Numer Algor (2013) 64:43–72
DOI 10.1007/s11075-012-9654-y

ORIGINAL PAPER

Blind image deconvolution using a banded matrix method

Antonios Danelakis · Marilena Mitrouli · Dimitrios Triantafyllou

Received: 3 June 2011 / Accepted: 24 September 2012 / Published online: 27 October 2012

Abstract In this paper we study the blind image deconvolution problem in the presence of noise and measurement errors. We use a stable banded matrix based approach in order to robustly compute the greatest common divisor of two univariate polynomials and we introduce the notion of approximate greatest common divisor to encapsulate the above approach, for blind image restoration. Our method is analyzed concerning its stability and complexity resulting to useful conclusions. It is proved that our approach has better complexity than the other known greatest common divisor based blind image deconvolution techniques. Examples illustrating our procedures are given.

Keywords Blind image deconvolution · Blurring function · Univariate polynomial · Greatest Common Divisor · Banded matrix · Convolution

Mathematics Subject Classifications (2010) 65F05 · 65G50 · 65Y20

A. Danelakis
Department of Informatics & Telecommunications,
University of Athens, Athens, Greece
e-mail: a.danelakis@gmail.com

M. Mitrouli (✉)
Department of Mathematics, University of Athens,
Panepistemiopolis 15784, Athens, Greece
e-mail: mmitroul@math.uoa.gr

D. Triantafyllou
Section of Mathematics & Engineering Sciences,
Department of Military Sciences, Hellenic Army Academy, Vari, Greece
e-mail: dtriant@math.uoa.gr

1 Introduction

Image restoration, using blind deconvolution methods, is an old problem in Image Processing, but it continues to attract the attention of researchers. A number of real-world problems from Astronomy to Satellite Imaging, finds applications for image restoration algorithms. Many scientific areas, such as Image Processing or Satellite Imaging, use the optical result of a rendering device, as input of their proceedings. Unfortunately, this input is often modified by blur, noise and measurement errors. The latest are the three situations that must be handled in Image Processing. Image blurring reduces the details and information of the image due to the additive haziness. Noise is expressed through the random variety of the image brightness, or color, and corrupts the image. There are many types of noise such as, Gaussian noise, salt and pepper noise, shot noise, quantization noise, film grain and anisotropic noise. Measurement errors also corrupt the image information as well as the noise.

Let P be the initial clean image, B the final modified image, F the blurring function, E the measurement errors and N the noise. Depending on the artifacts taking place, there are the following cases:

1. If there are only image blurring conditions, B can be defined as the convolution: $B = P * F$ (see Fig. 1b).

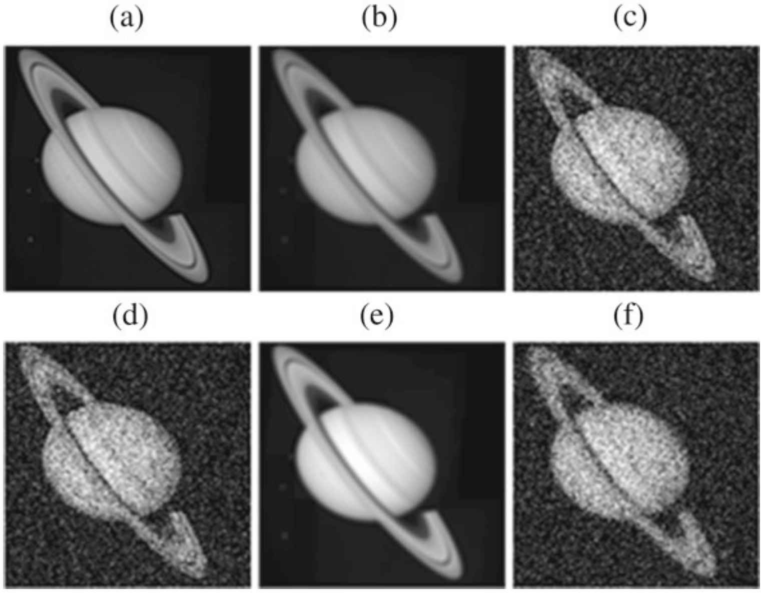


Fig. 1 Examples of artifacts impact on images. **a** Clear image. **b** Blurring. **c** Noise. **d** Blurring and noise. **e** Blurring and measurement errors. **f** Blurring, noise and measurement errors

2. In pure additive noise conditions, B can be defined as: $B = P + N$ (see Fig. 1c).
3. If there are measurement errors, in addition to image blurring conditions, then it holds: $B = (P + E) * F$ (see Fig. 1e).
4. In noise and blurring conditions there are two sub-cases:

   4.1 The initial image P will be blurred and then noise will be inserted: $B = P * F + N$ (see Fig. 1d).

   4.2 The initial image P includes measurement error as well. In the sequence, image P will be blurred and finally, noise will be inserted: $B = (P + E) * F + N$ (see Fig. 1f).

In case 1, image restoration can be resolved using image deblurring filters such as Linear, Wiener or Regularized, in case function f is known. Case 2 can be resolved using an image denoising algorithm [1, 4, 6, 10, 17]. Cases 3 and 4, which combine blurring, noise and measurement errors, are mostly found in real applications. We consider that the terms F, E and N are unknown. Previous works done for resolving cases 3 and 4 involve the development of blind image deconvolution methods. The mathematical tools, used by these methods, vary. In [29] a constant modulus algorithm is presented. In [2, 25] iterative blind deconvolution algorithms are illustrated. In [22] an equalization algorithm is used, while in [19] a recursive filtering algorithm is preferred. In addition, for blind image deconvolution, methods such as Bussgang deconvolution [20], alternating Krylov subspaces [27], fixed-point algorithms [9], maximum likelihood and projection [18], greatest common divisor [8, 15, 26] and total variation have been used [14, 31].

The present paper focuses on cases 1, 3 and 4. For 1D blurring, the proposed algorithms can handle filters of the size of the number of elements of a row or a column of the picture. For 2D blurring, our methods work for small blurring kernel. A small kernel is very common in practise and represents a great amount of blurring situations. More specifically it appears in the areas of medical imaging and robot and machine vision [13, 30] and in many common applications such as Magnetic Resonance Imaging (MRI), Computational Tomography (CT). In many of the above applications the blurring is caused by Gaussian or Laplacian filters, whose blurring kernel size is small compared to the image. In order to restore the initial image, we use a banded matrix based blind image deconvolution approach. Our contribution is the use of the Sylvester matrix, in order to compute the required Greatest Common Divisor (GCD), resulting to the image restoration. We present a stable method triangularizing the Sylvester matrix arising from two polynomials of the same degree. This approach reduces the complexity of the classical methods from $O(n^3)$ to $O(n^2)$ flops for computing the GCD of two polynomials of degree n (this algorithm will be applied to two blurred segments of n pixel each). In addition, for handling measurement errors and noise in blind deconvolution problem, we introduce the approximate GCD computations [21]. Pillai and Liang [26] also use a GCD method, but instead of the Sylvester matrix, they consider the Discrete Fourier Transformation (DFT). An improved version of Pillai and Liang method was introduced in [15] by Raymond Heindl, which also uses DFT to compute the GCD. Our approach reduces the required complexity one order than the existing algorithms using $DFTs$ for the GCD computation. Furthermore, we propose a new approach achieving the blind restoration of an image blurred with $2D$ (2-dimensional) blurring functions.

This paper is organized as follows. In Section 2, the mathematical formulation of the blind image deconvolution problem is presented. In Section 3, we introduce the fast upper triangularization of a modified Sylvester matrix, leading to the robust algorithms $MSGCD$ and $MAGCD$ specifying the exact and approximate GCD of univariate polynomials respectively. In Section 4, we present our approach (MA_1) achieving blind image deconvolution. A new method (VHD) for image restoration, assuming that the image is blurred using a $2D$ blurring function (i.e. vertical and horizontal blurring) satisfying certain properties, is introduced. Both methods are illustrated through numerical examples. In Section 5 we illustrate the performance of the presented methods and interesting comparisons are given. Comments concerning the error analysis and the computational complexity of the methods are also incorporated. Finally, in Section 6, we analyze the outcoming conclusions of the paper and propose future challenges.

2 Problem formulation

In applications a $2D$ image P of dimensions $m \times n$ is represented by a $2D$ matrix of size $m \times n$.

