Wavelet Deblurring Algorithms for Spatially Varying Blur from High-Resolution Image Reconstruction

Raymond H. Chan*, Tony F. Chan†, Lixin Shen‡, Zuowei Shen‡

May 19, 2003

Abstract

High-resolution image reconstruction refers to reconstructing a higher resolution image from multiple low-resolution samples of a true image. In [2], we considered the case where there are no displacement errors in the low-resolution samples, i.e. the samples are aligned properly, and hence the blurring operator is spatially invariant. In this paper, we consider the case where there are displacement errors in the low-resolution samples. The resulting blurring operator is spatially varying and is formed by sampling and summing different spatially invariant blurring operators. We represent each of these spatially invariant blurring operators by a tensor product of a lowpass filter which associates the corresponding blurring operator with a multiresolution analysis of $L^2(\mathbb{R}^2)$. Using these filters and their duals, we derive an iterative algorithm to solve the problem based on the algorithmic framework of [2]. Our algorithm requires a nontrivial modification to the algorithms in [2], which apply only to spatially invariant blurring operators. Our numerical examples show that our algorithm gives higher peak signal-to-noise ratios and lower relative errors than those from the Tikhonov least squares approach.

1 Introduction

In [2], we introduced wavelet algorithms for solving general deconvolution problems and applied them to *high-resolution image reconstruction problems* where higher resolution images are reconstructed from multiple low-resolution samples of the true images with the low-resolution sensors aligned properly. The blurring operator thus formed is spatially invariant and can be represented by a tensor product of a lowpass filter that generates a multiresolution analysis of $L^2(\mathbb{R}^2)$. The low-resolution samples are viewed as the high resolution image passed through the blurring operator. Since the blurring operator is spatially invariant, the reconstruction is essentially a deconvolution problem.

This paper considers the high-resolution image reconstruction from low-resolution sensors that have subpixel displacement errors, i.e. the sensors are not aligned properly. The resulting blurring operator is spatially varying and is formed by sampling and summing different spatially invariant blurring operators. Previous work in [1, 9] reduces the problem to a system of linear equation and solves it by the preconditioned conjugate gradient method.

*Department of Mathematics, the Chinese University of Hong Kong, Shatin, NT, P. R. China. Research supported in part by HKRGC Grant CUHK4212/99P and CUHK Grant DAG 2060183.

†Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90024-1555, USA. Research supported in part by the ONR under contract N00014-96-1-0277 and by the NSF under grant DMS-9973341

‡Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543. Research supported in part by the Center of Wavelets, Approximation and Information Processing (CWAIP) funded by the National Science and Technology Board, the Ministry of Education under Grant RP960 601/A.

Here, we represent the different spatially invariant blurring operators by tensor products of different lowpass filters. To take advantage of the ideas developed in [2], we first design a dual filter for each lowpass filter associated with the corresponding blurring operator. Using the simple structure of these filters, we then modify the algorithms in [2] to obtain an algorithm for the spatially varying case. We note that although the algorithmic framework laid out in [2] still applies, the modification is nontrivial since the problem itself is no longer a simple deconvolution problem. Numerical experiments indicate that our algorithm gives higher peak signal-to-noise ratios and lower relative errors than those of the Tikhonov least squares method.

The outline of the paper is as follows. In §2, we recall the mathematical model of the high-resolution image reconstruction problem. In §3, filters are designed and our wavelet algorithm is presented. Numerical experiments follow in §4.

2 The Mathematical Model

Here we give a brief introduction to the mathematical model of the high-resolution image reconstruction. Details can be found in [1, 2]. Let the intensity function of an underlying continuous image be $f(x_1, x_2)$. Our model assumes that an image at a given resolution is obtained by means of averaging f over the pixels which have size corresponding to that resolution. We note that higher the resolution, smaller in size are the pixels. Our mathematical problem is: given several averages of f at a low resolution, how can we deduce a good approximation to an average of f at a higher resolution? In what follows, we will make these notions more precise.

Suppose the image of a given scene can be obtained from sensors with $N_1 \times N_2$ pixels. Let the actual length and width of each pixel be T_1 and T_2 respectively. We will call these sensors *low-resolution sensors*. The scene we are interested in, i.e. the *region of interest*, can be described as:

$$\mathbf{S} = \left\{(x_1, x_2) \in \mathbb{R}^2 \mid 0 \le x_1 \le T_1 N_1, 0 \le x_2 \le T_2 N_2\right\}.$$

Our aim is to construct a higher resolution image of $\mathbf{S}$ by using an array of $K_1 \times K_2$ low-resolution sensors, i.e. we want an image of $\mathbf{S}$ with $M_1 \times M_2$ pixels, where $M_1 = K_1 N_1$ and $M_2 = K_2 N_2$. Thus the length and width of each of these *high-resolution pixels* will be T_1/K_1 and T_2/K_2 respectively. To maintain the aspect ratio of the reconstructed image, we consider only $K_1 = K_2 = K$.

Let $f(x_1, x_2)$ be the intensity of the scene at any point (x_1, x_2) in $\mathbf{S}$. By reconstructing the high-resolution image, we mean to find or approximate the values

$$\frac{K^2}{T_1 T_2} \int_{iT_1/K}^{(i+1)T_1/K} \int_{jT_2/K}^{(j+1)T_2/K} f(x_1, x_2) dx_1 dx_2, \quad 0 \le i < M_1, 0 \le j < M_2,$$

which is the average intensity of all the points inside the (i, j)th high-resolution pixel:

$$\left[i\frac{T_1}{K}, (i+1)\frac{T_1}{K}\right] \times \left[j\frac{T_2}{K}, (j+1)\frac{T_2}{K}\right], \quad 0 \le i < M_1, 0 \le j < M_2. \tag{1}$$

In order to have enough information to resolve the high-resolution image, there are subpixel displacements between the sensors in the sensor arrays. Ideally, the sensors should be shifted from each other by a value proportional to the length and the width of the high-resolution pixels. However, in practice there can be small perturbations around these ideal subpixel locations due to imperfection of the mechanical imaging system. Thus, for sensor (k_1, k_2), $0 \le k_1, k_2 < K$ with $(k_1, k_2) \ne (0, 0)$, its horizontal and vertical displacements $d^x_{k_1 k_2}$ and $d^y_{k_1 k_2}$ with respect to the $(0, 0)$ reference sensor are given by

$$d^x_{k_1 k_2} = \left(k_1 + \epsilon^x_{k_1, k_2}\right) \frac{T_1}{K} \quad \text{and} \quad d^y_{k_1 k_2} = \left(k_2 + \epsilon^y_{k_1, k_2}\right) \frac{T_2}{K}.$$

