Image deblurring with matrix regression and gradient evolution

Shiming Xiang a,*, Gaofeng Meng a, Ying Wang a, Chunhong Pan a, Changshui Zhang b

a National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
b Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

1. Introduction

Image deblurring is a classical problem that has been extensively studied in the circles of image processing, computer graphics and computer vision. Although great progresses have been achieved in the past years [1–9], this task still remains far from being solved for real-world applications. As an inverse problem, the main challenge lies in that it is under-constrained since we need to restore the high frequency (sharp) details from the ruined image. In the absence of prior knowledge about the blurring mechanism, this brings intrinsic ambiguity of modeling the blur function and restoring the needed details.

The efforts in image deblurring have surged in recent decades, with the development of numerous approaches and proposals for real-world occasions. Most early methods rely on the deconvolution tricks [10–12], such as Richardson–Lucy algorithm [13,14], Wiener filtering, least-squares deconvolution [10], and so on. The main shortcoming of deconvolution approaches lies in that the deblurring quality largely depends on the kernel estimate. In addition, different tools of mathematical analysis are also employed to deal with this task, typically including wavelet [15,16], variational [17], and regularization [18–22]. Along the line of deconvolution, some approaches are formulated in terms of blind deconvolution, and example methods can be found in [23–25,29,30]. In blind deconvolution, it is not easy to estimate a proper kernel that is well suited to the occasion.

Another family of deblurring algorithms have been developed, explicitly or implicitly, with prior knowledge to help reduce the degrees of freedom of the problem. Typically, natural image statistics are used as prior knowledge to guide the deblurring [31–33,8]. Based on the fact that the statistics of derivative filters on images may be significantly changed after blurring, Levin modeled the expected ones as a function of the width of blur kernel [2]. Fergus et al. proposed to recover the patch images by finding the values with highest probability guided by a prior on the statistics [34], which states that natural images obey heavy-tailed distributions of image gradients. Along this line, approaches of modifying the gradient fields have been proposed with gradient adaptive enhancement [35], gradient penalty by a hyper-Laplacian distribution [36], gradient projection [37], and so on. In addition, information related to sparse representation [27,38,36], color statistics [39], and multi-images [3,40], has also been used to improve the image quality.

The task of deblurring has also been addressed in view of statistic inference or machine learning. A popular modeling tool is the Bayesian framework [41,42,34,8]. Typically, Fergus et al. employed a Bayesian approach to estimate the blur kernel implied by a distribution of probable images [34]. Bayesian framework has also been used to find the most likely estimate of the sharp image [8]. Except the Bayesian frameworks, Su et al. constructed a hybrid learning system, in which both unsupervised and supervised learning methods are employed to deblur images [43]. Later, based on a pair of blurred images, Liu et al. proposed a non-blind deconvolution approach [3]. More recently, Kenig et al. proposed a subspace learning based framework to model the space of point spread functions [44]. To this end, they employed principal component analysis to learn the space from examples at hand [44]. Then, a blind deconvolution algorithm

*Corresponding author. Tel.: + 86 10 6282 5823; fax: + 86 10 6255 1993. E-mail address: sxiang@gmail.com (S. Xiang).

0031-3203/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
is developed for new image to be deblurred. During each iteration of
blind deconvolution, a prior term is added to attract the desired
point spread function to the learned point spread function space.
The application of the proposed algorithm is demonstrated on three-
dimensional images acquired by a wide-field fluorescence micro-
scope, indicating its ability to generate restorations with high
quality.

In the literature, regression has been applied to image restora-
tion. Typically, Hammond and Simoncelli developed a general
framework for combination of two local denoising methods [45].
In their framework, a spatially varying decision function is employed
to balance the two methods. Fitting the needed decision function to
data is then treated as a regression problem. Kernel ridge regression
is finally used to achieve this goal [45].

Additionally, regression is also applied to image deblurring.
Takeda et al. derived a deblurring estimator [6] in terms of kernel
regression. By using the Taylor series, they implicitly assumed that
the regression function is a locally smooth function up to some order.
The locally weighted kernel regression is then employed to achieve
this goal. As a whole, the image to be resorted is ordered into a
column-stacked vector, and the optimization problem is constructed
by integrating together the regression representations of all the
pixels with matrix operator [6]. Formally, this will generate a large-
scale optimization problem for large image. In addition, although
kernel regression is used, their method is unsupervised as no
prediction function is learned from samples at hand for new images.

This paper presents a supervised learning algorithm for image
deblurring. Our algorithm is developed on the conceptual framework
of matrix regression and gradient evolution. To our best knowledge,
this is the first time that the conception of matrix regression (MR) is
presented for image deblurring. In this framework, we do not
estimate the blur kernel, but learn a matrix mapping to transform
image patches to be the desired ones. To this end, a supervised
learning algorithm of MR is proposed to learn the matrix mapping
from the given set including blurred image patches and their
corresponding clear ones. The learned matrix mapping will be used
to map its patches of the image to be deblurred. The mapped result
is then evolved in a gradient field constructed to enhance the edges
of the final image. Comparative experiments illustrate the efficiency
and effectiveness of the proposed method. Its applications to the
interactive deblurring of out-of-focus images also indicate the
validity of our method.

Specifically, the advantages or details of our method can be
highlighted as follows:

1. Beyond blur identification and deconvolution used popularly
   in existing deblurring algorithms, our algorithm is addressed
   into the supervised learning framework, namely matrix
   regression framework. Instead, the learned matrix mapping
   is used to transform the blurred image patches.

2. The MR algorithm is formulated as an optimization problem.
The optimum can be obtained within a few iterations. In each
iteration, only two groups of linear equations should be solved.
The scale of the linear equations is very small since it equals to
the size of the training patches. As a result, the optimization
problem can be efficiently solved.

3. On the whole image level, the computational complexity of
   transforming the blurred image with the learned matrix mapping
   scales linearly in the number of image patches. Moreover,
   the gradient evolution can be fulfilled via pixel-wise update. Thus,
   the computational complexity is also linear in the number of
   pixels. Low computational complexity and low memory require-
   ment will facilitate its real-world applications of our method.

The remainder of this paper is organized as follows. In Section 2,
the MR framework is developed. Section 3 presents the gradient

evolution and describes the deblurring algorithm. Section 4 reports
the experimental results. Section 5 demonstrates the applications
to interactive deblurring of real-world out-of-focus images. Conclu-
sions will be drawn in Section 6.

2. Matrix regression

2.1. Problem formulation

Generally, the image blurring model can be formulated as follows:

\[ G = f * I + n, \]  

(1)

where \( I \) is the imaged objects, \( f \) denotes the imaging system, \( G \)
is the acquired image, \( n \) stands for the pixelwise additive noise, and
\( * \) is a convolution operator. In real world situations, blur often
comes from two types: out-of-focus lens or motion. For example, \( f \)
is usually assumed to be a Gaussian point spread function for
deblurring out-of-focus images. Here our task is to restore \( I \) from \( G \).

As an inverse problem, the task is under-constrained as \( f \) is unknown. This includes two cases. One is that the type of the
kernel is known, while its size and element values are unknown.
Another is that the type, size and element values are all unknown.
Actually, there are many possible solutions to problem (1). Thus,
employing prior knowledge about the blur kernel is fundamentally
necessary to help constrain the solution to the desired images.