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & p_{0,n-1} \\ p_{1,0} & p_{1,1} & p_{1,2} & \cdots & p_{1,n-1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ p_{m-1,0} & p_{m-1,1} & p_{m-1,2} & \cdots & p_{m-1,n-1} \end{bmatrix} \tag{1}$$

The (i, j) element of matrix P corresponds to the color values of the position (i, j) (**pixel**) of the image. The equivalent image representation in $1D$ (1-dimensional) space is achieved by folding matrix P of (1).

$$P = \left[\, p_0 \; p_1 \; p_2 \; \dots \; p_{n-1} \; p_n \; p_{n+1} \; p_{n+2} \; \cdots \; p_{m \cdot n-1} \,\right] \tag{2}$$

If the image is grayscale, only one matrix is enough for its representation. Each element of the matrix corresponds to the grayscale values, with value 0 corresponding to black and value 255 corresponding to white. If the image has colors, then we need three such matrices to represent it, one for the color values of each basic color of RGB model (Red, Green and Blue).

An image can be alternatively represented by a $2D$ polynomial of the form given in (3).

$$\mathtt{p}(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} P_{i,j} \cdot x^i \cdot y^j \tag{3}$$

or by 1D polynomial expressed in (4)

$$\mathtt{p}(x) = \sum_{i=0}^{m \cdot n-1} P_i \cdot x^i. \tag{4}$$

Quite similar is the representation of a blurring function. A blurring function affects all the pixels of the image. Its purpose is to coincide its middle element with every pixel and transform the color of the pixel according to the contribution weights of its neighbouring pixels, expressed by the function. Because a middle element is needed, the dimensions of a blurring function are usually odd.

Let P be an image of dimensions $m \times n$ $(n > m)$ and F an $r \times r$ blurring function with $m > r$. The operation of convolution between the blurring function F and the image P which produces the blurred image B of dimension $(m + r) \times (n + r)$, in matrix level is defined for each pixel (i, j) of the initial image as (5) illustrates.

$$B(i, j) = \sum_{k=0}^{r-1} \sum_{l=0}^{r-1} P(i - k, j - l) \cdot F(k, l) \tag{5}$$

Respectively, in polynomial level is even simpler expressed as (6) in 2D

$$b(x, y) = \mathtt{p}(x, y) \cdot f(x, y) \tag{6}$$

or as (7) in 1D

$$b(x) = \mathtt{p}(x) \cdot f(x) \tag{7}$$

The image may contain noise and/or measurement errors as well. Taking these factors into account, (6) and (7) are modified to the following equations respectively.

$$b(x, y) = (\mathtt{p}(x, y) + e(x, y)) \cdot f(x, y) + n(x, y) \tag{8}$$

$$b(x) = (\mathtt{p}(x) + e(x)) \cdot f(x) + n(x) \tag{9}$$

In the above equations, the terms $e(x, y)$ and $e(x)$ express the $2D$ and $1D$ polynomial representations of the measurement errors and the terms $n(x, y)$ and $n(x)$ the $2D$ and $1D$ polynomial representations of the noise.

The problem of blind image deconvolution is to define the polynomial $\mathtt{p}$ or the matrix P of the initial image, if the modified image polynomial b or matrix B is known.

Throughout the paper, we will focus on the polynomial definition. We will use grayscale images for conception facilitation. Colour images can be handled similarly.

3 Sylvester matrices for image restoration

Blurring an image means that a blurring function is applied to all rows and columns of it. This means, that every pixel of the blurred picture is a term of the product of the polynomial representing the clean one with the polynomial representing the blurring function. So, if we select two segments of the modified image containing the whole blurring function information and represent them by two polynomials, then finding their Greatest Common Divisor (GCD) is equal with finding the blurring function. Dividing then, the modified image with the computed GCD, we derive the clean one. So, it is of great importance to construct efficient and reliable methods specifying the GCD.

Let a(s) and b(s) be two polynomials representing two segments of a blurred image:

$$a(s) = a_n s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0$$
$$b(s) = b_n s^n + b_{n-1} s^{n-1} + ... + b_1 s + b_0$$

with $\vartheta\{a(s)\} = \vartheta\{b(s)\} = n$, where $\vartheta\{p(s)\}$ denotes the degree of the polynomial $p(s)$. We modify the classical Sylvester matrix [3], formed by the above $a(s), b(s)$, as follows.

$$S^*(a,b) = \begin{bmatrix} a_n & a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_0 & 0 & 0 & \dots & 0 \\ b_n & b_{n-1} & b_{n-2} & b_{n-3} & \dots & b_0 & 0 & 0 & \dots & 0 \\ 0 & a_n & a_{n-1} & a_{n-2} & \dots & a_1 & a_0 & 0 & \dots & 0 \\ 0 & b_n & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & 0 & \dots & 0 \\ 0 & 0 & a_n & a_{n-1} & \dots & a_2 & a_1 & a_0 & \dots & 0 \\ 0 & 0 & b_n & b_{n-1} & \dots & b_2 & b_1 & b_0 & \dots & 0 \\ . & . & . & . & \dots & . & . & . & \dots & . \\ 0 & 0 & 0 & 0 & \dots & 0 & a_n & a_{n-1} & \dots & a_0 \\ 0 & 0 & 0 & 0 & \dots & 0 & b_n & b_{n-1} & \dots & b_0 \end{bmatrix}$$

The modified Sylvester matrix has n same blocks of dimension $2 \times (n+1)$.

3.1 Fast upper triangularization of the modified Sylvester matrix

Let $S^* \in \mathbb{R}^{2(n+1)\times 2(n+1)}$ be the modified Sylvester matrix. There exists an orthogonal matrix $Q \in \mathbb{R}^{2(n+1)\times 2(n+1)}$ and an upper triangular matrix $R \in \mathbb{R}^{2(n+1)\times 2(n+1)}$ such that

$$Q^T \cdot S^*(a,b) = R.$$

This upper triangularization of the special structured matrix $S^*(a,b)$ can be efficiently computed as follows. For simplicity we will denote $S^*(a,b)$ as S^*.

Step 1 Let

$$B_0 = \begin{bmatrix} a_n & a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \\ b_n & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 \end{bmatrix}$$

and H_1 the Householder matrix achieving the following transformation.

$$H_1 \cdot B_0 = \begin{bmatrix} a_n^{(1)} & a_{n-1}^{(1)} & a_{n-2}^{(1)} & a_{n-3}^{(1)} & \dots & a_0^{(1)} \\ 0 & b_{n-1}^{(1)} & b_{n-2}^{(1)} & b_{n-3}^{(1)} & \dots & b_0^{(1)} \end{bmatrix}.$$

The block B_0 is repeated in the remaining rows of S^* right shifted per one element each time, thus is only needed to zero the second row of every B_0 and update the other entries without any additional calculations, by applying the following transformation.

$$diag(H_1, \dots, H_1) \cdot S^* = \begin{bmatrix} a_n^{(1)} & a_{n-1}^{(1)} & a_{n-2}^{(1)} & a_{n-3}^{(1)} & \dots & a_0^{(1)} & 0 & 0 & \dots & 0 \\ 0 & b_{n-1}^{(1)} & b_{n-2}^{(1)} & b_{n-3}^{(1)} & \dots & b_0 & 0 & 0 & \dots & 0 \\ 0 & a_n^{(1)} & a_{n-1}^{(1)} & a_{n-2}^{(1)} & \dots & a_1 & a_0^{(1)} & 0 & \dots & 0 \\ 0 & 0 & b_{n-1}^{(1)} & b_{n-2}^{(1)} & \dots & b_1^{(1)} & b_0^{(1)} & 0 & \dots & 0 \\ 0 & 0 & a_n^{(1)} & a_{n-1}^{(1)} & \dots & a_2^{(1)} & a_1^{(1)} & a_0^{(1)} & \dots & 0 \\ 0 & 0 & 0 & b_{n-1} & \dots & b_2^{(1)} & b_1^{(1)} & b_0^{(1)} & \dots & 0 \\ . & . & . & . & \dots & . & . & . & \dots & . \\ 0 & 0 & 0 & 0 & \dots & 0 & a_n^{(1)} & a_{n-1}^{(1)} & \dots & a_0^{(1)} \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & b_{n-1}^{(1)} & \dots & b_0^{(1)} \end{bmatrix}$$

$$= (S^*)^{(1)}$$

where $diag(H_1, \dots, H_1)$ denotes a diagonal matrix with blocks $A_i, i = 1, \dots, k$ in the main diagonal.