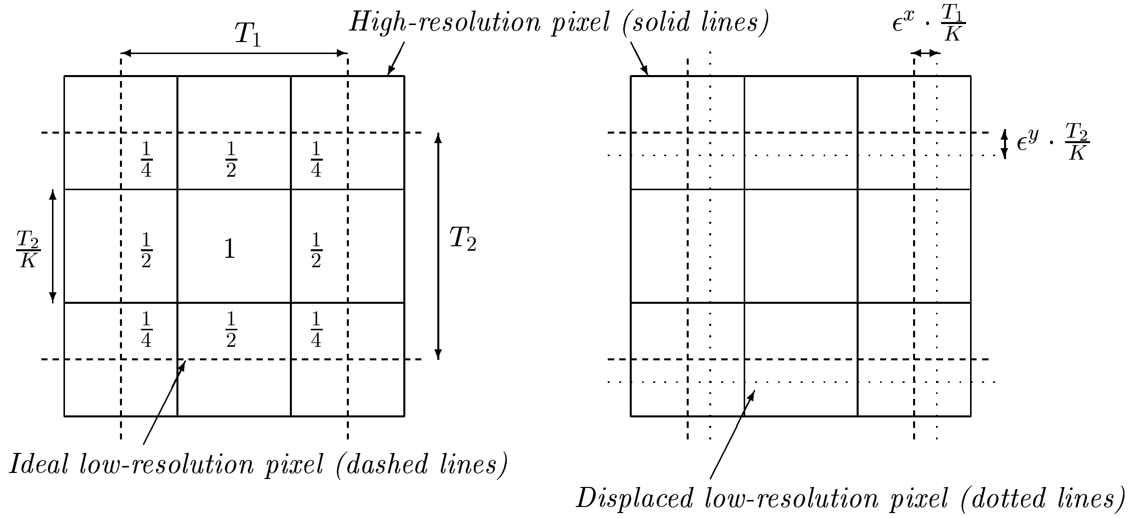

Figure 1: Sensors without and with displacement error when $K = 2$ (left and right respectively).

Here $\epsilon^x_{k_1,k_2}$ and $\epsilon^y_{k_1,k_2}$ are the horizontal and vertical *displacement errors* respectively. They can be obtained by the manufacturers during camera calibration. Figure 1 shows the case when we have a 2×2 sensor array. We assume that

$$|\epsilon^x_{k_1,k_2}| < \frac{1}{2} \quad \text{and} \quad |\epsilon^y_{k_1,k_2}| < \frac{1}{2}. \tag{2}$$

For if not, the low resolution images from two different sensor arrays will be overlapped so much that the reconstruction of the high resolution image is rendered impossible. For example, in Figure 1, if $\epsilon^x > 1/2$, then the three high-resolution pixels on the left hand side are not covered by the lower-resolution pixel at all whereas the three high-resolution pixels on the right hand side are covered twice by two adjacent lower-resolution pixels.

For sensor (k_1, k_2), the average intensity registered at its (n_1, n_2)th pixel is modeled by:

$$g_{k_1k_2}[n_1, n_2] = \frac{1}{T_1T_2} \int_{T_1(n_1-1/2)+d^x_{k_1k_2}}^{T_1(n_1+1/2)+d^x_{k_1k_2}} \int_{T_2(n_2-1/2)+d^y_{k_1k_2}}^{T_2(n_2+1/2)+d^y_{k_1k_2}} f(x_1, x_2)dx_1dx_2 + \eta_{k_1k_2}[n_1, n_2]. \tag{3}$$

Here $0 \le n_1 < N_1$ and $0 \le n_2 < N_2$ and $\eta_{k_1k_2}[n_1, n_2]$ is the noise, see [1]. Examples of low-resolution images are given in Figures 3(a) and 4(a). We intersperse all the low-resolution images $g_{k_1k_2}$ to form an $M_1 \times M_2$ image g by assigning

$$g[Kn_1 + k_1, Kn_2 + k_2] = g_{k_1k_2}[n_1, n_2]. \tag{4}$$

The image g is called the *observed high-resolution image*. It is already a better image than any one of the low-resolution samples g_{k_1,k_2} themselves, see Figures 3(b) and 4(b).

To obtain an even better image than g, one will have to solve (3) for f. According to [1], we solve it by first discretizing (3) using the rectangular quadrature rule, which is an approximation to the physics of the CCD arrays. Equivalently, we assume that for each (i, j)th high-resolution pixel given in (1), the intensity f is constant and is equal to $f[i, j]$ for every point in that pixel. Then carrying out the integration in (3), and using the re-ordering (4), we obtain a system of linear equations relating the unknown values $f[i, j]$ to the given low-resolution pixel values $g[i, j]$. This linear system, however, is not square. This is because the evaluation of $g_{k_1k_2}[n_1, n_2]$ in (3) involves points outside

$\mathbf{S}$. For example, $g_{0,0}[0,0]$ in (3) requires the value of $f[-1,-1]$. Thus we have more unknowns than given values, and the system is underdetermined.

To resolve this, one can impose boundary conditions on f for points outside $\mathbf{S}$. A standard way is to assume that f is periodic outside:

$$f(x+iT_1N_1, y+jT_2N_2) = f(x,y), \qquad i,j \in \mathbb{Z}, \tag{5}$$

see for instance [7, §5.1.3]. Using (5) and ordering the discretized values of f and g in a column-by-column fashion, the blurring matrix corresponding to the (k_1,k_2)th sensor can be written as

$$L(\epsilon^x_{k_1}, \epsilon^y_{k_2}) = L(\epsilon^x_{k_1,k_2}) \otimes L(\epsilon^y_{k_1,k_2}) \tag{6}$$

where $\otimes$ is the Kronecker tensor product and $L(\epsilon^x_{k_1,k_2})$ is an $M_1 \times M_1$ circulant matrix with the middle row given by

$$\frac{1}{K}[0,0,\cdots,0,\frac{1}{2}+\epsilon^x_{k_1,k_2},\underbrace{1,\ldots,1}_{K-1},\frac{1}{2}-\epsilon^x_{k_1,k_2},0,\cdots,0]. \tag{7}$$

Since we are using the rectangular rule in (3), the entries in (7) are just the area of the high-resolution pixels which fall inside the low-resolution pixel under consideration, cf. Figure 1. The $M_2 \times M_2$ blurring matrix $L(\epsilon^y_{k_1,k_2})$ is defined similarly. We note that there are other boundary conditions that one can impose on the image, see for instance [1, 10]. In this paper, we will only consider the periodic boundary condition.