Algorithmically, most existing approaches solve problem (1)
with tricks of deconvolution or blind deconvolution, in which a
blur kernel is estimated. Differently, we address this task into the
framework of supervised learning in terms of matrix regression.

As a supervised learning problem, now the task can be
formulated as follows. Suppose we are given \( N \) blurred image
patches in \( \mathcal{X} = \{ A_i \}_{i=1}^{N} \subset \mathbb{R}^{m \times n} \) and their corresponding clear
patches in \( \mathcal{Y} = \{ B_i \}_{i=1}^{N} \subset \mathbb{R}^{m \times n} \), our goal is to find a matrix mapping
\[ B = LA + n, \]

(2)
such that for each patch we have
\[ B_i = LA_i + n, \quad i = 1, 2, \ldots, N. \]

(3)

where \( A \) and \( B \) are two \( m \times n \) matrices, \( n \in \mathbb{R}^{m \times n} \) is a difference term
related to model errors or noises, \( L \) is an \( m \times m \) matrix and \( R \) is an
\( n \times n \) matrix.

The motivation behind the use of the mapping in (2) can be
explained as follows. Intrinsically, as a bilinear mapping, (2) can
be viewed as a combination of row deblurring and column
deblurring. As a whole, a linear restoration of patch \( A \) is achieved
since \( B \) is a linear function of all of the elements in \( A \). In addition,
behind converting patches \( A \) and \( B \) into two column-stacked
vectors \( a \in \mathbb{R}^{mn} \) and \( b \in \mathbb{R}^{mn} \) respectively, directly employing
matrix mapping can facilitate the computation. For example,
suppose the size of the blur kernel (2D filter) is \( 41 \times 41 \). Then,
totally there are 1681 coefficients in \( b = Lw^T a \) to be solved.
To accurately estimate \( w \), one needs to prepare at least 1681
pairs of samples in \( \mathcal{X} \) and \( \mathcal{Y} \) (that is, \( N \geq 1681 \)). Otherwise, we will
obtain an under-determined problem. In contrast, with matrix
mapping, 41 pairs of training samples could be enough to learn
the matrix mapping (see Eqs. (8) and (10) in Section 2.2).

2.2. Matrix regression

To learn \( L \) and \( R \) in (2) from the \( N \) pairs of image patches in \( \mathcal{X} \)
and \( \mathcal{Y} \), we employ the criterion of Bayesian maximum a posteriori
estimation. Under the Gaussian noise assumption, this turns out
to solve the following regularized minimization problem:

$$\min_{L, R} \sum_{i=1}^{N} \| B_i - L A_i R_i \|^2_F + \lambda \| L_i \|^2_F \| R_i \|^2_F,$$

(4)

where $\| \cdot \|_F$ denotes the Frobenius norm of matrix, and $\lambda$ is a positive regularization parameter. In view of Bayesian maximum a posteriori, the first term of the objective function in model (4) corresponds to the likelihood, while the second term corresponds to the prior, which is related to the regularization for smoothness of $L$ and $R$.

In the literature, the mapping with the form in (2) has been actually employed in tensor subspace learning [46–50]. Typically, Ye developed an approach of tensor subspace approximation. In contrast to problem (4), his model can be formulated as [46]

$$\min_{A_i, B_i} \sum_{i=1}^{N} \| B_i - L A_i R_i \|^2_F.$$  

(5)

As a subspace representation of $A_i$ matrix, $B_i$ in (5) is unknown in advance. Thus, it is necessary to introduce constraints $L_i L_i^T = I$ and $R_i R_i^T = I$ to make the problem solvable.\(^1\) This results in that the problem in (5) can only be solved in way of eigenvalue decomposition of matrix. Differently in our model, $B_i$ is known, and thus we need not introduce the two additional constraints. Without performing eigenvalue decomposition of matrix, our model can be solved via linear equations.

Based on Frobenius norm, for the first term in (4), we have

$$\sum_{i=1}^{N} \| B_i - L A_i R_i \|^2_F = \sum_{i=1}^{N} \text{tr}(L A_i R_i^T L^T A_i^T - 2 \text{tr}(L A_i R_i R_i^T A_i^T) + \text{tr}(L A_i R_i R_i^T A_i^T)).$$

$$= \sum_{i=1}^{N} \text{tr}(R_i^T A_i^T L^T A_i R_i - 2 \text{tr}(L A_i R_i R_i^T A_i^T) + \text{tr}(L A_i R_i R_i^T A_i^T)).$$

(6)

where tr is the trace operator of matrix.

Now it is easy to check that the optimization problem in (4) with respect to $L$ is convex when $R$ is fixed. This fact holds since matrix $\sum_{i=1}^{N} A_i R_i R_i^T A_i^T$ is positively semi-definite. Likewise, the problem with respect to $R$ is also convex when $L$ is fixed. However, problem (4) will not be jointly convex with respect to $L$ and $R$ together (An example is given in Appendix A.). Fortunately, we can also efficiently solve problem (4) in an alternative way as $L$ and $R$ are separated from each other.

For clarity, we denote the objective function in (4) by $G(L, R)$. Based on (6), differentiating the objective function in (4) with respect to $L$ and setting the derivative to be zero, we get

$$\partial G(L, R)/\partial L = 0$$

$$\Rightarrow \sum_{i=1}^{N} L_i (A_i R_i R_i^T A_i^T)^T = \sum_{i=1}^{N} B_i R_i R_i^T A_i^T + \lambda \| L_i \|^2_F = 0.$$

(7)

Thus, $L$ can be solved via the following linear equations:

$$L_i \left( \sum_{i=1}^{N} A_i R_i R_i^T A_i^T + \lambda \| R_i \|^2_F I_m \right) = \sum_{i=1}^{N} B_i R_i R_i^T A_i^T,$$

(8)

where $I_m$ is an $m \times m$ identity matrix.

Furthermore, based on (6), differentiating the objective function with respect to $R$ and setting the derivative to be zero, we have

$$\partial G(L, R)/\partial R = 0 \Rightarrow \sum_{i=1}^{N} (A_i^T L_i L_i A_i - \lambda \| L_i \|^2_F I_m) R_i = 0.$$

(9)

---

\(^1\) Note that, according to the Theorem 3.2 in [46], solving problem (5) is just equivalent to solving the following optimization problem: $\max_{L_i L_i^T = I} \min_{R_i R_i^T = I} \sum_{i=1}^{N} \| B_i - L_i A_i R_i \|^2_F$. We see there exists many solutions if the two constraints $L_i L_i^T = I$ and $R_i R_i^T = I$ are removed.

Thus, $R$ can be solved via the following linear equations:

$$\left( \sum_{i=1}^{N} A_i^T L_i L_i A_i + \lambda \| L_i \|^2_F I_m \right) R_i = \sum_{i=1}^{N} A_i^T L_i^T B_i,$$

(10)

where $I_n$ is an $n \times n$ identity matrix.

From Eqs. (8) and (10), we see that $L$ and $R$ could be solved from $\min(m, n)$ training samples, without the need to prepare $m \times n$ samples.