Let

$$B_1 = \begin{bmatrix} b_{n-1}^{(1)} & b_{n-2}^{(1)} & b_{n-3}^{(1)} & \dots & b_0^{(1)} & 0 \\ a_n^{(1)} & a_{n-1}^{(1)} & a_{n-2}^{(1)} & \dots & a_1 & a_0^{(1)} \end{bmatrix}.$$

After step 1, $(S^*)^{(1)}$ contains $n-1$ same blocks B_1, right shifted per one entry each time.

Step 2 We use a Householder transformation H_2 to zero the first element of the second row of B_1 and update the other entries. Next, we apply the transformation $diag(I_1, H_2, \dots, H_2) \cdot (S^*)^{(1)} = (S^*)^{(2)}$, where I_i denotes the $i \times i$ identity matrix.

Since the initial matrix is of dimension $2n \times 2n$, we apply $2n-3$ iterations. In each iteration we have to zero only one element and to update at most $n+1$ elements. In the final step is needed to apply a Householder transformation H_{2n-2} to the right bottom 3×3 resultant block.

Let $Q_1 = diag(H_1, \ldots, H_1)$ and $Q_i = diag(I_{i-1}, H_i, \ldots, H_i), i = 2, \ldots, 2n-2$. Setting $Q = Q_1^T \cdot Q_2 \cdot \ldots Q_{2n-2}^T$, we obtain $Q^T S^* = R$.

Computational complexity Applying the above described triangularization to the modified $2n \times 2n$ Sylvester matrix we reduce per one order the complexity of the classical triangularization methods [7]. The total complexity of this modified QR factorization is $O(6n^2)$ flops in contrast with the $O(\frac{16n^3}{3})$ flops of the classical one.

Error analysis The classical error analysis of QR factorization [7] can be adjusted to our case as follows.

The QR factorization of S^* is the exact QR factorization of a slightly perturbed matrix $\hat{S}^* = S^* + E_1$ satisfying

$$S^* + E_1 = Q \cdot R,$$

with

$$\|E_1\|_F \leq 12.5n \cdot \mu \cdot \|S^*\|_F,$$

where Q is an orthogonal matrix computed from Householder's transformations and μ is the machine precision.

3.2 Greatest Common Divisor of two polynomials

Below, we present a proposition relating the previous factorization with the computation of the GCD of two polynomials.

Proposition 1 *If we perform QR factorization to the Sylvester matrix, then the nonzero elements of the last nonzero row of the upper triangular resultant matrix defines the coefficients of GCD in reverse order* [3].

The above proposition, properly modified, holds for $S^*(a, b)$ also.

The following algorithm attains the image restoration through the computation of the exact GCD of polynomials.

Algorithm *MSGCD* (Modified Sylvester *GCD*)

```
BEGIN
1. Construct the matrix $S^*$ from 2 segments of the image.
2. Perform the fast upper triangularization of $S^*$.
3. $GCD$ = the last non zero row of the resultant upper triangular matrix.
END
```

Implementation of the MSGCD The numerical evaluation of *MSGCD* algorithm, due to error measurements, noise and floating point errors, can lead to prime polynomials. In order our proposed procedure to compute efficiently the GCD we use a variable inner tolerance ϵ_t. If any element during the QR process is less in absolute value than ϵ_t we zero it. Cases 1 and 3 can be handled using an accuracy of smaller order than the one appearing in error measurements. $MSGCD$ algorithm is more sensitive in cases that include noise (cases 2 and 4). By choosing ϵ_t to be of the same order with the noise, $MSGCD$ efficiently computes the GCD, having an error of the same order with the perturbation.

3.3 Approximate GCD of polynomials

The computation of the exact GCD of polynomials often is not realistic. In order to handle even better cases 2 and 4, we can compute the approximate GCD of polynomials instead of the exact one. The approximate GCD of polynomials is computed relaxing the exact conditions for computing the exact GCD. In the case that the computation of the exact GCD is pointless, the use of the above mentioned method, which is based on matrices and the computation of the "near null space" [21] through the Singular Value Decomposition (SVD), gives approximate solutions which may approach the blurring function more efficient. It follows the $MAGCD$ algorithm.

Algorithm $MAGCD$ (Modified Approximate GCD)

```
BEGIN

1. Define a threshold $t > 0$.
   Apply the SVD algorithm to $S^*$ using the modified QR factorization (Theorem 1) for the first phase of SVD.
   Define a basis M for the right "near nullspace" of the Sylvester matrix $S^*$ as follows:
   $S^* = U \Sigma V^T$. The columns of $V$ corresponding to the smaller than $t$ singular values, define the right "near null space" of $S^*$.
2. Define the $GCD$ Matrix Pencil $Z(s) = s \cdot M_1 - M_2$, where $M_1$ and $M_2$ are the matrices obtained from $M$ by deleting the last and the first row of $M$ respectively.
3. Form the polynomial matrix $Z(s)$ with elements all nonzero determinants of maximal order.
   Specify the matrix $B$ containing the coefficients of the polynomials of $Z(s)$.
4. Find the SVD of $B$: $B = \widetilde{U} \cdot \widetilde{\Sigma} \widetilde{V}^T$.
   The corresponding to the largest singular value column of $\widetilde{V}$ defines the approximate $GCD$.

END
```

Implementation of the $MAGCD$ The above algorithm is implemented in a hybrid way, combining symbolic and numeric operations. The complexity of the numerical part of $MAGCD$ is of $O(n^3)$ flops. Cases that include noise can be handled using an even smaller accuracy (e.g. of order $O(10^{-16})$ for noise of order $O(10^{-3})$) than ϵ_t needed for the computation of the exact GCD.

4 Image recovery using one modified instance

Let P be the $m \times n$ initial, clean image and F the $r \times r$ blurring function, where $r < n$. We add measurement error E and then F is applied through convolution on the resulting matrix. Finally, noise N is added resulting into a degraded instance of the clean image $B = (P + E) * F + N$. The degraded instance has dimensions $(m + r) \times (n + r)$. Given the modified degraded instance B, our purpose is to restore image P.

Pillai and Liang [26] and Heindl [15] suggest that we have to consider two distinct partitions (segments) of the modified image. These partitions are *sub-images* of B. This means that they can be represented by two $2D$ polynomials. After the convolution multiplication between the polynomials of the clean image and the blurring function, a new polynomial is produced which corresponds to the modified image. Each polynomial coefficient corresponds to the color value of the pixel in the position specified by the power of respective variables x, y. This coefficient is produced after the multiplication of some elements of the filter and some elements of the initial image, and after the addition of the coefficients with the same power of the corresponding variables x, y. The goal is to find two groups of pixels, which constitute the two pre-mentioned sub-images. Each group consults from the multiplication of all the elements of the blurring function with some elements of the initial image. It is clear that the GCD of these two sub-images, would be the polynomial of the blurring function.

Since we know the blurring function, we can find the initial image. The key here is to select the two sub-images so that each one contains the whole

Algorithm A_1 [15]

```
BEGIN

1.  Read the $2D$ matrix $B$.
2.  Partition the matrix into 2 sub-images $B_1, B_2$, containing the whole blurring function.
3.  Find the $2D$ polynomials of $B_1, B_2$, namely $b_1(x, y), b_2(x, y)$, using (3).
4.  Calculate the $2D$ polynomial corresponding to the blurring function as:
        $f(x, y) = GCD(b_1(x, y), b_2(x, y))$
5.  Calculate the $2D$ polynomial corresponding to the initial image as:
        $p(x, y) = b(x, y)/f(x, y)$

END
```

blurring function. The noise smoothing process takes place during their GCD procedure, but they do not take into consideration measurement errors.

Algorithm A_1 requires the computation of the GCD of two $2D$ polynomials. In [15, 26] a DFT based algorithm is applied for this computation. A tight lower bound is not known on the number of DFT operation required additions, although lower bounds have been proved under some restrictive assumptions on the algorithms. The lowest published count for power-of-two block length K was long achieved by Frigo and Johnson [11] and Lundy and Van Buskirk [23] and is approximately $\frac{34}{9} \cdot K \cdot \log_2 K$.

If the minimal number of floating point multiplications required for computing the exact DFT of block length K, is denoted by $M_{DFT}(K)$, then the following theorem holds.