The blurring matrix for the whole sensor array is made up of matrices from each sensor:

$$L(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y) = \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1} D_{k_1,k_2} L(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2}) = \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1} D_{k_1,k_2}[L(\epsilon^x_{k_1,k_2}) \otimes L(\epsilon^y_{k_1,k_2})]. \tag{8}$$

Here both $\boldsymbol{\epsilon}^x$ and $\boldsymbol{\epsilon}^y$ are $K \times K$ matrices, and D_{k_1,k_2} are the sampling matrices, which are diagonal matrices with diagonal elements equal to 1 if the corresponding component of g comes from the (k_1,k_2)th sensor and zero otherwise, see (4) or [1] for more details. Because of the sampling matrices, $L(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y)$ is spatially variant and has no tensor structure or Toeplitz structure. Furthermore, since (3) is an averaging process, it is ill-conditioned and susceptible to noise. To remedy this, we can employ the Tikhonov regularization which solves the system

$$(L(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y)^* L(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y) + \beta R)\mathbf{f} = L(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y)^* \mathbf{g} \tag{9}$$

for $\mathbf{f}$. Here $\mathbf{f}$ and $\mathbf{g}$ are the column vectors formed by f and g respectively, R is a regularization operator (usually chosen to be the identity operator or some differential operators) and β is the regularization parameter, see [7, §5.3].

The normal equation (9) is derived from the least squares approach. In the next section, we will derive an algorithm by using the wavelet approach.

3 Filter Design and the Algorithm

Since (3) is an averaging process, the blurring matrix $L(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2})$ corresponding to the (k_1,k_2)th sensor can be considered as a lowpass filter acting on the image f. From (6) and (7), this lowpass filter is a tensor product of the univariate refinement masks

$$\frac{1}{K}(\frac{1}{2}+\epsilon,\underbrace{1,\ldots,1}_{K-1},\frac{1}{2}-\epsilon), \tag{10}$$

where the parameters ϵ are different in the x and y directions for each sensor.

For simplicity, we consider $K = 2$ in this section. The general case can be analyzed similarly. Recall that a function ϕ in $L^2(\mathbb{R})$ is refinable if it satisfies

$$\phi = 2 \sum_{\alpha \in \mathbb{Z}} m(\alpha)\phi(2 \cdot -\alpha).$$

The sequence m is called a refinement mask, or lowpass filter. The symbol of the sequence m is defined as $\widehat{m}(\alpha) = \sum_{\alpha \in \mathbb{Z}} m(\alpha) e^{-i\alpha\omega}$. The function ϕ is stable if its shifts form a Riesz system, i.e., there exist constants $0 < c \leq C < \infty$, such that for any sequence $q(\alpha) \in \ell^2(\mathbb{Z})$,

$$c\|q\|_2 \leq \left\| \sum_{\alpha \in \mathbb{Z}} q(\alpha)\phi(\cdot - \alpha) \right\|_2 \leq C\|q\|_2.$$

Stable functions ϕ and ϕ^d are called a *dual pair* when they satisfy

$$\langle \phi, \phi^d(\cdot - \alpha) \rangle = \begin{cases} 1, & \alpha = 0; \\ 0, & \alpha \in \mathbb{Z} \setminus \{0\}. \end{cases}$$

We will denote the refinement mask of ϕ^d by m^d.

For a given compactly supported refinable stable function $\phi \in L^2(\mathbb{R})$, define $S(\phi) \subset L^2(\mathbb{R})$ to be the smallest closed shift invariant subspace generated by ϕ and define $S^k(\phi) := \{u(2^k \cdot) : u \in S(\phi)\}$, $k \in \mathbb{Z}$. Then the sequence $S^k(\phi)$, $k \in \mathbb{Z}$, forms a multiresolution of $L_2(\mathbb{R}^2)$. Here we recall that a sequence $S^k(\phi)$ forms a *multiresolution* when the following conditions are satisfied: (i) $S^k(\phi) \subset S^{k+1}(\phi)$; (ii) $\cup_{k \in \mathbb{Z}} S^k(\phi) = L^2(\mathbb{R})$ and $\cap_{k \in \mathbb{Z}} S^k(\phi) = \{0\}$; (iii) ϕ and its shifts form a Riesz basis of $S(\phi)$, see [4]. The sequence $S^k(\phi^d)$, $k \in \mathbb{Z}$, also forms a multiresolution of $L^2(\mathbb{R})$.

The biorthonormal wavelets ψ and ψ^d are defined by

$$\psi := 2 \sum_{\alpha \in \mathbb{Z}} r(\alpha)\phi(2 \cdot -\alpha), \quad \text{and} \quad \psi^d := 2 \sum_{\alpha \in \mathbb{Z}} r^d(\alpha)\phi^d(2 \cdot -\alpha),$$

where $r(\alpha) := (-1)^\alpha m^d(1-\alpha)$, and $r^d(\alpha) := (-1)^\alpha m(1-\alpha)$ are the wavelet masks, see for example [4] for details. From the wavelet theory (see e.g. [3]), the refinement masks m, m^d and the wavelet masks r, r^d satisfy the perfect reconstruction equation:

$$\overline{\widehat{m^d}}\widehat{m} + \overline{\widehat{r^d}}\widehat{r} = 1. \tag{11}$$

The existence of a biorthogonal wavelet pair for a given refinement mask is the basis of our analysis in [2]. In the next subsection, we therefore first construct wavelet masks corresponding to the lowpass filters in (10).

3.1 Filter design

Our wavelet algorithm depends on the existence of wavelet masks corresponding to the lowpass filters of the low-resolution sensors. When there are no displacement errors, the lowpass filters are the tensor product refinement masks of $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ for the 2×2 sensor array, and $(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8})$ for the 4×4 sensor array, cf. (7) with $\epsilon^x_{k_1,k_2} = 0$. Thus the filters from different sensors are the same and so are the corresponding wavelet masks.

When there are displacement errors, the lowpass filters of the sensors are perturbations of the above filters. They are tensor products of the filter in (10) and are different for different sensors.