In addition, the above deduction about $L$ and $R$ indicates that we can develop an alternate iteration algorithm to solve the optimization problem. The steps of the MR algorithm are listed in Algorithm 1. In each iteration, we only need to solve two groups of linear equations.

The convergence of the algorithm can be theoretically guaranteed. This can be explained as follows. Actually, in Algorithm 1, steps 4 and 5 solve two subtasks, each of which is a standard Quadratic Programming (QP) (An explanation is given in Appendix B.). Let the solution obtained at the $i$th iteration be $(L^i, R^i)$. Then, based on the convexity of QP problem, we have $G(L^i, R^i) \geq G(L^{i+1}, R^i) \geq G(L^{i+1}, R^{i+1})$. This indicates that Algorithm 1 is always convergent. In general, it will be converged within about 10 iterations.

Algorithm 1. Matrix regression (MR) algorithm (for training).

**Input**: $N$ blurred image patches $(A_i)_{i=1}^{N} \subset \mathbb{R}^{m \times n}$ and their corresponding $N$ clear patches $(B_i)_{i=1}^{N} \subset \mathbb{R}^{m \times n}$, the regularization parameter $\lambda$; and the maximum number of iterations $T$.

**Output**: $L \in \mathbb{R}^{m \times m}$ and $R \in \mathbb{R}^{n \times n}$.

1: Let $R = I_n$, $L = 0$, $t = 1$ and $\varepsilon = 10^{-7}$.
2: while $t \leq T$ do
3: Record $R_0 = R$ and $L_0 = L$.
4: Solve $L$, according to (8).
5: Solve $R$, according to (10).
6: if $(\| R_0 - R \|^2_F + \| L_0 - L \|^2_F) < \varepsilon$ then
7: Stop.
8: end if
9: $t \leftarrow t + 1$.
10: end while

Finally, we analyze the computational complexity of Algorithm 1. With matrix operator, it can be easily justified that solving $L$ in (8) will cost about $O(N(2mn^2 + mn^2))$. In parallel, solving $R$ in (10) will cost about $O(N(2nm^2 + nm^2))$. As a result, the computational complexity will be linear in the number of training patches. Given $N = 10,000$, $21 \times 21$ patch size and $T = 8$, for example, it will take only about 2.08 s to run Algorithm 1, using Matlab 7.0 on a PC with 3.00 GHz CPU and 4.0G RAM.

### 2.3. The MR algorithm for image deblurring

When applying the learned $L$ and $R$ to an image to be deblurred, we need to divide the image into patches with $m \times n$ pixels. This task can be achieved in a row-scan way. Thus, the patches are overlapped with each other. The averaged values can be taken for the pixels in the overlapped regions. Algorithm 2 gives the steps of the above computations.

The main computations in Algorithm 2 lie in the matrix mapping in step 6. For a patch with $m \times n$ pixels, the computational complexity of this step will scale in about $O(4nm^2 + mn^2)$. Including the computations in step 11, the whole computational complexity of Algorithm 2 will be up to about $O(N(2m - 2 + [m/2] + 1)w)$. We see it is linear in the number
of the pixels of the image to be deblurred. For example, suppose the patch size is $21 \times 21$, for image with $481 \times 321$ pixels, finishing the computations in Algorithm 2 will cost totally about 3.95 s, using Matlab 7.0 on a PC with 3.0 GHz CPU and 4.0G RAM.

Algorithm 2. Image deblurring with MR (for prediction).

**Input**: Image $G$ with $h$ rows and $w$ columns to be deblurred, the matrix mapping with $L \in \mathbb{R}^{m \times m}$ and $R \in \mathbb{R}^{n \times n}$.

**Output**: The deblurred result $I$.

1. Allocate two matrices: $I, F \in \mathbb{R}^{h \times w}$, and let $I = 0, F = 0$.
2. for $r = [\frac{h}{2}] + 1, \ldots, [\frac{h}{2}]$ do
3.     for $c = [\frac{w}{2}] + 1, \ldots, [\frac{w}{2}]$ do
4.         Get the patch of $m \times n$ pixels with pixel $(r,c)$ as its center, and denote this patch (matrix of intensities) by $U$.
5.         Construct the set of the coordinates of the pixels in $U$: $C = \{(x,y) | c-\frac{m}{2} \leq x \leq c+\frac{m}{2}; r-\frac{n}{2} \leq y \leq r+\frac{n}{2}\}$.
6.         Transform $U$ and accumulate the result into the corresponding sub-matrix of $I$, namely, $I(C) = I(C) + LUR$.
7.         Count the times of the treated pixels, $F(C) = F(C) + 1$.
8.     end for
9. end for
10. for each pixel $(r,c)$ with $1 \leq r \leq h$ and $1 \leq c \leq w$ do
11.     Average the accumulated result, $I(r,c) = I(r,c)/F(r,c)$.
12. end for
13. Output matrix $I$ as the final result $I$.

3. Deblurring with MR and gradient evolution

3.1. Gradient modification

In Section 2, we have developed MR algorithm for image deblurring. Here we give two examples to illustrate its ability. The first column in Fig. 1 shows two blurred images obtained by Gaussian blurring function with parameter $\sigma = 1.9$ and filter size $s = 15$ pixels. To learn a matrix mapping, we employed five (grayscale) images from the Berkeley database [51] (see Fig. 3) and blurred them with the same Gaussian blur kernel. Then totally $N = 1000$ pairs of blurred-clear patches of $15 \times 15$ pixels are randomly sampled from these images. After a matrix mapping is learned with Algorithm 1, it is supplied to Algorithm 2. Note that it will be called three times for red, green and blue bit planes for color image. The deblurred results are shown in the second column in Fig. 1. We see the images are significantly deblurred. In contrast to the ground truth as shown in the last column in Fig. 1, however, some edges are still not clear. This can be witnessed from the edges in the regions indicated by the rectangles in the second column in Fig. 1 and the zoomed regions shown in the bottom panel in Fig. 1.

Actually, gradient diffusion along edges will largely cause the decrease of clarity, which may be more easily perceived by human
eye. In other words, clear image has sharp gradient distribution, while blurred image has blur gradient distribution. Here we use Fig. 2 to explain this fact. Fig. 2(a) and (b) shows a blurred circle and a clear circle. Their gradient magnitudes are given in Fig. 2(c) and (d). As can be seen from Fig. 2(c), the gradient magnitudes of the pixels along the circle are significantly diffused. Such diffusion decreases the clarity of edges. This motivates us to modify the gradients to improve the quality.

To this end, we propose to extract the central pixels crossing along the diffused edge regions. For convenience, we call these pixels as edge profiles. In other words, edge profiles are the pixels located on the central line along the diffused edges, and they have largest gradient magnitudes across the diffusing directions. Thus, edge profiles are defined here to capture the desired edges. Fig. 2(e) gives an example to show the edge profiles of the blurred circle. In the literature, Steger had developed such an algorithm that can be used to fulfill our task [52]. In computation, the algorithm is run with the gradient magnitudes of the image as input. The third column in Fig. 1 shows the edge profiles of two natural images extracted by Steger's algorithm. In general, this algorithm is very effective, and the computation is also fast. Given a gradient magnitude image with 481 × 321, for example, it will take only about 0.5 s to finish the computations.