Theorem 1 *For a given $K = \prod_{i=1}^{m} p_i^{e_i}$, where $p_i, i = 1, ..., m$ are distinct primes and $e_i, i = 1, ..., m$ are positive integers, it follows:*

$$M_{DFT}(K) = 2 \cdot K - \sum_{i_1=0}^{e_1} \sum_{i_2=0}^{e_2} \cdots \sum_{i_m=0}^{e_m} \phi\left(GCD\left(\prod_{i=1}^{m} p_j^{i_j}, 4\right)\right) \cdot 1$$
$$+ \sum_{d_1 | \frac{\phi(p_1^{i_1})}{\phi(GCD(p_1^{i_1},4))}} \sum_{d_2 | \frac{\phi(p_2^{i_2})}{\phi(GCD(p_2^{i_2},4))}} \cdots \sum_{d_m | \frac{\phi(p_m^{im})}{\phi(GCD(p_m^{im},4))}}$$
$$\times \frac{\prod_{k=1}^{m} \phi(d_k)}{\phi(LCM(d_1, d_2, ..., d_m))},$$

where $\phi(.)$ is the Euler's totient function, $GCD(.,.)$ denotes the greatest common divisor, and $LCM(.,.)$ is the least common multiple [16].

In addition, most algorithms compute the DFT *approximately*, with an error that can be made arbitrarily small at the expense of increased computations [5, 12, 28].

$MSGCD$ and $MAGCD$ algorithms do not use DFT operation and work in the $1D$ matrix—polynomial form taking advantage of the structure of modified Sylvester matrix.

Image recovery using one degraded instance will be studied in two parts. The first part concerns plain vertical or horizontal ($1D$) blurring, and the second part concerns simultaneous vertical and horizontal ($2D$) blurring.

4.1 Plain vertical or horizontal ($1D$) blurring

It is not a trivial problem, to partition the image so that each part contains the whole blurring function information. For most methods, only the horizontal and vertical blurring cases are feasible. Thus, we can solve the cases where linear blurring functions are used i.e. F is $1 \times r$, or $r \times 1$, and the sub-images are two rows or two columns respectively. Columns and rows are one-dimensional which means that a $1D$ polynomial is more appropriate for its representation, and hence a $1D$ polynomial GCD should be used. Algorithm A_1 can be rewritten for $1D$ matrix—polynomial form using our $MSGCD$ or $MAGCD$ algorithm. It handles cases 1, 3 and 4 described in the introduction.

```
Algorithm MA_1 (Modified A_1)
BEGIN
1. Read the $2D$ matrix $B$.
2. Find the $1D$ polynomial of image $B$, namely $b(x)$ using (4).
3. Choose 2 distinct rows (columns) of the matrix which will be our two sub-
   images $B_1$, $B_2$.
4. Find the $1D$ polynomials of $B_1$, $B_2$, namely $b_1(x)$, $b_2(x)$, using (4).
5. Calculate the $1D$ polynomial corresponding to the blurring function as:
       $f(x) = MSGCD(b_1(x), b_2(x))$
       or
       $f(x) = MAGCD(b_1(x), b_2(x))$
6. Calculate the $1D$ polynomial corresponding to the initial image as
       $\mathrm{p}(x) = b(x)/f(x)$
END
```

Noise smoothing is taking place in our $MSGCD$ or $MAGCD$ procedure, as well as the measurement errors. It follows an example illustrating Algorithm MA_1.

Example 1 Let us consider the modified image corresponding to left part of Fig. 2. This image was blurred by applying the $1D$ Gaussian blurring function

$$[0.1393, 0.1428, 0.1450, 0.1457, 0.1450, 0.1428, 0.1393].$$

In addition, noise of order 10^{-3} and measurement error of order 10^{-5} have been added. The modified image has dimensions 97×121 pixels.

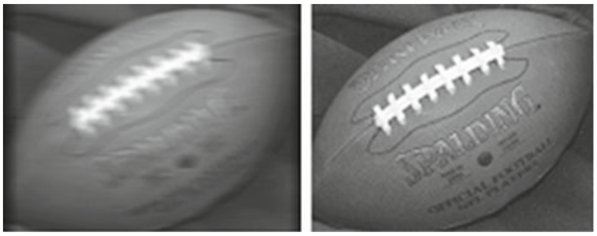


Fig. 2 *Left*: Modified image for Example 1. *Right*: Restored image of Example 1

Let us consider two distinct rows of the modified image. Each row is a sub-image represented by a $1D$ polynomial. By performing the $1D$ polynomial $MSGCD$ or $MAGCD$ algorithm to these two row polynomials, we get the $1D$ polynomial $f(x) = 0.1393 + 0.1428 \cdot x + 0.1450 \cdot x^2 + 0.1457 \cdot x^3 + 0.1450 \cdot x^4 + 0.1428 \cdot x^5 + 0.1393 \cdot x^6$ corresponding to the used blurring function. Finally, by dividing the $1D$ polynomial representing the whole image, with $f(x)$ we get the final $1D$ polynomial which corresponds to the image shown in the right part of Fig. 2.

Note that the dimensions of the resulting image are 97×115 pixels as the filter is 1×7, meaning that it leaves the number of rows of the initial image unchanged, but increases the number of columns by 6 since the larger power of the filter is x^6.

4.2 Simultaneous vertical and horizontal ($2D$) blurring

Plain vertical or horizontal blurring is just a small subset of the whole set of blurring functions used in practice. Here, we develop a new approach handling simultaneous vertical and horizontal blurring, which is much more common. In this case, the image partition is not trivial (see Appendix A.2). In practice, blurring of images, is frequently caused by 3×3 blurring functions [13], satisfying the following properties.

Blurring function properties

1. The blurring function is symmetric with respect to its middle element.
2. The blurring functions sustain unchanged the image illumination.

The first property concerns the blurring functions data. The color contribution weights of the blurring function must be symmetric with respect to the middle element. The second property has a very strong mathematical interpretation. If a blurring function sustain unchanged the image illumination, then the sum of all the color contribution weights of the blurring function equals to 1. A blurring function satisfying the above properties has already been presented in Section 2. Taking into account the blurring functions categories presented in [25], the above properties describe 3×3 spatially variant blurring functions with flexible boundaries. In the sequel, we present the VHD algorithm achieving blind image restoration of the initial image, when the prementioned properties hold, for the blurring function. It handles cases 1, 3 and 4 described in Section 1.

Analysis Steps 1 and 2 are trivial. Steps 3 and 4 will give us the $1D$ polynomial corresponding to the first and last row and the first and last column of the blurring function respectively. This happens because of the symmetry of the blurring function, which is demanded by the first property. Up until now, we have found all the elements of the blurring function except the middle one, as you can see in Fig. 3.

Algorithm *VHD* (Vertical and Horizontal Deblurring)

```
BEGIN
1. Read the $1D$ matrix $B$ and find its $1D$ polynomial, namely $b(x)$ using (4).
2. Find the $1D$ polynomial of the first row of $B$, namely $b_{\text{row-first}}(x)$, the $1D$
   polynomial of the last row of $B$, namely $b_{\text{row-last}}(x)$, the $1D$ polynomial of
   the first column of $B$, namely $b_{\text{column-first}}(x)$ and the $1D$ polynomial of the
   last column of $B$, namely $b_{\text{column-last}}(x)$ using (4).
3. Calculate the $1D$ polynomial corresponding to the first and last row of the
   blurring function as:
       $f_{\text{row}}(x) = MSGCD(b_{\text{row-first}}(x), b_{\text{row-last}}(x))$
       or
       $f_{\text{row}}(x) = MAGCD(b_{\text{row-first}}(x), b_{\text{row-last}}(x))$
4. Calculate the $1D$ polynomial corresponding to the first and last column of
   the blurring function as:
       $f_{\text{column}}(x) = MSGCD(b_{\text{column-first}}(x), b_{\text{column-last}}(x))$
       or
       $f_{\text{column}}(x) = MAGCD(b_{\text{column-first}}(x), b_{\text{column-last}}(x))$
5. Calculate the middle element of the blurring function by subtracting from
   1 the sum of the coefficients of the above $1D$ polynomials, corresponding
   to different elements of the blurring function.
6. Calculate the initial image polynomial as
       $\mathrm{p}(x) = b(x)/f(x))$
END
```

Because of the second property, we know that the sum of all the blurring functions elements equals to 1. This means that if we subtract from 1 the sum of the coefficients of the already found $1D$ row and column polynomials, we