However, we can still identify their refinement masks, their corresponding dual refinement masks and the wavelet masks. We note that since the blurring matrix of the whole sensor array is made up by adding the individual blurring matrix from each sensor (see (8)), there does not exist a tensor product bivariate filter corresponding to the whole sensor array.

As examples, we give below the refinement masks and wavelet masks for each sensor for $K = 2$ and $K = 4$. Again, for simplicity, we give only the univariate masks. The actual masks for each sensor are obtained by taking the tensor product.

Example 1. *For $K = 2$, the corresponding mask (10) is*

$$m(-1) = \frac{1}{2}(\frac{1}{2} + \epsilon),\ m(0) = \frac{1}{2},\ m(1) = \frac{1}{2}(\frac{1}{2} - \epsilon),$$

and $m(\alpha) = 0$ for all other α. It has many dual masks. The nonzero terms of one of its dual masks are

$$m^d(-2) = -\frac{1}{8} + \frac{\epsilon}{4},\ m^d(-2) = \frac{1}{4},\ m^d(0) = \frac{3}{4},\ m^d(1) = \frac{1}{4},\ m^d(2) = -\frac{1}{8} - \frac{\epsilon}{4}.$$

The dual pair of the wavelet masks are

$$r(\alpha) := (-1)^\alpha m^d(1-\alpha), \quad \text{and} \quad r^d(\alpha) := (-1)^\alpha m(1-\alpha).$$

It can be shown, by applying Theorem 3.14 in [11] (also see [8]), that if $|\epsilon| < \frac{1}{2}$ (i.e. (2) holds), then m and m^d are the refinement masks of a dual pair of stable functions ϕ and ϕ^d with dilation 2.

When $\epsilon = 0$, Example 1 is the well-known biorthogonal linear spline filter (see [4, p.277]).

Example 2. *For $K = 4$, the corresponding mask (10) is*

$$m(\alpha) = \frac{1}{4}(\frac{1}{2} + \epsilon), \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}(\frac{1}{2} - \epsilon), \quad \alpha = -2, \dots, 2,$$

with $m(\alpha) = 0$ for all other α. The nonzero terms of a dual refinement mask of m is

$$m^d(\alpha) = -\frac{1}{16} + \frac{\epsilon}{8}, \frac{1}{8}, \frac{5}{16} + \frac{\epsilon}{8}, \frac{1}{4}, \frac{5}{16} - \frac{\epsilon}{8}, \frac{1}{8}, -\frac{1}{16} - \frac{\epsilon}{8}, \quad \alpha = -3, \dots, 3.$$

The nonzero terms of the corresponding wavelet masks are

$$r_1(\alpha) = -\frac{1}{8} - \frac{\epsilon}{4}, -\frac{1}{4}, \frac{\epsilon}{2}, \frac{1}{4}, \frac{1}{8} - \frac{\epsilon}{4}, \quad \alpha = -2, \dots, 2,$$

$$r_2(\alpha) = -\frac{1}{16} - \frac{\epsilon}{8}, -\frac{1}{8}, \frac{5}{16} - \frac{\epsilon}{8}, -\frac{1}{4}, \frac{5}{16} + \frac{\epsilon}{8}, -\frac{1}{8}, -\frac{1}{16} + \frac{\epsilon}{8}, \quad \alpha = -2, \dots, 4,$$

$$r_3(\alpha) = \frac{1}{16} + \frac{\epsilon}{8}, \frac{1}{8}, -\frac{7}{16} - \frac{\epsilon}{8}, 0, \frac{7}{16} - \frac{\epsilon}{8}, -\frac{1}{8}, -\frac{1}{16} + \frac{\epsilon}{8} \quad \alpha = -2, \dots, 4.$$

The dual highpass filters are

$$r_1^d(\alpha) = (-1)^{1-\alpha} r_3(1-\alpha),\ r_2^d(\alpha) = (-1)^{1-\alpha} m(1-\alpha),\ r_3^d(\alpha) = (-1)^{1-\alpha} r_1(1-\alpha),$$

for appropriate α. Again, if $|\epsilon| < \frac{1}{2}$, then m and m^d are the refinement masks of a dual pair of the stable functions ϕ and ϕ^d with dilation 4.

3.2 The Algorithm

In this subsection, we present our algorithm. For simplicity, we let $K = 2$. Since the blurring matrix from each sensor is a tensor product (see (8)), it suffices to consider the one-dimensional case, i.e. the 1×2 sensor array. The general case of $K \times K$ sensors can be derived similarly by taking the tensor products. For simplicity, we denote the number of low-resolution pixels by N and the number of high-resolution pixels by $M(= 2N)$.

For the 1×2 sensor array, the blurring matrix for the whole sensor array is given by

$$L(\boldsymbol{\epsilon}) = D_1 L(\epsilon_0) + D_2 L(\epsilon_1).$$

Here D_ℓ are the sampling matrices (by factor 2), i.e. $D_\ell = I_N \otimes \text{diag}(\mathbf{e}_\ell)$ where $\mathbf{e}_\ell$ denotes the ℓth column of the 2×2 identity matrix and $\boldsymbol{\epsilon} = (\epsilon_0, \epsilon_1)$. The $M \times M$ matrices $L(\epsilon_k)$, $k = 0, 1$ are defined in (12) below. They are the blurring matrices corresponding to the kth sensor with displacement error ϵ_k. In matrix forms, all the matrices corresponding to sensor k are the circulant matrices generated by the corresponding masks. They are:

$$\begin{aligned}
L^d(\epsilon_k) &= \text{circulant}(\frac{3}{4}, \frac{1}{4}, -\frac{1}{8} + \frac{\epsilon_k}{4}, 0, \cdots, 0, -\frac{1}{8} - \frac{\epsilon_k}{4}, \frac{1}{4}); \\
L(\epsilon_k) &= \text{circulant}(\frac{1}{2}, \frac{1}{2}(\frac{1}{2} - \epsilon_k), 0, \cdots, 0, \frac{1}{2}(\frac{1}{2} + \epsilon_k)); \\
H^d(\epsilon_k) &= \text{circulant}(\frac{1}{4} - \frac{\epsilon_k}{2}, 0, \cdots, 0, \frac{1}{4} + \frac{\epsilon_k}{2}, -\frac{1}{2}); \\
H(\epsilon_k) &= \text{circulant}(\frac{1}{4}, -\frac{3}{4}, \frac{1}{4}, \frac{1}{8} - \frac{\epsilon_k}{4}, 0, \cdots, 0, \frac{1}{8} + \frac{\epsilon_k}{4}).
\end{aligned} \tag{12}$$

Here $\text{circulant}(c_1, \cdots, c_M)$ denotes the $M \times M$ circulant matrix with $(c_1, \cdots, c_M)$ as the first row.