Note that, with the proposed MR algorithm, we have already had a good initial solution. Thus, we need not compress the diffusions, but increase the gradient magnitudes of the edge profiles to enhance the clarity of the edges. Let \( I_0 \) be the deblurred image obtained with the MR algorithm (Algorithm 2). Further let \( g_i \) be the gradient magnitude of pixel \( p_i \) in \( I_0 \). Then, we modify the gradient magnitudes as follows:

\[
\tilde{g}_i = \begin{cases} 
  s \cdot g_i, & \text{if } p_i \text{ is on the edge profile}, \\
  g_i, & \text{otherwise},
\end{cases}
\]  

(11)

where \( \tilde{g}_i \) is the final magnitude of pixel \( p_i \). In (11), \( s \) is set to be a number greater than 1.0 so that the gradient magnitude of pixel \( p_i \) is enlarged. As we have a good initial image \( I_0 \), we fix \( s = 1.25 \) in our implementation to slightly enhance the gradient magnitudes. Although this is just an empirical choice, experiments on many types of natural images show its good performance (see the fourth column in the first panel in Fig. 1).

### 3.2. Gradient evolution

Given a blurred image \( G \), we use \( I_0 \) to denote the deblurred result obtained by Algorithm 2, and use \( \nabla I_0 \) to denote its gradient, whose gradient magnitudes are modified according to (11). To obtain the final deblurred image \( I \), we introduce the following optimization problem:

\[
\min_{\nabla I} ||I - I_0||^2 + \gamma ||\nabla I - \nabla I_0||^2.
\]  

(12)

The first term in (12) is the fitting constraint on the pixel intensities, which means that the desired image should not change too much from that obtained with the MR algorithm. The second term is the fitting constraint on the gradients, which means that the gradient magnitudes of the pixels on the extracted edge profiles will be enlarged. As a result, the gradient distribution along edges will become more sharp. In (12), \( \gamma \) is a positive trade-off parameter, which is introduced to balance the contributions of these two terms.

Note that problem (12) is convex, and can be equivalently formulated as a large-scale QP problem (An explanation is given in Appendix C). However, this will result in large scale matrix operations. To solve problem (12) efficiently, we consider its continuous formulation of the objective function:

\[
\int \int (f(x,y) - f_0(x,y))^2 + \gamma \|
abla f(x,y) - \nabla f_0(x,y)\|^2 \, dx \, dy,
\]  

(13)

where \( f(x,y) \) and \( f_0(x,y) \) are the intensities of \( I \) and \( I_0 \) at \( (x,y) \), \( \nabla f(x,y) \in \mathbb{R}^2 \) is the gradient of image \( I \) at \( (x,y) \), and \( \nabla f_0(x,y) \in \mathbb{R}^2 \) is the modified gradient, according to (11).

Let \( F = (f(x,y) - f_0(x,y))^2 + \gamma \|
abla f(x,y) - \nabla f_0(x,y)\|^2 \). According to variational theory, function \( F \) that minimizes the above integral must satisfy the Euler–Lagrange equation [53]:

\[
\frac{\partial F}{\partial x} - \frac{\partial}{\partial x} \frac{\partial F}{\partial f_x} - \frac{\partial}{\partial y} \frac{\partial F}{\partial f_y} = 0,
\]  

(14)

where \( f_x \) and \( f_y \) denote the gradient magnitudes along \( x \) and \( y \) directions. Based on (14), we have [53]

\[
f(x,y) - f_0(x,y) - \gamma \nabla^2 f(x,y) + \gamma \nabla \cdot \nabla f_0(x,y) = 0,
\]  

(15)

where \( \nabla^2 \) is the Laplacian operator, and \( \nabla \cdot \nabla f_0 \) stands for the divergence of gradient \( \nabla f_0 \).

Finally, problem (12) can be minimized by a gradient descent algorithm:

\[
f^{t+1}(x,y) = f^t(x,y) - \tau L,
\]  

(16)

where \( L = f(x,y) - f_0(x,y) - \gamma \nabla^2 f(x,y) + \gamma \nabla \cdot \nabla f_0(x,y) \), and \( \tau \) is the step size during iterations.

### 3.3. The Algorithm

Algorithm 3 lists the steps of our algorithm of matrix regression and gradient evolution (MRGE) for image deblurring. In step 11, \( m(||I_1 - I||^2) \) calculates the average of the squared image differences over all pixels. This step is introduced to justify whether the iteration can be now stopped.

### Algorithm 3. Image deblurring with MRGE.

**Input**: Image \( G \) to be deblurred, the matrix mapping with \( L \in \mathbb{R}^{m \times m} \) and \( R \in \mathbb{R}^{n \times n} \), parameters \( \gamma \) in (12) and \( \tau \) in (16), and the maximum number of iterations \( T \) for gradient descent approach.

**Output**: The deblurred image \( I \).

1. Run Algorithm 2 to obtain \( I_0 \).
2. Calculate the gradient \( \nabla I_0 \) of \( I_0 \) and the magnitudes \( g_0 \).
3. Extract edge profiles from \( g_0 \) with Steger's method [52].
4. Modify the gradient magnitudes of \( g_0 \) according to (11).
5. Let \( I_1 = 0, T = I_0, t = 1, \) and \( \varepsilon = 10^{-7} \).
6. while \( t < T \) do
   7. for each pixel of \( I \) located at \( (x,y) \) do
      8. Update the intensity according to (16).
   9. end for
10. if \( m(||I_1 - I||^2) < \varepsilon \) then
11. Stop.
12. end if
13. \( I_1 = I \).
14. \( t = t + 1 \).
15. end while
Note that Steger’s method (step 3) is very fast. Thus, the main computations will be located in steps 7–9. In these steps, there are only pixel-wise computations. The computational complexity is linear in the number of pixels, and thus the computation is very fast.

Note that problem (12) is convex (see Appendix C), which has only one global optimum. In Algorithm 3, we employ the gradient descent approach to find the optimal solution. As $I_0$ is a good initial point, tens of iterations will converge to the final solution. In our implementation, we fix the maximum number of iterations $T$ to be 30. The results in the fourth column in the upper panel in Fig. 1 are obtained with the above setting for iterations. Compared with those listed in the second column in Fig. 1, we see the deblurring quality is further improved.

Now we have developed three algorithms. The relations between them can be illustrated as follows. As a supervised learning algorithm, matrix mapping is learned under the MR framework via Algorithm 1. Algorithm 2 is actually in the prediction phase after the training is fulfilled. In addition, Algorithm 2 provides an initial solution to run Algorithm 3.

4. Experimental evaluation

4.1. Parameter settings of matrix regression

We have developed a supervised learning algorithm for image deblurring. Algorithm 1 gives the steps of training. Except the model parameters $L$ and $R$, it has three parameters: regularization parameter $\lambda$, size of training patches $m \times n$, and number of training patches $N$. In addition, it also includes, implicitly, the parameters of blur function used to generate the blurred patches. Fig. 3 lists the five images [51] used to generate the training data.