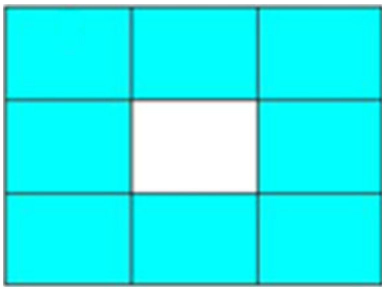


Fig. 3 Blurring function elements found up until step 4 of algorithm *VHD*

will get the unknown middle element (step 5). The whole blurring function $1D$ polynomial $f(x)$ is known. $MSGCD$ or $MAGCD$ algorithm takes care of measurement error and noise artifacts, so we can now safely assume that the following relation holds: $b(x) = f(x) \cdot \mathrm{p}(x)$, where $\mathrm{p}(x)$ is the $1D$ polynomial corresponding to the initial, clean image. So the final step would be to perform the division $b(x)/f(x)$, in order to get $\mathrm{p}(x)$ and hence to restore the image (step 6). Note that throughout the whole algorithm, only $1D$ polynomials were used combined with the $1D$ polynomial $MSGCD$ or $MAGCD$ algorithm. In addition, due to the filter symmetry, the row GCD will be always equal to the column GCD, which means that we have to apply the $MSGCD$ or $MAGCD$ algorithm only once in order to get the whole blurring function, except the middle element.

4.3 Treatment of same segments in a image

If there are same segments in the picture, we have to find them and apply the previous algorithms to two different segments. Alternatively, we can compute the GCD of many or all segments using the algorithm proposed in [24] which handles in an efficient way the Generalized Sylvester matrix. This matrix corresponds to many polynomials and the procedure computes their GCD in one order less ($O(n^3)$ flops instead of $O(n^4)$ flops of the the classical methods for GCD computation of many polynomials). This algorithm can be applied to the first phase of SVD for computing the approximate GCD of polynomials [21], reducing per one order the required complexity.

Example 2 We apply Algorithm VHD on the blurred image which corresponds to the left part of Fig. 4, and was produced by applying noise of order 10^{-3}, measurement errors of order 10^{-5} and the 3×3 Gaussian blurring function

$$\begin{bmatrix} 0.110740742 & 0.111295832 & 0.110740742 \\ 0.111295832 & 0.111853705 & 0.111295832 \\ 0.110740742 & 0.111295832 & 0.110740742 \end{bmatrix}$$

on the initial image. The blurred image has dimensions 45×61 pixels.

Let us consider the first and the last row of the blurred image. Each row is a sub-image represented by a $1D$ polynomial. By applying the $MSGCD$ algorithm on these two row polynomials, using an inner accuracy $\epsilon_t = 10^{-3}$, we get the $1D$ polynomial $0.110740742 + 0.111274383249113 \cdot x + 0.110731093922592 \cdot x^2$ which corresponds to the first and last row of the blurring function. So far, we know that the blurring function is of the following form:

$$\begin{bmatrix} 0.110740742 & 0.111274383249113 & 0.110731093922592 \\ u_1 & u_2 & u_3 \\ 0.110740742 & 0.111274383249113 & 0.110731093922592 \end{bmatrix}$$

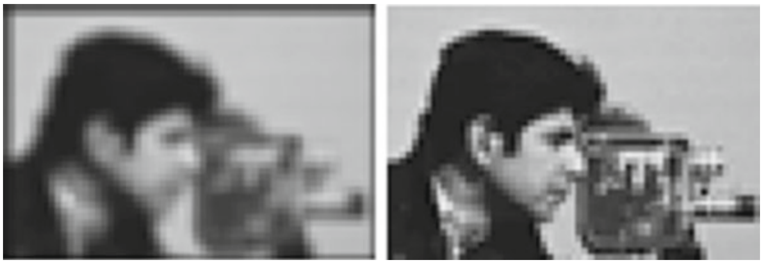


Fig. 4 *Left*: Modified image for Example 2. *Right*: Restored image of Example 2

Let us now consider the first and last column of the modified image. Each column is a sub-image represented by a $1D$ polynomial. By performing $MSGCD$ algorithm on these two column polynomials, using again $\epsilon_t = 10^{-3}$, we get the $1D$ polynomial $0.110740742 + 0.111278478584018 \cdot x + 0.110786105719007 \cdot x^2$ which corresponds to the first and last column of the blurring function. The last operation is not needed, because, due to the filter symmetry, we know that the row GCD will be always equal to the column GCD. So, now the blurring function is of the following form:

$$\begin{bmatrix} 0.110740742 & 0.111274383249113 & 0.110731093922592 \\ 0.111274383249113 & u_2 & 0.111274383249113 \\ 0.110740742 & 0.111274383249113 & 0.110731093922592 \end{bmatrix}$$

We will calculate the middle element of the blurring function by subtracting from 1 the sum of the coefficients of the above and we have that $u_2 = 1 - 0.888167609774086 = 0.111832390225914$. Thus, the used blurring function is the following:

$$\begin{bmatrix} 0.110740742 & 0.111274383249113 & 0.110731093922592 \\ 0.111274383249113 & 0.111832390225914 & 0.111274383249113 \\ 0.110740742 & 0.111274383249113 & 0.110731093922592 \end{bmatrix}$$

which corresponds to a $1D$ polynomial. If we divide the $1D$ polynomial of the blurred image with the latter polynomial, we get the $1D$ polynomial corresponding to the right part of Fig. 4. The dimensions of the resulting image is 43×62 pixels.

Applying the $MAGCD$ algorithm instead of $MSGCD$, using an inner tolerance $\epsilon_t = 10^{-16}$ and a threshold $t = 5 \cdot 10^{-4}$ we get $0.110740742 + 0.111296636539046 \cdot x + 0.110738647919187 \cdot x^2$ as the approximate GCD of the polynomials representing the first and the last row of the blurred image and $0.110740742 + 0.111287008954487 \cdot x + 0.110755082242792 \cdot x^2$ as the approximate GCD of the polynomials representing the first and the last column of the blurred image. Thus, the blurring function is the following:

$$\begin{bmatrix} 0.110740742 & 0.111296636539046 & 0.110738647919187 \\ 0.111296636539046 & 0.111854674005443 & 0.111296636539046 \\ 0.110740742 & 0.111296636539046 & 0.110738647919187 \end{bmatrix}$$

5 Performance of the algorithms

This section concerns the error analysis, the computational and the storage complexity of our methods. Comparisons with other existing GCD based methods are given.

Error analysis of $VHD-MSGCD-MA_1$ algorithms In our methods the GCD (blurring function) is computed through the modified QR factorization. For 2D blurring the following proposition holds.

Proposition 2 *Let $\tilde{P}=\begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$ be the matrix of two different segments P_1 and P_2 of the initial picture containing the whole blurring function information, S^* the modified Sylvester matrix of P_1 and P_2, F the 3×3 symmetric unknown filter, $\hat{F}$ the computed one, E, N the measurement error and the added noise matrix respectively. It holds*

$$\|F-\hat{F}\|_\infty \le 12.5n^{\frac{5}{2}}\cdot\mu\cdot(9(n+1)\|P\|_\infty\|F\|_\infty \\ +9(n+1)\|E\|_\infty\cdot\|F\|_\infty+\|N\|_\infty),$$

where $\|.\|_F$ and $\|.\|_\infty$ the Frobenious and infinity norm of a matrix respectively, Q is an orthogonal matrix computed from Householder's transformations and μ is the machine precision.

Proof From the error analysis of the QR factorization of the modified Sylvester matrix it holds $S^*+E_1=Q\cdot R$, with $\|E_1\|_F\le 12.5n\cdot\mu\cdot\|S^*\|_F$. From norm equivalence, $\|E_1\|_\infty\le 12.5n^{\frac{5}{2}}\cdot\mu\cdot\|S^*\|_\infty$. Our algorithms compute the first three coefficients of the filter F as shown below

$$F=\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ x & x & x \\ x & x & x \end{bmatrix}.$$

Since the filter is symmetric, $f_{21}=f_{32}=f_{23}=f_{12}$ and $f_{31}=f_{33}=f_{11}$. The middle element equals to $1-\sum_{i,j} f_{ij}$, i, j not simultaneously equal to 2.

The last non zero row of the computed upper triangular matrix $R=Q^T(S^*+E_1)$ gives the first three coefficients of F.

