GPGPU
Mean Running Time (s)	0.65	0.041	0.00064

Table 4.9: Mean running time comparison.

With the vectorization technique, we reduce the running time of s.v.c. operation to 1/16 times. If we use GPGPU, we can further accelerate the computation and the running time reduces to another 1/64 times.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we reviewed Computational Fourier Optics in Chapter 2 as a method to simulate monochromatic aberrations. First, we described aberration of the optical system using the notion of PSFs and spatially variant convolutions. Then, we proposed mathematical models and introduced preliminaries for aberration correction in Chapter 3. For implementation, we used small support assumption on PSFs to reduce computational cost. Then, we proposed 5 types of regularizers and associated algorithms. We also discussed vectorization technique on convolution operations to reduce computational complexity.

In Chapter 4, we first generated two types of PSFs (with big and small support) based on theory introduced in Chapter 2. Later, we created synthetically blurred images using generated PSFs with 3 different noise levels. Then, we showed noise robustness/sensitivity of suggested algorithms including Van Cittert's iteration ($J \equiv 0$). After we listed up the results of the algorithms separately, we presented the results on other sample images for readers to compare performances of suggested algorithms. While the proposed algorithms performed well with the small noise level, performance gap between the best and the worst appeared to grow as the noise level got higher.

After a series of experiments, we can conclude that with the aids of mathematical modeling and computation optimizing techniques, we can efficiently and successfully solve monochromatic aberration correction problems. Note further that proposed methods in this thesis can be generalized to *spatially variant deconvolution* problems.

5.2 Future Work

Our proposed methods can be utilized as an efficient tool to the following problems.

- Blind monochromatic aberration correction:
  Our methods focus on non-blind monochromatic aberration correction since PSFs are given, as future work research on blind monochromatic correction could be done. The problem can be described as follows: Find a pair (u, k) such that

  $$s(x) = \int_{\Omega} k(x,y)u(y)\,dy + n(x).$$

  There are infinitely many pairs (u, k) solving the equation above. We need a proper prior for both u and k.

- Chromatic aberration correction:
  In this thesis, we covered only monochromatic aberrations. However, chromatic aberrations (Figure 5.1) can be also covered.

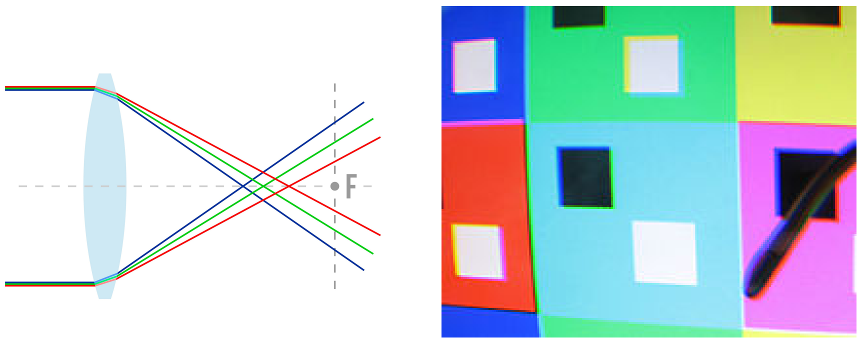


Figure 5.1: Chromatic aberrations.

Yue et al. [26] used *warping and splitting* method to solve blind chromatic aberration correction. We expect that using direct computation of *spatially variant convolutions* in our methods, we can improve Yue et al. [26].

- Camera shake correction:
  Since a camera sensor has a positive area, the camera shake also can be modelled by *spatially variant convolutions*. We can consider this problem as a blind deconvolution problem with *spatially variant convolutions*. We may refer Zhu et al. [29] for estimating PSFs.

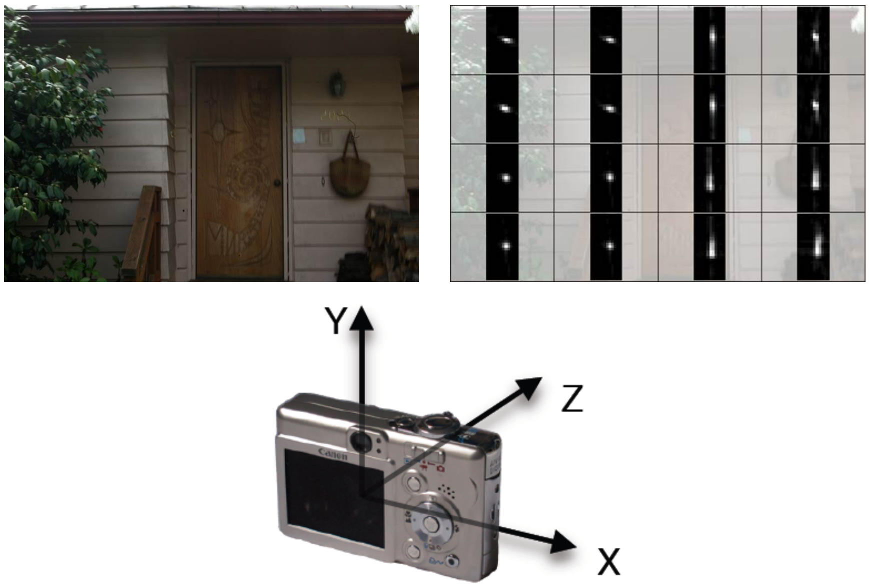


Figure 5.2: Modelling of camera shakes.