4.1.1. Performance of parameter $\lambda$

In these experiments, the images are blurred with Gaussian blur kernels. That is, the point spread function is a Gaussian function with parameter $\sigma$. To construct training samples, we first blur the five images as shown in Fig. 3 with $\sigma=1.9$ and kernel size $s=15$. Accordingly, this will generate five pairs of blurred and clear images. Then, we randomly sample $N=1000$ pairs of blurred and clear patches from these images as training data. The size of each patch is also taken as $15 \times 15$. That is, we set $m=15$ and $n=15$. In addition, Algorithm 1 is run with $\lambda=10^{-3}, 10^{-4}, 10^{-5}$. Therefore, totally 11 matrix mappings are learned. Each mapping will be employed by Algorithm 2 to deblur the test images.

The blurred images for test are also generated by Gaussian blur function with $\sigma=1.9$ and $s=15$. Here five test images are reported in Fig. 4. The deblurred results are obtained with $\lambda=10^{-3}, 10^{-4}, 10^{-5}$, and 1000, respectively. Furthermore, we take the peak signal-to-noise ratio (PSNR) to give a quantitative evaluation about the performance of parameter $\lambda$. The PSNR is calculated by taking the ground truth image as a reference. Fig. 5 shows the PSNR curves of the five blurred images, different numbers of training patches. These five blurred images are also used in Fig. 4. We see there do not exist significant (perceptible) differences between the results obtained even with $N=1000$ and $N=2000$ training samples.

4.1.2. Performance of parameter $N$

To evaluate the performance of the number of training samples, nine training sets are constructed, with $N=300, 500, 1000, 2000, 4000, 8000, 10,000, 15,000, 20,000$, respectively. The pairs of blurred and clear patches are also randomly taken from the five images (see Fig. 3). A matrix mapping will be learned from each training set. When learning these matrix mappings, we take $\lambda=0.0001$. All the other parameters are taken as the same as those used in Fig. 4. Fig. 5 shows the PSNR curves of the five blurred images, with different numbers of training patches. These five blurred images are also those used in Fig. 4. We see there do not exist significant (perceptible) differences between the results.

4.1.3. Performance of parameters $m$ and $n$

In Fig. 5(a) and (b), the size of the training patches is exactly equal to that of the blur kernel. However, it may be unknown in real-world applications. To illustrate if the patch size will affect the performance of deblurring, 12 training data sets are constructed, with patch size $m \times n=7 \times 7, 11 \times 11, 21 \times 21, 31 \times 31, ..., 101 \times 101, 111 \times 111$, respectively. In addition, all the other parameters are set as the same as those used in Fig. 4. In experiments, each training set contains 1000 pairs of blurred and clear patches randomly sampled from the five images shown in Fig. 3. After the matrix mappings are learned with different patch sizes, they are employed to deblur the five images used in Fig. 4. Fig. 5(c) shows the PSNR curves. We see the deblurring accuracy is decreased if the size of the training patches is smaller than that of the true blur kernel. This is reasonable as there is no adequate spatial size to capture the blurring. When the size of training patches is greater than that of the blur kernel, each curve shows no significant change on the image.

Furthermore, note that only 1000 pairs of blurred and clear patches are used as samples. Taking $m \times n=111 \times 111$ for example, L and R in (2) totally have 24 642 elements to be estimated. However, if we select to estimate a point spread function with $111 \times 111$ pixel size, one could prepare at least 12 321 samples. What is more, solving a group of linear equations with 12 321 variables on a PC is not an easy task as the coefficient matrix in $R^{12321 \times 12321}$ is non-sparse.

4.1.4. Performance of Gaussian parameter $\sigma$

In the above three experiments, the blurred images and the test images are generated by Gaussian blurring function with the same parameter $\sigma$. In real world applications, $\sigma$ may not be exactly known. To test the performance of our MR algorithm in this case, here the training patches are collected with different parameters $\sigma$. Specifically, given a $\sigma$ and its variance (bias) $s$, a number $t$ is randomly selected from interval $[\sigma-s, \sigma+s]$. Then, we take $t$ as the Gaussian parameter to blur the image randomly selected from those shown in Fig. 3. From the blurred image and

Fig. 3. The five images [51] used to generate the training data. The size of each image is $481 \times 321$ or $321 \times 481$ pixels.
the corresponding clear image, only one pair of blurred and clear image patches are randomly selected. Thus, given a pair of $s$ and $s$, a training data set of $N = 1000$ samples can be constructed, by performing the above steps 1000 times. In experiments, we take $s = 1:5:7, 1:7:9, 1:9:11, 2:1:3$ and $2:5$, and set $s = 0.0, 0.05, 0.1, 0.15, \ldots$, and 0.5, respectively. Totally 66 training sets will be constructed in this way. Patches with different degrees of blurring are mixed together for learning. In this process, the size of training patches is set to be $m/C_n = 11/7, 13/7, 11/11, 21/21, 31/21, 41/21, \ldots$, and $101/101$. Thus, in this way, totally 66 groups of $L$ and $R$ will be learned by Algorithm 1, each of which will be supplied respectively to Algorithm 2 for deblurring the test image.

The test images are generated by Gaussian blur function with $s = 1:5, 1:7, 1:9, 2:1, 2:3$, and 2:5, respectively. Then the learned matrix mappings are employed respectively to deblur the five images used in Fig. 4. Fig. 6 gives the PSNR curves of the five images. The variance used for sampling is labeled near the corresponding curve. For clarity, Fig. 7 gives two examples, where the test images are generated by Gaussian function with $s = 1:9$. In the first column are the results deblurred with the learned matrix mapping with $s = 0$. In this case, the test images and the blurred images for training are all generated with the same Gaussian blur function. From the second to the last column in Fig. 7 are the results obtained with matrix mappings learned from the patches of the blurred images with $s$ randomly in $[1:9 - s, 1:9 + s]$, where $s = 0.0, 0.1, 0.2, 0.3, 0.4$, and 0.5, respectively. As can be seen from Fig. 7, there are no significant differences between the deblurred results. Actually, in Fig. 6, the largest difference of PSNR values is about 1.2. This indicates that the MR algorithm can be adaptive to a large range of changing of parameter $s$, without degrading significantly the quality of deblurring. This will facilitate its real-world applications.

4.1.5. Algorithm convergence

In Section 2.2, we mentioned that generally Algorithm 1 will be converged within about 10 iterations. Fig. 8 illustrates the experimental evidences about this fact. In Fig. 8, the curves are obtained
with the same experimental setting used in Section 4.1.1 for evaluating the performance of parameter $\lambda$ in Algorithm 1. That is, in experiments, the same $N=1000$ pairs of blurred and clear patches are employed here to run Algorithm 1. Fig. 8 reports the values of the objective function in (4). The curves obtained with $\lambda = 10^{-7}, 10^{-4}, 10^{-2},$ and 1.0 are illustrated here for clarity. We see that the convergence is achieved within about 10 iterations. We also tested the convergence performance with experiments by taking different numbers of training samples $N$, different sizes of training patches. It can be summarized that similar descending curves can be obtained within 10 iterations.