The modified Sylvester Matrix S^* is constructed from $(P+E)*F+N$, where $*$ denotes the matrix convolution. The following hold.

$$P*F=\sum_{k=1}^{r}\sum_{l=1}^{r} P_{i-k,j-l}\cdot F_{k,l}=P_{ij}F_{11}+P_{i,j-1}F_{12}+P_{i,j-2}F_{13}$$
$$+P_{i-1,j}F_{21}+P_{i-1,j-1}F_{22}+P_{i-1,j-2}F_{23}+P_{i-2,j}F_{31}+P_{i-2,j-1}F_{32}$$
$$+P_{i-2,j-2}F_{33}\Rightarrow \|S^*_{ij}\|\leq \max\{|P_{ij}|\}\max\{|F_{ij}|\}\ldots\max\{|P_{ij}|\}\max\{|F_{ij}|\}$$
$$=9\max\{|P_{ij}|\}\max\{|F_{ij}|\}.$$

Because the number of non zero elements in every row of S^* is $n+1$, it holds,

$$\|P*F\|_\infty\leq 9(n+1)\max\{|P_{ij}|\}\max\{|F_{ij}|\}.$$

Similarly,

$$\|E*F\|_\infty\leq 9(n+1)\max\{|E_{ij}|\}\max\{|F_{ij}|\}.$$

Thus, $\|F-\hat{F}\|_\infty=\|\hat{E}\|_\infty\leq 12.5n^{\frac{5}{2}}\cdot\mu\cdot\|(P+E)*F+N\|_\infty\leq 12.5n^{\frac{5}{2}}\cdot\mu\cdot(\|(P*F)\|_\infty+\|E*F\|_\infty+\|N\|_\infty)\leq 12.5n^{\frac{5}{2}}\cdot\mu\cdot(9(n+1)\max\{|P_{ij}|\}\max\{|F_{ij}|\}+9(n+1)\max\{|E_{ij}|\}\max\{|F_{ij}|\}+\|N\|_\infty)\leq 12.5n^{\frac{5}{2}}\cdot\mu\cdot(9(n+1)\|P\|_\infty\|F\|\|_\infty+9(n+1)\|E\|_\infty\cdot\|F\|_\infty+\|N\|_\infty)$.

In the presence of round-off errors, we computed the GCD, i.e. the blurring filter, of a slightly permuted matrix $S^*(a,b)+E_1$, corresponding to a slightly perturbed image. □

Remark 1 (MA_1 Algorithm) For the 1D case, the filter is a vector of at most $n+1$ coefficients. The error analysis is similar with the above one, except that the computed error $\hat{E}$ concerns the whole blurring function.

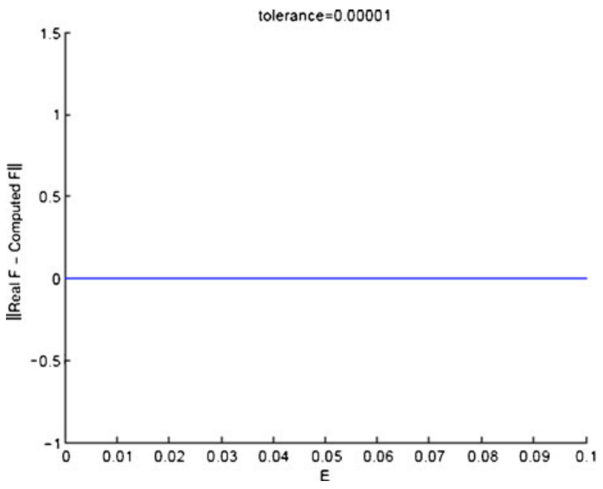
Remark 2 For the case that the computation of the approximate GCD, $MAGCD$ algorithm is required. Its error analysis can be found in [21].

Next, for Example 2, we demonstrate the behaviour of the absolute error of the blurring function $abs_F=\| F-\hat{F}\|_\infty$, in respect of measurement errors, noise and inner tolerance.

In Fig. 5, the abs_F through $MSGCD$ is presented, in respect of the added measurement errors E. As it is shown, the abs_F is of order $O(10^{-9})$, using an inner tolerance of 10^{-5} for measurement errors in the range 10^{-9} to 10^1. The inner tolerance can be reduced downto 10^{-7}. The method computes efficiently the blurring function.

In Fig. 6, the norm of the difference of the real and the computed blurring function is presented, in respect of the added noise N. The $MSGCD$ algorithm

Fig. 5 abs_F applying the $MSGCD$ algorithm in respect of measurement errors using constant inner tolerance



is more sensitive in noise and can handle efficiently values of noise upto order 10^{-2} with inner tolerance of the same order of the added noise. But the $MAGCD$ algorithm can handle even larger noise, upto order of 10^1 using an inner tolerance of order of the epsilon of the machine (Fig. 7).

Computational complexity The time complexity of Algorithms A_1, MA_1 and VHD clearly depends on the step calculating the GCD, as it is the most time consuming step of all. The maximum power of the input polynomials, for Algorithms A_1, MA_1 and VHD, would be $n-1$. This is because we use two

Fig. 6 abs_F applying the $MSGCD$ algorithm in respect of noise using constant inner tolerance

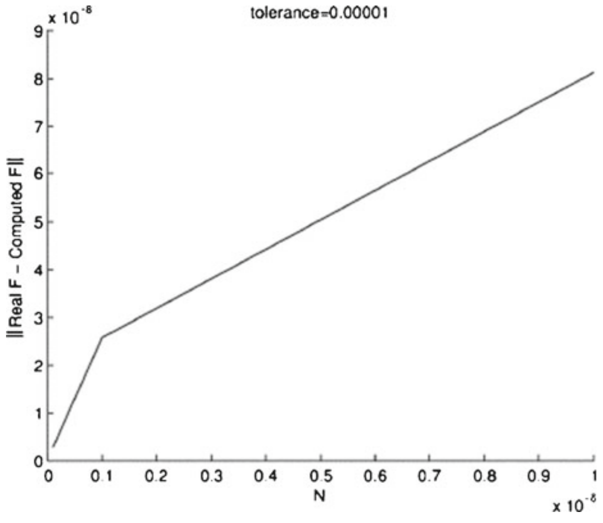
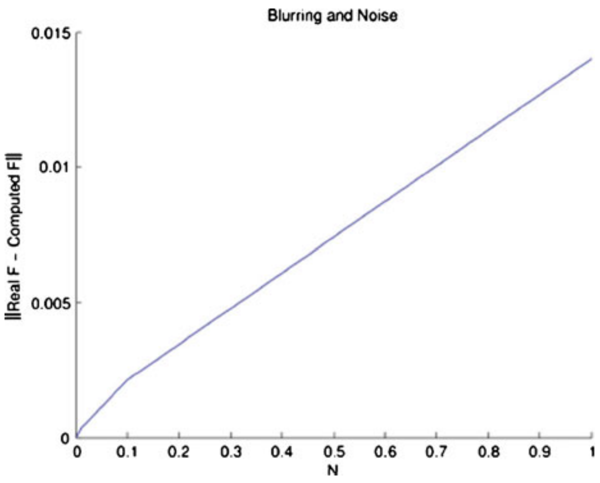


Fig. 7 abs_F applying the $MSGCD$ algorithm in respect of noise using constant inner tolerance



rows or columns of the image matrix at a time, having n elements each. In this case, the computational complexity of Algorithm A_1 decreases even further to $O(n^3)$ flops.

Storage complexity Let P be the initial clean image and B the degraded instance of P. Let us consider, for easier comparisons, that the dimensions of image B is $n \times n$. Algorithms A_1, MA_1 and VHD need $O(n^2)$ space to store the image B_1. Since the dimensions of the blurring function are one or two orders less than that of the degraded images, the space needed for its storage does not increase the order of the total space complexity.