For each sensor k (k is 0 or 1), the matrices $L^d(\epsilon_k)$, $L(\epsilon_k)$, $H^d(\epsilon_k)$, and $H(\epsilon_k)$, satisfy

$$L^d(\epsilon_k) L(\epsilon_k) + H^d(\epsilon_k) H(\epsilon_k) = I, \tag{13}$$

because of (11). Our iterative algorithm starts from this identity. Suppose that at Step n, we have the nth approximation $\mathbf{f}_n$. Then (13) gives

$$L^d(\epsilon_k) L(\epsilon_k) \mathbf{f}_n + H^d(\epsilon_k) H(\epsilon_k) \mathbf{f}_n = \mathbf{f}_n.$$

Assume that $L(\epsilon_k)\mathbf{f}$ is available, then we replace $L(\epsilon_k)\mathbf{f}_n$ by $L(\epsilon_k)\mathbf{f}$ to improve the approximation. By this, we define

$$\mathbf{f}_{n+1} = L^d(\epsilon_k) L(\epsilon_k) \mathbf{f} + H^d(\epsilon_k) H(\epsilon_k) \mathbf{f}_n. \tag{14}$$

For the case with no displacement error, i.e. $\epsilon_1 = \epsilon_2 = 0$, we have $L(\epsilon_1) = L(\epsilon_2) = L(0)$. We have analyzed (14) in [2] through multiresolution analysis by using the lowpass and highpass filters that generate the matrices $L(0)$, $L^d(0)$, $H(0)$ and $H^d(0)$. We showed that the blurred image can be represented by a function in the low resolution space, the reconstructed image is in a high resolution space, and $H\mathbf{f}_n$ is the high frequency component of $\mathbf{f}_n$ which can be represented by a function in the wavelet space. At each iterate, the term $L(0)\mathbf{f}$ is always chosen to be $\mathbf{g}$ which is the low frequency content of the original image and is given by the observed image. The high frequency content of the original image is updated by the high frequency content of the previous iterate. It was further shown in [2] that the sequence of functions corresponding to the high resolution images at each iteration converges to the function corresponding to the original image $\mathbf{f}$ in L^2-norm. When

$\mathbf{g}$ contains noise, then $\mathbf{f}_n$ has noise brought in from the previous iteration. To build a denoising procedure into the algorithm, we further decompose the high frequency component $H(0)\mathbf{f}_n$ via the standard wavelet decomposition algorithm. This gives a wavelet packet decomposition of $\mathbf{f}_n$. Then, applying a wavelet thresholding denoising algorithm to this decomposition and reconstructing $H(0)\mathbf{f}$ back via the standard reconstruction algorithm leads to a denoising procedure for $\mathbf{f}_n$. The details of this algorithm and its analysis can be found in [2].

For the case with displacement errors, the blurred image $\mathbf{g}$ has error from the displacement and the matrices differ from one sensor to the other. To implement (14), we need to approximate $L(\epsilon_k)\mathbf{f}$, the first term on the right-hand side of (14). (As we have seen in the previous paragraph, for the case with no displacement error, $L(0)\mathbf{f}$ is simply $\mathbf{g}$). For the case with displacement error, we may simply ignore the different matrices used at the two sensors and fix on only one set of matrices, say $L^d(\epsilon_0)$, $L(\epsilon_0)$, $H^d(\epsilon_0)$ and $H(\epsilon_0)$. Then we apply (14) with $L(\epsilon)\mathbf{f} = \mathbf{g}$ as the (approximation of) the observed image. This gives an algorithm close to Algorithm 3 of [2] and it converges independent of the choice of ϵ. But doing this will ignore the displacement errors between the sensors.

In order to take into the consideration the displacement errors and use our algorithm (14), we modify it by updating the approximation of $L(\epsilon_k)\mathbf{f}$ through exploring the available information at each iterate. More precisely, we divide the $(n+1)$th iteration into the following two steps:

- Choose $\mathbf{g}_{n+\frac{1}{2}} = D_1\mathbf{g} + D_2L(\epsilon_0)\mathbf{f}_n$ and define

$$\mathbf{f}_{n+\frac{1}{2}} = L^d(\epsilon_0)\mathbf{g}_{n+\frac{1}{2}} + H^d(\epsilon_0)H(\epsilon_0)\mathbf{f}_n.$$

- Choose $\mathbf{g}_{n+1} = D_1L(\epsilon_1)\mathbf{f}_{n+\frac{1}{2}} + D_2\mathbf{g}$ and define

$$\mathbf{f}_{n+1} = L^d(\epsilon_1)\mathbf{g}_{n+1} + H^d(\epsilon_1)H(\epsilon_1)\mathbf{f}_{n+\frac{1}{2}}.$$

Since $L(\epsilon_0)\mathbf{f} = D_1L(\epsilon_0)\mathbf{f} + D_2L(\epsilon_0)\mathbf{f}$ and $D_1\mathbf{g} = D_1(D_1L(\epsilon_0) + D_2L(\epsilon_1))\mathbf{f} = D_1L(\epsilon_0)\mathbf{f}$, we only need to approximate $D_2L(\epsilon_0)\mathbf{f}$ in order to get an approximation of $L(\epsilon_0)\mathbf{f}$. Thus, in the first step of $(n+1)$th iteration, we use $D_2L(\epsilon_0)\mathbf{f}_n$ to approximate $D_2L(\epsilon_0)\mathbf{f}$. Similarly, in the second step of $(n+1)$th iteration, we use $D_1L(\epsilon_1)\mathbf{f}_{n+\frac{1}{2}}$ to approximate $D_1L(\epsilon_1)\mathbf{f}$.