4.1.6. Discussions for motion blurring

Besides out of focus lens, motion is another basic source of blurring in real-world situations. We evaluated the performance of our developed MR algorithm when it is applied to motion blurring. To construct a training data set, the five images in Fig. 3 are blurred with $s$ randomly in $[1.9 - s, 1.9 + s]$. Thus, patches with different blur degrees are mixed together for learning. In the first column, “$s=0.0$” means that the test images and the blurred images for training are all generated with the same Gaussian blur function.

Fig. 6. The PSNR curves of the five deblurred images, obtained with matrix mappings learned from the patches of the blurred images with $s$ randomly in $[1.9 - s, 1.9 + s]$, where $s=0.0, 0.05, 0.1, 0.15, \ldots$, and 0.5, respectively. The corresponding $s$ used in experiments is labeled near each curve. The five test images to be deblurred are those used in Fig. 4.

Fig. 7. From the first to the last column are the deblurred results, by taking $s=0.0, 0.1, 0.2, 0.3, 0.4,$ and 0.5, respectively. In experiments, each matrix mapping is learned from $N=1000$ patches. Each patch is randomly selected from one of the five images in Fig. 3, and each image is blurred by a Gaussian blur function with parameter $s$ randomly selected from $[1.9 - s, 1.9 + s]$. Thus, patches with different blur degrees are mixed together for learning. In the first column, “$s=0.0$” means that the test images and the blurred images for training are all generated with the same Gaussian blur function.

Fig. 8. The curves of the values of the objective function in (4) with respect to the number of iterations by taking different $\lambda$. 

with the same experimental setting used in Section 4.1.1 for evaluating the performance of parameter $\lambda$ in Algorithm 1. That is, in experiments, the same $N=1000$ pairs of blurred and clear patches are employed here to run Algorithm 1. Fig. 8 reports the values of the objective function in (4). The curves obtained with $\lambda = 10^{-7}, 10^{-4}, 10^{-2},$ and 1.0 are illustrated here for clarity. We see that the convergence is achieved within about 10 iterations. We also tested the convergence performance with experiments by taking different numbers of training samples $N$, different sizes of training patches. It can be summarized that similar descending curves can be obtained within 10 iterations.
with a motion descriptor with seven pixel length and 30° motion direction. Then \( N = 1000 \) training patches are randomly selected from these blurred images and their corresponding clear ones. The patch size is \( 15 \times 15 \) pixels. Then a matrix mapping is learned with Algorithm 1, by taking \( \lambda = 0.0001 \). Finally, the learned matrix mapping is employed to deblur the images, which are blurred with the same motion descriptor used to construct the training patches. The first row in Fig. 9 shows the blurred images, while the second row illustrates the deblurred results. We see that the image quality has not significantly improved. Actually, the matrix mapping with the form in (2) has no ability to describe the anisotropic spatial motion.

Although our approach is unsuitable for motion blur images, it shows the power to deal with Gaussian blur. For this case, most traditional methods have been developed in terms of deconvolution. Under deconvolution frameworks, estimating an accurate blur kernel is essential to guarantee the deblurring quality. This is actually an open problem that has not been well solved currently. Differently, we view the task as a problem of pattern recognition and use matrix regression to learn a matrix mapping. This generates a flexibility framework as knowledge in training samples is used to guide the deblurring. In addition, the formulation with matrix regression has low complexity (see Section 2.3). This will facilitate its real-world applications.

### 4.2. Parameter settings of MRGE

Given the learned matrix mapping, MRGE has only two parameters, namely, the trade-off parameter \( \gamma \) in (12), and the iteration step \( \tau \) in (16). In experiments, the training set used in Fig. 4 is employed here to evaluate the performance. When training the matrix mapping, we take \( \lambda = 0.0001 \). The test images are also those used in Fig. 4. In experiments, we take \( \tau = 0.2 \). Fig. 10 shows the deblurred results. We see the image is sharpened with the increase of \( \gamma \). This can be explained as the second term in (13) is a gradient fitting constraint, where the gradient magnitudes of the pixels on the edge profiles are increased. Actually, a larger \( \gamma \) indicates that this constraint should contribute more to the final result. For example, in the case of \( \gamma = 5.0 \), the results are all over sharpened. However, there are no significant differences between the results when we take \( \gamma \in [0.0005, 0.5] \). In other words, \( \gamma \) is insensitive in this interval. This fact also be witnessed from the PSNR curves shown in Fig. 11(a), which are obtained with \( \gamma = 0.0005, 0.005, 0.05, 0.5, 1.0, 2.0 \) and \( 5.0 \), respectively.

Fig. 11(b) shows the results obtained with \( \tau = 0.0002, 0.002, 0.02, 0.2, 1.0, 2.0 \) and \( 5.0 \), respectively. In experiments, we take \( \gamma = 0.5 \). All other parameters are set equally to those used in Fig. 11(a). We see that there are almost no differences between
the results obtained with $\tau$ in [0.0002, 0.2]. We also see that the deblurring quality will be significantly decreased in the case of $\tau \geq 2.0$. Actually, a large $\tau$ indicates that the iteration will may walk along the image gradient related to Laplacian operator, which will cause image sharpening via gradient amplification. In applications, we can take $\tau = 0.2$.

Finally, we test the convergence of Algorithm 3 via experiments. In Section 3.3, we have suggested that the number of iterations for running Algorithm 3 can be set as 30. Fig. 12 gives experimental evidences about this fact. The curves in Fig. 12 are obtained with the same experimental setting used in Fig. 10. To calculate the values of the objective function in (12), we set $\gamma = 0.5$ and $\tau = 0.2$ when running Algorithm 3. We see that, on all of the five images in Fig. 10, Algorithm 3 converges within 30 iterations. This indicates that we could set the maximum number of iterations as 30 to run Algorithm 3.

4.3. Comparisons

As a classical deblurring algorithm, Richardson–Lucy (RL) algorithm [14,13] will be taken as the baseline for comparison. In addition, two typical yet popularly used algorithms, the Weighted Least Squares (WLS) [10], and the Sparse Prior (SP) [36], will be also compared. In experiments, the true blur kernel is supplied to these algorithms to guarantee that they can generate the best results.

In experiments, the matrix mapping is learned from the same training set used in Fig. 4 with $z = 0.0001$. The learned mapping is then employed to deblur different types of natural images. When running Algorithm 3, we take $\gamma = 0.5$ and $\tau = 0.2$, respectively. To run SP algorithm, we downloaded the source codes from the author’s homepage and ran them for deblurring. In addition, all the test images are generated with the same parameters used in Fig. 4.

![Fig. 11.](image1.png)  ![Fig. 12.](image2.png)

Fig. 11. (a) The PSNR curves of the five deblurred images, by taking $\gamma = 0.0005, 0.005, 0.05, 0.5, 1.0, \text{ and } 2.0$; (b) the PSNR curves of the five deblurred images, by taking $\tau = 0.0002, 0.002, 0.02, 0.2, 1.0, \text{ and } 2.0$. In (a) and (b), the x-axis stands for the index of the value. For example, “index of $\gamma = 1$” in (a) indicates that the result is obtained with $\gamma = 0.0005$.