Bibliography

[1] E. S. Angel and A. K. Jain. Restoration of images degraded by spatially varying pointspread functions by a conjugate gradient method. *Appl. Opt.*, 17(14):2186–2190, Jul 1978.

[2] J. B. Baillon and G. Haddad. Quelques propriétés des opérateurs angle-bornés etn-cycliquement monotones. *Israel Journal of Mathematics*, 26(2):137–150, Jun 1977.

[3] J. Bardsley, S. Jefferies, J. Nagy, and R. Plemmons. A computational method for the restoration of images with an unknown, spatially-varying blur. *Optics express*, 14:1767–82, 04 2006.

[4] H. H. Bauschke and P. L. Combettes. The Baillon-Haddad Theorem Revisited. *Journal of Convex Analysis*, 17:781–787, 2010.

[5] F. Bociort and J. Kross. Seidel aberration coefficients for radial gradient-index lenses. *J. Opt. Soc. Am. A*, 11(10):2647–2656, Oct 1994.

[6] J. B. Breckinridge and D. G. Voelz. *Computational Fourier Optics: A MATLAB Tutorial*. SPIE Press monograph. SPIE Press, 2011.

[7] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, June 2005.

[8] P. L. Combettes. Fejér monotonicity in convex optimization. *Encyclopedia of Optimization*, 2:311–330, 2001.

[9] P. L. Combettes. Solving monotone inclusions via compositions of non-expansive averaged operators. *Optimization*, 53:475–504, 2004.

[10] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[11] G. Conforti. Zernike aberration coefficients from seidel and higher-order power-series coefficients. *Opt. Lett.*, 8(7):407–408, Jul 1983.

[12] P. Escande and P. Weiss. Numerical Computation of Spatially Varying Blur Operators A Review of Existing Approaches with a New One. working paper or preprint, April 2014.

[13] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2009.

[14] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[15] J. W. Goodman. *Introduction to Fourier Optics.* McGraw-Hill Series in Electrical and Computer Engineering: Communications and Signal Processing. McGraw-Hill, 1996.

[16] C. W. Groetsch. Integral equations of the first kind, inverse problems and regularization: a crash course. 2007.

[17] F. Heide, M. Rouf, M. Hullin, B. Labitzke, W. Heidrich, and A. Kolb. High-quality computational imaging through simple lenses. *ACM Trans. Graph.*, 32:149:1–149:14, 09 2013.

[18] J. Liang, J. Fadili, and G. Peyré. Convergence rates with inexact non-expansive operators. *Mathematical Programming*, 159(1):403–434, Sep 2016.

[19] A. Mikš. Modification of the formulas for third-order aberration coefficients. *J. Opt. Soc. Am. A*, 19(9):1867–1871, Sep 2002.

[20] C. Molinari, J. Liang, and J. Fadili. Convergence rates of forward–douglas–rachford splitting method. *Journal of Optimization Theory and Applications*, Apr 2019.

[21] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[22] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268, 1992.

[23] J. D. Simpkins and R. L. Stevenson. A spatially varying psf model for seidel aberrations and defocus. In *Visual Information Processing and Communication*, 2013.

[24] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. 1963.

[25] R. K. Tyson. Conversion of zernike aberration coefficients to seidel and higher-order power-series aberration coefficients. *Opt. Lett.*, 7(6):262–264, Jun 1982.

[26] T. Yue, J. Suo, J. Wang, , and Q. Dai. Blind optical aberration correction by exploring geometric and visual priors. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1684–1692, June 2015.

[27] C. Zalinescu. *Convex analysis in general vector spaces.* World Scientific, 2002.

[28] Y. Zhang, I. Minema, and T. Ueda. Restoration of radially blurred image created by spherical single lens system of cell phone camera. *Proceedings of IEEE Sensors*, pages 1333–1337, 11 2010.

[29] X. Zhu, S. Cohen, S. Schiller, and P. Milanfar. Estimating spatially varying defocus blur from a single image. *IEEE Transactions on Image Processing*, 22(12):4879–4891, Dec 2013.

국문초록

이 연구는 단색 수차 보정 문제를 풀기 위한 효율적이고 효과적인 방법들을 소개한다. 제안된 방법들은 Forward-Backward proximal splitting 방법에 기반한 것으로 이 방법은 최적화 문제를 경사하강법과 노이즈 제거의 두 문제로 나누어 반복 방법을 통해 푼다. 단색 수차 문제에 있어서 경사하강법은 큰 계산 비용을 요구하기 때문에 수차 연산자의 저비용 구현 방법을 개발한다. 이어서 6가지의 서로 다른 정칙 연산자에 기반한 노이즈 제거 방법을 적용한 영상 복원 방법을 제안한다. 이 연구에서는 제안된 영상 복원 방법들에 대한 실험을 수행한다. 실험에서는 점확산함수 (Point Spread Function)을 이용해 합성된 수차 영상을 이용하는데, 해당 점확산함수는 현대 디지털 카메라의 단색 수차 효과를 모방한 것이다.

주요어휘: 영상 처리, 단색 수차, 수차 보정
학번: 2012-23030