When $\mathbf{g}$ contains noise, the wavelet thresholding algorithm can be built in naturally again as in [2]. To do it, we first introduce a truncation operator:

$$\mathcal{D}_\lambda((x_1, \ldots, x_l, \ldots)^T) \equiv (x_1\chi_{|x_1|>\lambda}, \ldots, x_l\chi_{|x_l|>\lambda}, \ldots)^T.$$

Here $\chi_{|x|>\lambda}$ equals to 1 if $|x| > \lambda$, and 0 otherwise. Then for any given $L(\epsilon)$, $L^d(\epsilon)$, $H(\epsilon)$ and $H^d(\epsilon)$ satisfying (13), and a data vector $\mathbf{v}$ with noise, we define the thresholding operator:

$$\mathcal{T}_{J,\epsilon}(\mathbf{v}) \equiv (L^d(\epsilon))^J(L(\epsilon))^J\mathbf{v} + \sum_{j=0}^{J-1}(L^d(\epsilon))^jH^d(\epsilon)\mathcal{D}_\lambda\left(H(\epsilon)L^j(\epsilon)\mathbf{v}\right), \quad J = 1, 2, \ldots.$$

The thresholding operator $\mathcal{T}_{J,\epsilon}$ consists of three steps. The first step is a translation invariant wavelet transformation with $L(\epsilon)$ and $H(\epsilon)$. Let v be a function at certain level of multiresolution analysis representing the data $\mathbf{v}$. The operator $\mathcal{T}_{J,\epsilon}$ transforms the data $\mathbf{v}$ into $(L(\epsilon))^J\mathbf{v}$, which is the coefficients of the representation of a coarse approximation of v at J-th level down and contains mainly low frequency content of $\mathbf{v}$; and $H(\epsilon)L^j(\epsilon)\mathbf{v}$, $j = 0, \ldots, J-1$, the detailed parts of $\mathbf{v}$ at level j that are the wavelet coefficients of v and contain high frequency contents of $\mathbf{v}$.

The second step in $\mathcal{T}_{J,\epsilon}$ is noise removal by thresholding. To guarantee that the thresholded $\mathcal{D}_\lambda\left(H(\epsilon)L^j(\epsilon)\mathbf{v}\right)$ keeps the original information of $\mathbf{v}$, a proper λ must be selected. Here we choose $\lambda = \sigma\sqrt{2\log(M)}$ which was shown to be an optimal threshold from a number of perspectives [5, 6], and σ is the variance of $\mathbf{v}$ estimated numerically by the method in [5].

The third step is the inverse transformation of the translation invariant wavelet transformation with $L^d(\epsilon)$ and $H^d(\epsilon)$. The thresholding step enables us to discriminate the information between signal and noise, and therefore obtain a good approximation of v with less noise from the original data $\mathbf{v}$ after applying the third step.

Our algorithm is now given as follows.

Wavelet Algorithm:

(i) Choose an initial approximation $\mathbf{f}_0$ *(e.g.* $\mathbf{f}_0 = \mathbf{g}$*);*

(ii) Iterate on n *until convergence;*

(1) take $\mathbf{g}_{n+\frac{1}{2}} = D_1\mathbf{g} + D_2 L(\epsilon_0)\mathbf{f}_n$ *and do*

$$\mathbf{f}_{n+\frac{1}{2}} = L^d(\epsilon_0)\mathbf{g}_{n+\frac{1}{2}} + H^d(\epsilon_0)\mathcal{T}_{J,\epsilon_0}\left(H(\epsilon_0)\mathbf{f}_n\right).$$

(2) take $\mathbf{g}_{n+1} = D_1 L(\epsilon_1)\mathbf{f}_{n+\frac{1}{2}} + D_2\mathbf{g}$ *and do*

$$\mathbf{f}_{n+1} = L^d(\epsilon_1)\mathbf{g}_{n+1} + H^d(\epsilon_1)\mathcal{T}_{J,\epsilon_1}\left(H(\epsilon_1)\mathbf{f}_{n+\frac{1}{2}}\right).$$

(3) increase n *to* $n+1$ *and go to Step (1).*

(iii) Let $\mathbf{f}_{n_0}$ *be the final iterate from Step (ii). The final solution of our Algorithm is*

$$\mathbf{f}_c = \mathcal{T}_{J,0}(\mathbf{f}_{n_0}).$$

The computational complexity of our algorithm depends on the number of iterations required for convergence. In each iteration, we essentially go through a J-level wavelet decomposition and reconstruction procedure K times, therefore it needs $O(M) = O(KN)$ operations. As for the value of J, the larger it is, the finer the wavelet packet decomposition of $\mathbf{f}_n$ and $\mathbf{f}_{n+\frac{1}{2}}$ will be before it is denoised. This leads to a better denoising scheme. However, a larger J will cost slightly more computational time. From our numerical tests, we find that it is already good enough to choose J to be either 1 or 2. The variances $\sigma_{n,k}$ are estimated by the method given in [5] which uses the median of the absolute value of the entries in the vector $H(\epsilon_k)\mathbf{f}_{n+\frac{k}{2}}$. Hence the cost of computing $\sigma_{n,k}$ is $O(M\log(M))$, see for instance [12]. Finally, the cost of Step (iii) is less than one additional iteration of Step (ii). As a comparison, each iteration of the preconditioned conjugate gradient method used in [9] would require the same amount of work, i.e. $O(M\log(M))$ operations. One nice feature of our algorithm is that it is parameter-free if we choose $\lambda = \sigma\sqrt{2\log(M)}$. We then do not have to choose the regularization parameter β as in the Tikhonov method (9).

As shown above, one of the key facts used in our algorithm is (13), the matrix form of the "perfect reconstruction" identity from the masks. The equation was derived under the periodic boundary assumption we imposed on the images. Since the masks, which are determined by the lowpass filters of the sensors, are not symmetric, one cannot obtain (13) if one imposes the symmetric boundary condition instead.

4 Numerical Experiments

In this section, we implement the wavelet algorithm developed in §3.2 and compare it with the Tikhonov least squares method (9). We evaluate the methods using the relative error (RE) and the peak signal-to-noise ratio (PSNR) which compare the reconstructed image $\mathbf{f}_c$ with the original image $\mathbf{f}$. They are defined by

$$\mathrm{RE} = \frac{\|\mathbf{f} - \mathbf{f}_c\|_2}{\|\mathbf{f}\|_2} \qquad \text{and} \qquad \mathrm{PSNR} = 10 \log_{10} \frac{255^2 N^2}{\|\mathbf{f} - \mathbf{f}_c\|_2^2},$$

where the size of the restored images is $N \times N$.