Comparisons In order to quantify the noise level applied in our experiments we use the ratios SNR and $PSNR$. To be more precise, SNR (Signal-to-noise ratio) is defined as the power ratio between the ground-truth signal (meaningful information) and the background noise (unwanted signal):

$$SNR = \frac{P_{\text{groundtruthsignal}}}{P_{\text{noise}}}$$

Table 1 Artifact values of experimental images

Method	Measurement error	Noise	$PSNR\ (dB)$	$SNR\ (dB)$
$VHD - MSGCD, VHD - MAGCD$	10^{-7}	10^{-3}	14.25	0.5
$VHD - MSGCD, VHD - MAGCD$	10^{-7}	10^{-2}	29.40	19.8
$VHD - MSGCD, VHD - MAGCD$	10^{-7}	10^{-1}	48.64	42
$VHD - MAGCD$	10^{-7}	10^{0}	49.3	47.3
$VHD - MAGCD$	10^{-7}	10^{1}	50.4	52.3

Table 2 Methods comparison with respect to noise value

Method	SNR (dB)	$PSNR$ (dB)
El-Khamy [8]	–	30.00
Heindl [15]	45	–
Pillai [26]	45	–
Lin [14]	2.6	–
$VHD-MSGCD, VHD-MAGCD$	42	48.64
$VHD-MAGCD$	52.3	50.4

where P is average power. For dB units we use the following definition:

$$SNR_{dB} = 10 \cdot \log_{10}\left(\frac{P_{\text{groundtruthsignal}}}{P_{\text{noise}}}\right)$$

$PSNR$ (Peak Signal-to-noise ratio) in dB is defined as:

$$PSNR_{dB} = 20 \cdot \log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right)$$

where MAX is the maximum pixel value of the ground-truth signal and MSE is the mean square error between the ground-truth signal and the background noise.

In our experiments, we successfully recovered images characterized by the artifact values, illustrated in Table 1, where our methods are compared with other GCD-based blind image deconvolution techniques taking into account the SNR and the $PSNR$.

In Table 2, are presented the SNR and $PSNR$ values for estimating the applied noise in the pre-mentioned state-of-the-art procedures. A "– " appears when the corresponding value is not provided from the authors. Our methods have the same space, but better time complexity than [8, 14, 15, 26] and work for even better artifact and noise values, handling efficiently disturbed images with the above SNR and PSNR values.

Table 3 Comparison of the proposed image deblurring algorithms

Blurring	Algorithm	Meas. errors	Noise	ϵ_t	Abs. error	Time
1D	$MA_1, MSGCD$	YES: $O(10^{-5})$	NO	10^{-12}	$8.788 \cdot 10^{-15}$	0.113
1D	$MA_1, MSGCD$	NO	YES: $O(10^{-3})$	10^{-3}	$8.455 \cdot 10^{-6}$	0.114
1D	$MA_1, MSGCD$	YES: $O(10^{-5})$	YES: $O(10^{-3})$	10^{-3}	$1.503 \cdot 10^{-5}$	0.159
1D	$MA_1, MAGCD$	YES: $O(10^{-5})$	NO	10^{-16}	$6.866 \cdot 10^{-15}$	409
1D	$MA_1, MAGCD$	NO	YES: $O(10^{-3})$	10^{-16}	$1.640 \cdot 10^{-5}$	419
1D	$MA_1, MAGCD$	YES: $O(10^{-5})$	YES: $O(10^{-3})$	10^{-3}	$4.881 \cdot 10^{-6}$	385
2D	$VHD, MSGCD$	YES: $O(10^{-5})$	NO	10^{-7}	$2.049 \cdot 10^{-9}$	0.168
2D	$VHD, MSGCD$	NO	YES: $O(10^{-3})$	10^{-3}	$7.332 \cdot 10^{-5}$	0.169
2D	$VHD, MSGCD$	YES: $O(10^{-5})$	YES: $O(10^{-3})$	10^{-3}	$1.126 \cdot 10^{-4}$	0.164
2D	$VHD, MAGCD$	NO	YES: $O(10^{-3})$	10^{-16}	$7.536 \cdot 10^{-6}$	337
2D	$VHD, MAGCD$	YES: $O(10^{-5})$	NO	10^{-16}	$9.999 \cdot 10^{-10}$	626
2D	$VHD, MAGCD$	YES: $O(10^{-5})$	YES: $O(10^{-3})$	10^{-16}	$3.204 \cdot 10^{-6}$	488

Table 4 Deblurring of 1D blurred image

Algorithm	Storage space	Complexity	Stability	Mathematical tool
A_1 [15, 26]	$O(2n^2)$	$O(n^4)$	Not stable	DFT Operation
MA_1	$O(2n^2)$	$O(n^4)$	Stable	Sylvester matrices

All algorithms proposed in [15, 26] have not taken into account that it is possible for a blurred image to contain same segments. In that case, the polynomials representing the chosen segments are the same, leading to algorithm failure, since the computed GCD will be one of the initial polynomials and will not represent the blurring function. We have tested our procedures through several examples. In Table 3, a comparison of the numerical implementation of the proposed algorithms is presented. Meas. Errors and Abs. Error stands for Measurement Errors and Absolute Error respectively. For 1D case a 1×3 blurring function was used, whereas for 2D case a 3×3. The examples were executed on a *Core 2 Duo* 2.8 GHz CPU.

As it is shown in Table 3, the computation of the blurring function through the exact GCD is significantly faster than that through the approximate GCD. The main reason for the increase of the required time is that the $MSGCD$ algorithm is implemented numerically whereas the $MAGCD$ in a hybrid way. The symbolic computation of the required determinants is time consuming. Concerning the required precision, the $MAGCD$ algorithm demands a very small ϵ_t, of order of the machine precision. For 1D blurring, the $MSGCD$ algorithm requires a small ϵ_t, of order of $O(10^{-12})$ in case that there are only measurement errors. When noise is inserted, $MSGCD$ needs an internal accuracy ϵ_t, of order of the used perturbation, both for 1D or 2D blurring. Finally, the absolute error computing the blurring function through $VHD - MAGCD$ is less than the corresponding one of $VHD - MSGCD$.

In Table 4, we compare our methods with other GCD based, in respect of their storage space, complexity, stability and mathematical tool that they use.

In Table 5 the performance of the new VHD algorithm is presented.

From Table 4 we see that our MA_1 algorithm attain same complexity with the existing one but it is stable. Finally, as Table 5 suggests, the new proposed VHD algorithm performs in an efficient way image restoration, using one vertical and horizontal blurred image. Alternatively, other denoising algorithms can be fitted in the methods proposed in this manuscript, as a preprocessing step. The complexity of such algorithms varies. If a low-complexity denoising algorithm is used, then the overall complexity of the proposed algorithms remains unaffected.

Table 5 Deblurring of 2D blurred image

Algorithm	Storage space	Complexity	Stability	Mathematical tool
VHD	$O(n^2)$	$O(n^3)$	Stable	Sylvester matrices

An important observation is that we are completely depended on our image and its size. In general, there is not a trivial way to partition the given blurred image in order to achieve restoration. If the image is too small, the solution may be impossible. A counter example of the above statement is presented in the Appendix A.2.

6 Conclusions

In the present paper we studied the blind image deconvolution problem. The kernel of many presented methods is the computation of the GCD of two polynomials. Our approaches are based on structured matrices of small bandwidth and more precisely on Sylvester ones. We exploited the banded structure resulting to fast and stable algorithms computing the blurring function through GCDs, reducing in most cases, per one the order of the required complexity. Our algorithms can handle in an efficient way error measurements or/and noise.

We studied 1D and 2D blurring, having one instance of the image with measurement errors or/and noise. We introduced the use of the approximate GCD of polynomials in order to handle more peculiar circumstances, especially in respect of added noise. MA_1 algorithm uses a variable inner tolerance in order to compute the 1D blurring function through the exact GCD of two polynomials ($MSGCD$ algorithm) or the approximate GCD ($MAGCD$ algorithm). The behavior of $MSGCD$ is better in case that there are only blurring and measurement errors, but its performance is good in case that there is also noise. In this case an inner tolerance of the same order of the added noise is needed. If the noise is big, the use of $MAGCD$ is more appropriate, using an inner tolerance of order of that of the machine precision (10^{-16}).

Our contribution in image deblurring concerns also the case that there is simultaneously vertical and horizontal blurring. We introduced a fast and stable algorithm, named VHD, which computes in an efficient way the 3×3 blurring function, which often appears in practice.

A very interesting future challenge is the study of the case where our blurring functions are of dimensions greater than 3×3, but the blurring function properties, as presented above, still hold. In this case we can apply

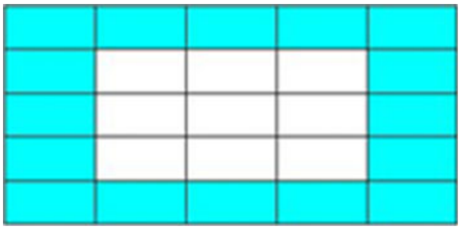


Fig. 8 Structure of a 5×5 blurring function. Only first and last column—row can be found

the first two steps of Algorithm VHD. This way we get the first and the last row as well as the first and the last column of the blurring function, due to the first property, as we can see in Fig. 8. But now, the intermediate elements are more than one, which nullify the use of the second property.