Fig. 12. The curves of the values of the objective function in (12) with respect to the number of iterations on the five (grayscale) images in Fig. 10.
Figs. 13 and 14 illustrate the deblurred results of different types of natural images. The source images are available in public Berkeley image database [51], Grabcut image database [54] or Corel Photo database. All the images are deblurred with the original sizes. Only a part of each image is shown here for clarity. From Figs. 13 and 14, we see that the deblurred results obtained by our algorithm are more clear. To further compare these algorithms, Table 1 gives a quantitative comparison. The number in Table 1 stands for the PSNR of the deblurred image, by taking the ground truth as a reference. The first column in Table 1 indicates the indices of the eight images in Figs. 13 and 14. We see that our algorithm achieves the highest accuracy.

In computation time, for image with $481 \times 321$ pixels, WLS, RL, SP and our algorithm will take about 91.2 s, 9.7 s, 713.2 s, and 8.32 s, respectively, using Matlab 7.0 on a PC with 3.0 GHz CPU and 4.0G RAM. We see our algorithm is much faster than SP and WLS. It is also slightly faster than RL. This will facilitate its real-world applications.

5. Applications

In this section, we show some applications of our method to deblurring of real-world blurred images due to out-of-focus lens. The foundation of such applications is lain on the well-known fact

<table>
<thead>
<tr>
<th>No. of images</th>
<th>WLS</th>
<th>RL</th>
<th>SP</th>
<th>MRGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29.5301</td>
<td>29.6253</td>
<td>29.5415</td>
<td>30.4187</td>
</tr>
<tr>
<td>2</td>
<td>20.5080</td>
<td>25.0782</td>
<td>20.3688</td>
<td>21.8470</td>
</tr>
<tr>
<td>3</td>
<td>26.0958</td>
<td>25.8023</td>
<td>26.2432</td>
<td>27.3504</td>
</tr>
<tr>
<td>4</td>
<td>29.9976</td>
<td>30.1190</td>
<td>26.2432</td>
<td>30.9667</td>
</tr>
<tr>
<td>5</td>
<td>29.7448</td>
<td>29.7943</td>
<td>30.4088</td>
<td>29.0762</td>
</tr>
<tr>
<td>6</td>
<td>25.7576</td>
<td>25.4701</td>
<td>25.7021</td>
<td>26.6259</td>
</tr>
<tr>
<td>7</td>
<td>25.0128</td>
<td>25.8022</td>
<td>24.9695</td>
<td>25.9310</td>
</tr>
<tr>
<td>8</td>
<td>27.7502</td>
<td>27.4152</td>
<td>27.5944</td>
<td>28.6480</td>
</tr>
</tbody>
</table>

Table 1: PSNRs of the deblurred images in Figs. 11 and 12.

Fig. 13. Demo I: deblurred results with WLS, RL, SP and our algorithm. The last column shows the ground truth for comparison.

Fig. 14. Demo II: deblurred results with WLS, RL, SP and our algorithm. The last column shows the ground truth for comparison.
Fig. 15. The edge profiles and their diffusions extracted in interactive image segmentation setting. In each row, the first column illustrates the user specified strokes about the blurred object (foreground) and the background, while the second column shows the segmentations obtained by interactive image segmentation. The last column shows the extracted edge profiles and their diffusions over the object region to be deblurred, where the diffusions of the extracted edge profiles are illustrated along their normal directions.

Fig. 16. Demo I (the man on the left): deblurred results with WLS, RL, SP and our algorithm. The source image is available at http://slide.sports.sina.com.cn/o/slide_2_18735_10734.html#p=6.

Fig. 17. Demo II (the man): deblurred results with WLS, RL, SP and our algorithm. The source image is available at http://www.wisdom.weizmann.ac.il/~vision/courses/2009_2/files/Blind_Deconvolution.ppt.
that, on the same deep plane, out-of-focus blurring can be approximately described by a Gaussian blur kernel.

Note that in real world situations the blur degree is actually unknown. This motivates us to construct a candidate set to describe different blur degrees. To this end, we employ the five images in Fig. 3, and blur them by using Gaussian blur kernels with different parameters $\sigma$. Totally, 15 sample sets are constructed, with $\sigma = 0.9, 1.1, 1.3, \ldots,$ and 3.7, respectively. Here large $\sigma$ corresponds to large blur degree. In experiments, each set contains $N = 1000$ pairs of blurred and clear patches randomly sampled from these five images, which are then employed by Algorithm 1 to learn a matrix mapping. Our task is to select an approximate matrix mapping for test image.

As Algorithm 3 does not take care of how to estimate the blurring degree, for an image to be deblurred, we need to estimate a proper $\sigma$ to select one of the learned matrix mappings, according to the above

---

**Fig. 18.** Demo III (the rose): deblurred results with WLS, RL, SP and our algorithm.

**Fig. 19.** Demo IV (the man on the right): deblurred results with WLS, RL, SP and our algorithm.
construction of sample sets. Unfortunately, estimating \( \sigma \) actually becomes another open problem. Alternatively, we employ Steger’s method \([52]\) again to estimate the diffusion width of edges caused by blurring. In computation, this task can be achieved by two steps. The first step is to estimate the edge profiles of the images blurred by Gaussian blur kernel with parameter \( \sigma \). The second step is to estimate the diffusion width of edges along the normal directions of the extracted edge profiles. These two tasks can be solved simultaneously by Steger’s method in the gradient field of image. Then, the averaged diffusion width is calculated to label the blur degree. For robustness, this value is calculated only from the first three longest edge profiles. Finally, it is used to label the degree of blurring corresponding to this \( \sigma \).

Fig. 15 shows three examples, where some objects are blurred due to out-of-focus lens. To deblur the object, an interactive image segmentation is used to cut out it from the background. In Fig. 15, the first column shows the user specified strokes about the foreground (the blurred object) and the background. The second column in Fig. 15 is the segmented results with local spline regression algorithm \([55]\). The third column shows the extracted edge profiles and diffusions along the normal directions of the edge profiles. For robustness, only the inner region of the blurred object is considered when performing Steger’s method.

Like the training phase, the diffusion width is also calculated from the first third longest edge profiles located in the inner region of the blurred object. This value is used to label the blur degree of the object. Among the previously learned matrix mappings, the one corresponding to the nearest degree is then selected. This mapping will be finally applied to the segmented region of the object. When running Algorithm 3, we take \( \gamma = 0.5 \) and \( \tau = 0.2 \).

Note that, based on the estimated diffusion width, we can find a \( \sigma \) among the training sets. This task can be achieved as each sample set is generated by Gaussian blur function with known \( \sigma \) and it is latter labeled by diffusion width. This indicates that the corresponding \( \sigma \) as well as the Gaussian blur kernel can be employed by WLS, RL and SP algorithms. Figs. 16–20 report the experimental results of five images taken in real world scenes. In contrast, our algorithm achieves the highest visual quality. Actually, WLS and RL generate unsatisfactory results for these images. In contrast, the performance of our algorithm is comparable to that of SP. However, the result obtained by SP looks slightly smooth and lacks some details (for example, the rose in Fig. 18). In addition, in Fig. 17, a large degree of ring appears in the result obtained by SP (see the edge between the man and the girl). This can also be perceived from the hair of the man in Fig. 20. In addition, for image with \( 481 \times 321 \) our algorithm will only take

![Fig. 20. Demo V (the man): deblurred results with WLS, RL, SP and our algorithm.](image-url)
a few seconds to generate the final result, while SP algorithm will take about 700 s on a PC with 3.0 GHz CPU and 4.0G RAM, using Matlab 7.0.