We use the "Boat" image of size 260×260 shown in Figure 2 as the original image in our numerical tests. To simulate the real world situations, the pixel values of the low-resolution images near the boundary are obtained from the discrete equation of (3) by using the actual pixel values of the "Boat" image. No periodic boundary conditions are imposed on these pixels. For the Tikhonov method (9), we will use the identity matrix I as the regularization operator R. The optimal regularization parameter β^* is chosen by trial and error so that they give the best PSNR values for the resulting equations. For our algorithm, we stop the iteration as soon as the values of PSNR peaked. We use $J = 1$ in our algorithm as it incurs the least cost and the result is already better than that of the Tikhonov method. In case that PSNR is not available, we stop the iteration, when the two consecutive iterants are less than a given tolerance.

Figure 2: The original "Boat" image.

4.1 2×2 Sensor Array

For 2×2 sensor arrays, the bivariate filter for the blurring process is the tensor product of the lowpass filter given in Example 1. By applying the matrix $L_2(\boldsymbol{\epsilon}^x, \boldsymbol{\epsilon}^y)$ of size 260×260 on the true "Boat" image and then adding white noise, the resulting image is then chopped to size 256×256 to form our observed high-resolution image $\mathbf{g}$. We note that the four 128×128 low-resolution frames can be obtained by downsampling $\mathbf{g}$ by a factor of 2 in both the horizontal and the vertical directions.

In what follows, all images are viewed as column vectors by reordering the entries of the images in a column-wise order. The blurring matrices and the wavelet matrices are formed by the tensor product, see (6). In particular, we have

$$
\begin{aligned}
L_{k_1,k_2}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= L(\epsilon^x_{k_1,k_2}) \otimes L(\epsilon^y_{k_1,k_2}),\\
H_{k_1,k_2,(0,1)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= L(\epsilon^x_{k_1,k_2}) \otimes H(\epsilon^y_{k_1,k_2}),\\
H_{k_1,k_2,(1,0)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= H(\epsilon^x_{k_1,k_2}) \otimes L(\epsilon^y_{k_1,k_2}),\\
H_{k_1,k_2,(1,1)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= H(\epsilon^x_{k_1,k_2}) \otimes H(\epsilon^y_{k_1,k_2}),\\
L^d_{k_1,k_2}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= L^d(\epsilon^x_{k_1,k_2}) \otimes L^d(\epsilon^y_{k_1,k_2}),\\
H^d_{k_1,k_2,(0,1)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= L^d(\epsilon^x_{k_1,k_2}) \otimes H^d(\epsilon^y_{k_1,k_2}),\\
H^d_{k_1,k_2,(1,0)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= H^d(\epsilon^x_{k_1,k_2}) \otimes L^d(\epsilon^y_{k_1,k_2}),\\
H^d_{k_1,k_2,(1,1)}(\epsilon^x_{k_1,k_2},\epsilon^y_{k_1,k_2}) &= H^d(\epsilon^x_{k_1,k_2}) \otimes H^d(\epsilon^y_{k_1,k_2}),
\end{aligned}
$$

for $k_1, k_2 = 0, 1$. Here $L(\epsilon)$, $L^d(\epsilon)$, $H(\epsilon)$, and $H^d(\epsilon)$ are given by (12)–(13). In our test, the 2×2 parameter matrices $\boldsymbol{\epsilon}^x$ and $\boldsymbol{\epsilon}^y$ are randomly chosen to be

$$
\boldsymbol{\epsilon}^x = \begin{bmatrix} 0.4751 & 0.3034 \\ 0.1156 & 0.2430 \end{bmatrix}, \qquad \boldsymbol{\epsilon}^y = \begin{bmatrix} 0.4456 & 0.2282 \\ 0.3810 & 0.0093 \end{bmatrix}.
$$

Table 1 gives the PSNR and RE values of the reconstructed images for different Gaussian noise levels, the optimal regularization parameter β^* for the Tikhonov method and also the number of iterations required for Step (ii) in our algorithm. We see that our algorithm is better than the Tikhonov method. Figure 3 depicts the reconstructed high-resolution image with noise at PSNR = 30dB. The values of the parameter λ used in our algorithm are given in Table 2 for reference.

	Least Squares Model			Our Algorithm		
SNR(dB)	PSNR	RE	β^*	PSNR	RE	Iterations
30	28.00	0.0734	0.0367	30.94	0.0524	2
40	28.24	0.0715	0.0353	31.16	0.0511	2

Table 1: The results for the 2×2 sensor array with the periodic boundary condition.

	SNR(dB)=30			SNR(dB)=40		
	First Iteration			First Iteration		
(1,1) sensor	7.895506	8.207295	10.129142	6.874447	7.152288	8.553909
(1,2) sensor	7.279207	7.996708	7.429606	6.580909	7.285088	6.449729
(2,1) sensor	6.294780	6.636084	5.206178	5.644065	6.150834	4.560466
(2,2) sensor	5.456852	6.134216	4.610481	5.044906	5.615505	4.107130
	Second Iteration			Second Iteration		
(1,1) sensor	6.092229	6.646058	4.689150	5.533156	6.070886	4.193728
(1,2) sensor	5.191947	5.777596	4.444771	4.795744	5.417667	3.983984
(2,1) sensor	5.250779	5.795288	4.169715	4.785113	5.407161	3.746287
(2,2) sensor	4.979601	5.730307	4.129098	4.614156	5.288188	3.676112

Table 2: The values of λ used for the 2×2 sensor array with the periodic boundary condition.

4.2 4×4 Sensor Array

We have done similar tests for 4×4 sensor arrays. The bivariate filters are the tensor products of the filters in Example 2. The observed high-resolution image $\mathbf{g}$ is generated by applying

(a)

(b)

(c)

(d)

Figure 3: (a) Low-resolution 128×128 image from the $(0,0)$th sensor; (b) Observed high-resolution 256×256 image (with white noise at SNR=30dB added); (c) Reconstructed 256×256 image from the least squares method with periodic boundary condition; (d) Reconstructed 256×256 image from our algorithm with periodic boundary condition.

the bivariate lowpass filter on the true "Boat" image. Again, true pixel values are used and no boundary conditions are assumed in generating $\mathbf{g}$. After adding white noise, the vector $\mathbf{g}$ is then used in the Tikhonov method and our algorithm to recover $\mathbf{f}$. The matrices $L_{k_1,k_2}(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2})$, $L^d_{k_1,k_2}(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2})$, $H_{k_1,k_2,\nu}(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2})$ and $H^d_{k_1,k_2,\nu}(\epsilon^x_{k_1,k_2}, \epsilon^y_{k_1,k_2})$, $\nu \in \mathbb{Z}^2_4 \setminus \{(0,0)\}$ can be generated by the corresponding filters in Example 2 like what we did in §4.2. In our test,

$$\boldsymbol{\epsilon}^x = \begin{bmatrix} 0.4751 & 0.4456 & 0.4107 & 0.4609 \\ 0.1156 & 0.3810 & 0.2224 & 0.3691 \\ 0.3034 & 0.2282 & 0.3077 & 0.0881 \\ 0.2430 & 0.0093 & 0.3960 & 0.2029 \end{bmatrix}, \qquad \boldsymbol{\epsilon}^y = \begin{bmatrix} 0.4677 & 0.0289 & 0.0694 & 0.1361 \\ 0.4585 & 0.1764 & 0.1014 & 0.0994 \\ 0.2051 & 0.4066 & 0.0994 & 0.0076 \\ 0.4468 & 0.0049 & 0.3019 & 0.3734 \end{bmatrix}.$$

From Table 3, we see that the performance of our algorithm is again better than that of the least squares method. Figure 4 depicts the reconstructed high-resolution image with noise at SNR $= 30$dB. Since the problem is more difficult than the 2×2 case, we see that the algorithm requires few more iterations to get to the solution.

	Least Squares Model			Our Algorithm		
SNR(dB)	PSNR	RE	β^*	PSNR	RE	Iterations
30	24.63	0.1084	0.0492	27.80	0.0752	5
40	24.67	0.1078	0.0505	26.81	0.0751	6

Table 3: The results for the 4×4 sensor array with the periodic boundary condition.

References

[1] N. Bose and K. Boo, *High-Resolution Image Reconstruction with Multisensors*, International Journal of Imaging Systems and Technology, 9 (1998), 294–304.

[2] R. Chan, T. Chan, L. Shen, and Z. Shen, *Wavelet Algorithms for High-Resolution Image Reconstruction*, Research Report #CUHK-2000-20, Department of Mathematics, The Chinese University of Hong Kong, 2000.

[3] A. Cohen, I. Daubechies, and J. Feauveau, *Biorthogonal Bases of Compactly Supported Wavelets,* Comm. Pure Appl. Math., 45 (1992), 485-500.

[4] I. Daubechies, *Ten Lectures on Wavelets*, CBMS Conference Series in Applied Mathematics, Vol. 61, SIAM, Philadelphia, 1992.

[5] D. Donoho, *De-Noising by Soft-Thresholding*, IEEE Trans. Inform. Theory, 41 (1995), 613–627.

[6] D. Donoho and I. Johnstone, *Ideal spatial adaptation by wavelet shrinkage,* Biometrika, 81(1994), 425-455.

[7] R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, 1993.

[8] W. Lawton, S. Lee, and Z. Shen, *Stability and Orthonormality of Multivariate Refinable Functions*, SIAM J. Matrix Anal. Appls., 28 (1997), 999–1014.

[9] M. Ng, R. Chan, T. Chan, and A. Yip, *Cosine Transform Preconditioners for High Resolution Image Reconstruction*, Linear Algebra Appls., 316 (2000), 89–104.

[10] M. Ng, R. Chan, and W. Tang, *A Fast Algorithm for Deblurring Models with Neumann Boundary Conditions*, SIAM J. Sci. Comput., 21 (2000), 851–866.

[11] Z. Shen, *Refinable Function Vectors*, SIAM J. Math. Appl., 29 (1998), 235–250.

[12] S. Skiena, *The Algorithm Design Manual*, Telos/Springer-Verlag, 1997.

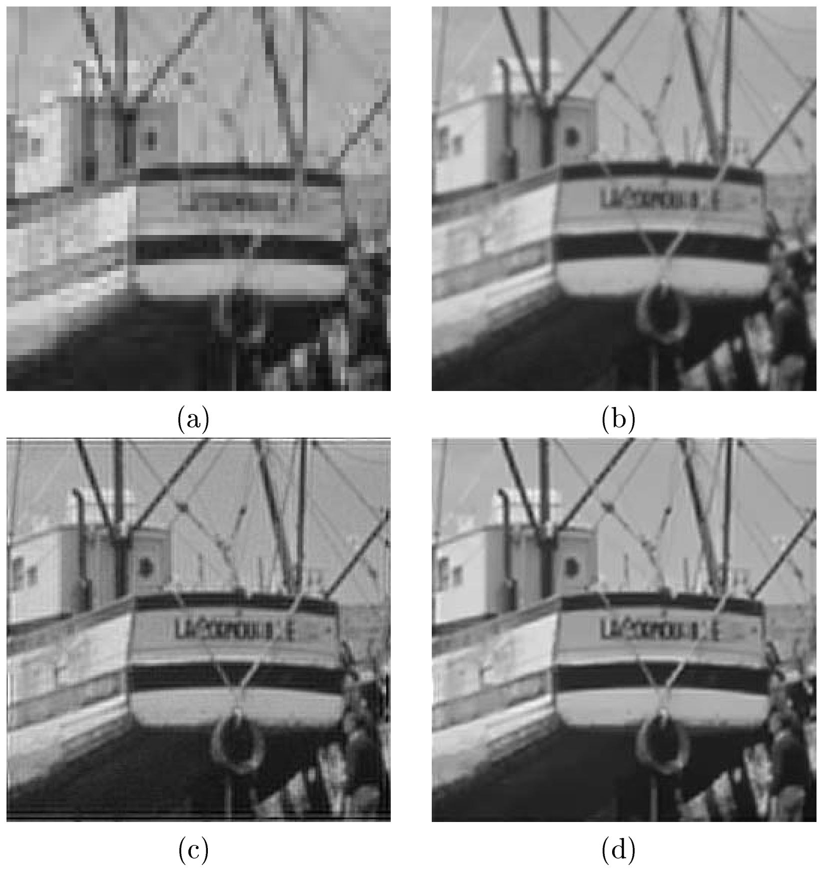

Figure 4: (a) Low-resolution 64×64 image from the $(0,0)$th sensor; (b) Observed high-resolution 256×256 image (with white noise at SNR=30dB added); (c) Reconstructed 256×256 image from the least squares method with periodic boundary condition; (d) Reconstructed 256×256 image from our algorithm with periodic boundary condition.