6. Conclusion

In this paper, we proposed a novel algorithm for image deblurring. We formulated the task as a problem of matrix regression and gradient evolution. Matrix regression technique is proposed to learn a matrix mapping, with which each blurred image patch can be directly mapped to be a new patch with more sharp details. The quality of the deblurred image is further enhanced by gradient evolution in the promoted gradient field. We also analyzed the performance of the proposed algorithm, including the convergence, the computational complexity, and the parameter setting. Comparative experiments illustrate the validity of our method. Finally, the applications of the proposed algorithm to interactive deblurring of blurred objects due to out-of-focus lens have also been reported to illustrate its validity in real-world situations.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant nos. 60975037, 61005036, 61175025, and 91120301), and the National Basic Research Program of China (Grant no. 2012CB316304).

Appendix A. About the non-convexity of problem (4)

In Section 2, we point out that problem (4) may be non-convex. Here we give an example to illustrate this fact. Suppose $A$ and $B$ are two $2 \times 2$ identity matrix. We only need to prove function $f(L, R) = \|A - LBR\|_F^2$ is non-convex.

To this end, we denote $L = (L_t, \; L_r)$, $R = (R_t, \; R_r)$. Further let $x = [x_1, \; x_2, \ldots, x_4]^T \in \mathbb{R}^8$ collect the elements of $L$ and $R$. Then function $f(L, R)$ can be formally denoted by $f(x)$. Let $x_1 = [-1, \; 0, \; 0, \; -1, \; 0, \; 0, \; 0, \; 1]^T$, $x_2 = [1, \; 0, \; 0, \; 1, \; 0, \; 0, \; 0, \; 1]^T$, and $\alpha = \beta = \frac{1}{2}$. Now it is easy to check that

$$ f(x_1) + f(x_2) = 0. \quad (17) $$

According to the property of convex function [56], (17) indicates that $f(x)$ is not convex. Thus problem $\min_{L, \; R} \|A - LBR\|_F^2$ is not convex. This can further imply that problem (4) may be non-convex.

Appendix B. About the QP problems related to Eqs. (8) and (10)

Here our task is to illustrate the fact that solving (8) is just equivalent to solving a QP problem.

Let $H = \sum_{i=1}^n A_1 R_i A_1^T + A_2 R_2 I_m$ and $D = \sum_{i=1}^n B_1 R_i A_1^T$. Then matrix $L$ in (8) is just the solution to the following problem:

$$ \min_{L} \frac{1}{2} tr(HL^T L) - tr(LD). \quad (18) $$

Further denote $L = [l_1, \; l_2, \ldots, l_m]^T \in \mathbb{R}^{m \times m}$, where vector $l_i \in \mathbb{R}^m$ collects the $m$ components in the $i$-th row of $L$. Similarly, we denote $D = [d_1, \; d_2, \ldots, d_m]^T \in \mathbb{R}^{m \times m}$. Then, we have

$$ \frac{1}{2} tr(HL^T L) - tr(LD) = \frac{1}{2} l_1^T H l_1 - d_1 + \cdots + \frac{1}{2} l_m^T H l_m - d_m = \frac{1}{2} l^T H l - d. $$

where $H_0 = \text{diag}(H, H, \ldots, H) \in \mathbb{R}^{m^2 \times m^2}$, and $t = [t_1, \; t_2, \ldots, t_m]^T \in \mathbb{R}^{m^2}$. As a result, we see that problem (18) is equivalent to the following problem:

$$ \min_{t} \frac{1}{2} t^T H_0 t - t^T d. \quad (19) $$

Note that $H_0$ is positively semi-definite. Thus $H_0$ is also positively semi-definite. Accordingly, we see (19) is just a convex QP problem. This indicates that solving the linear equations in (8) is just equivalent to solving a standard QP problem.

Finally, we point out that the above analysis can be also made on Eq. (10).

Appendix C. About the convexity of problem (12)

Here our task is to illustrate the fact that problem (12) is convex. Let $\nabla_x I$ and $\nabla_y l$ be the $x$-direction gradient and the $y$-direction gradient of images $\nabla I$. Then we have $\|\nabla I - \nabla I_0\|^2_2 = \|\nabla_x I - \nabla_x I_0\|^2_2 + \|\nabla_y I - \nabla_y I_0\|^2_2$. We further convert $I, I_0, \nabla I_0$ and $\nabla y I$ into column-stacked $x, x_0, g_x, g_y$, $\in \mathbb{R}^m$, respectively. Note that $\nabla_x$ and $\nabla_y$ are two linear operators. By assembling them into matrices $G_x \in \mathbb{R}^{m \times m}$ and $G_y \in \mathbb{R}^{m \times m}$ [57], we have $\|\nabla_x I - \nabla_x I_0\|^2_2 = \|G_x x - g_x\|^2_2$ and $\|\nabla_y I - \nabla_y I_0\|^2_2 = \|G_y y - g_y\|^2_2$.

As a result, problem (12) will be equivalent to

$$ \min_{x} \|x - x_0\|^2_2 + \gamma \|G_x x - g_x\|^2_2 + \gamma \|G_y y - g_y\|^2_2. \quad (20) $$

It is easy to check that problem (20) is a QP problem. It is also convex as the Hessian matrix $(I + \gamma G_x G_x^T + \gamma G_y G_y^T)$ is positively semi-definite.

In computation, problem (20) can be solved via a group of linear equations. However, as the Hessian matrix is a large-scale matrix, we consider to solve it in an iterative way (see Section 3.2).

References

Changshui Zhang received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986 and the Ph.D. degree from Tsinghua University, Beijing, China, in 1992. In 1992, he joined the Department of Automation, Tsinghua University, and is currently a Professor. His interests include pattern recognition, machine learning, etc. He has authored over 200 papers. Prof. Zhang currently serves on the editorial board of the Pattern Recognition Journal.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, China, in 1993, the M.S. degree from Chongqing University, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2004. From 1996 to 2001, he was a postdoctorate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently an Associate Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing. His interests include pattern recognition and image processing.

Gaofeng Meng received the B.S. degree in applied mathematics from Northwestern Polytechnical University, Xi’an, China, in 2002, and the M.S. degree in applied mathematics from Tianjin University, Tianjin, China, in 2005, and the Ph.D. degree in control science and engineering from Xian Jiaotong University, Xi’an, Shaanxi, China, in 2009. In 2009, he joined the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, as an Assistant Professor. His research interests include image processing, computer vision, and pattern recognition.

Ying Wang received his B.S. degree from the Department of Information and Communications Technologies, Nanjing University of Information Science and Technology, Wuhu, China. He was a postdoctorate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently a Ph.D. candidate in the Institute of Automation, Chinese Academy of Sciences, Beijing. His research interests include computer vision and pattern recognition.

Chunhong Pan received his B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1987, his M.S. degree from Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, China, in 1990, and his Ph.D. degree in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences, Beijing, in 2000. He is currently a Professor at National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, image processing, computer graphics, and remote sensing.