Acknowledgements We thank the reviewers for their valuable comments and suggestions that contributed to a significant improvement of the present paper.

The third author (D.T.) acknowledge financial support from State Scholarships Foundation (IKY).

Appendix

A.1 Implicit numerical example

Below we present an analytical example deblurring an image with measurement errors and noise, computing

i. the exact GCD
ii. the approximate GCD

for achieving the blurring function.

Let

$$P = \begin{bmatrix} 182 & 181 & 178 & 183 & 182 \\ 180 & 178 & 181 & 183 & 178 \\ 182 & 179 & 184 & 183 & 179 \\ 182 & 184 & 184 & 185 & 183 \\ 184 & 185 & 182 & 187 & 184 \end{bmatrix}$$

be the initial clean image. We add measurement errors by adding a matrix $E = 10^{-5} \cdot rand(5,5) \in R^{5\times5}$, where $rand(m,n)$ is a $m \times n$ matrix of pseudorandom values drawn from the standard uniform distribution on the open interval $(0,1)$. Following we deblur the resultant matrix through matrix-convolution using the following filter

$$F = \begin{bmatrix} 0.110740742 & 0.111295832 & 0.110740742 \\ 0.111295832 & 0.111853705 & 0.111295832 \\ 0.110740742 & 0.111295832 & 0.110740742 \end{bmatrix}.$$

The blurred with measurements errors image is

$$PEF = \begin{matrix} 20.154815814224 & 40.299917177208 & 60.011214519184 & 60.120289830559 \\ 40.189176582504 & 80.247024579421 & 120.100090098025 & 120.543598937248 \\ 60.342882536085 & 120.322679527885 & 180.552039072819 & 181.110190210105 \\ 60.343992925059 & 120.656573095111 & 181.553704989714 & 182.333897902157 \\ 60.786955268139 & 121.879716751273 & 182.889267803449 & 183.667227973072 \\ 40.633249310931 & 81.802951618012 & 122.436758167888 & 123.101194572458 \\ 20.376296728058 & 40.965471423939 & 61.120841603251 & 61.451399381861 \end{matrix}$$

$$\begin{matrix} 60.233805814161 & 40.521398290998 & 20.154815511449 \\ 120.658794386261 & 80.800727492742 & 39.967694809155 \\ 181.224266745628 & 120.986006278080 & 59.788067509908 \\ 182.224264297834 & 121.319341663579 & 59.899363691746 \\ 183.447969322169 & 122.433422462732 & 60.566028756925 \\ 122.882502532412 & 82.247580242003 & 40.743990789386 \\ 61.343434319558 & 41.186953963893 & 20.376297617639 \end{matrix}.$$

The size of the blurred image is increased because of the blurring-convolution to 7×7. Finally we add noise $N = 10^{-3} \cdot rand(7,7) \in R^{5\times 5}$. The blurred with measurement errors and noise image is

$$B = \begin{matrix} 20.155726485951 & 40.300866316643 & 60.012038803599 & 60.120949281114 \\ 40.189899288988 & 80.247797365802 & 120.100871633383 & 120.544464166928 \\ 60.342947992218 & 120.323495943412 & 180.552815034439 & 181.110571316990 \\ 60.344172822592 & 120.656719009858 & 181.554235326181 & 182.334383301477 \\ 60.787717105594 & 121.879790532712 & 182.890170434506 & 183.667705157687 \\ 40.633950835402 & 81.803663655457 & 122.437157034334 & 123.102051086887 \\ 20.377104200105 & 40.965675181228 & 61.121143170382 & 61.452220309827 \end{matrix}$$

$$\begin{matrix} 60.234540211441 & 40.522345322323 & 20.155448548154 \\ 120.659430651095 & 80.801545276122 & 39.967814881173 \\ 181.224665322727 & 120.986161159213 & 59.788121314001 \\ 182.225189548709 & 121.320113079081 & 59.900295418986 \\ 183.448646084739 & 122.433676838906 & 60.566492094675 \\ 122.882709440636 & 82.248189088586 & 40.744153442484 \\ 61.344218379459 & 41.187510372286 & 20.376818458681 \end{matrix}$$

Knowing the image B, our scope is to find the unknown to us filter F. We select two segments which contain the whole blurring information, e.g. the first and the last row: $a(x) = 20.15572648595144 + 40.30086631664272x + 60.01203880359884x^2 + 60.12094928111444x^3 + 60.23454021144070x^4 + 40.52234532232279x^5 + 20.15544854815354x^6$ and $b(x) = 20.37710420010474 + 40.96567518122846x + 61.12114317038242x^2 + 61.45222030982712x^3 + 61.34421837945899x^4 + 41.18751037228594x^5 + 20.37681845868128x^6$.

We construct the modified Sylvester matrix S^* of $a(x), b(x)$.

i. We apply the modified QR factorization using an inner tolerance $\epsilon_t = 10^{-3}$. The last non zero row of the upper triangular resultant from QR matrix is:

$$[0.11074074200000 \;\; 0.11135754386367 \;\; 0.11074866970442]$$

which is the first and the last row of the filter.
Selecting the first and the last column of B, constructing again their modified Sylvester matrix and applying once again the modified QR factorization we achieve their GCD: [0.11074074200000 0.11146644291393 0.11109148267456].
Thus we have computed the 8 of 9 elements of the filter:

$$F = \begin{bmatrix} 0.110740742 & 0.11135754386367 & 0.11074866970442 \\ 0.11146644291393 & * & 0.11146644291393 \\ 0.110740742 & 0.11135754386367 & 0.11074866970442 \end{bmatrix}.$$

The $F_{2,2}$ element is achieved by subtracting the sum of the other element from 1: $F_{2,2} = 0.11137320303596$ and thus the filter is computed with an absolute error $5.428449760527508 \cdot 10^{-4}$, which is of the same order of the added noise.

ii. We will compute the approximate GCD of $a(x), b(x)$. Having construct the modified Sylvester matrix S^* of $a(x), b(x)$, we apply the singular value decomposition using the modified QR factorization in its first phase. Its singular values are: $\sigma = [360.6765710641447$ 209.7177977007718 $83.800263426691129.5264138040203$ 16.9507264946333 12.9731959346732 0.3176825247179 0.2981359254880 0.0478187888456 0.0364952003963 0.0002740576638 $0.0000525698564]$ The inner tolerance ϵ_t we use for computing the approximate GCD is 10^{-16}. Using a threshold $t = 10^{-4}$ the right "near nullspace" of S^* is:

$$M = \begin{bmatrix} 0.39305720437705 & 0.12011339400492 \\ -0.09332771895450 & -0.39968383810498 \\ -0.29806168388289 & 0.28207686001740 \\ 0.39170657448694 & 0.11445025359782 \\ -0.09562861690373 & -0.39452725049458 \\ -0.29437899100794 & 0.28036304160719 \\ 0.39025683551845 & 0.11317276662630 \\ -0.09660624884324 & -0.39369530867918 \\ -0.29438785148953 & 0.28080596459294 \\ 0.39248731123472 & 0.11403867342849 \\ -0.09889395405157 & -0.39712020582137 \\ -0.29426801496527 & 0.28555010472527 \end{bmatrix}.$$

We construct matrices M_1 and M_2 by deleting the last and the first row of M respectively and following we construct the matrix pencil $Z(s) =$

$s \cdot M_1 - M_2$. Next, we compute all determinants of $Z(s)$ which are polynomials of degree 2 and we construct the matrix B containing the coefficients of the previous polynomials. We apply the SVD to B, $B = U\sigma V^T$ and its singular values are $\sigma = [1.581189085948278\ 0.003302816278459\ 0.003021007759667]$. The maximum singular value is the first one and thus the first column of V gives the approximate GCD:

$$v(:,1) = \begin{bmatrix} 0.110740742000000 \\ 0.111097327117126 \\ 0.110688204308170 \end{bmatrix}.$$

This is the first and the last row of the filter.
Selecting the first and the last column of B, constructing again their modified Sylvester matrix and applying once again the previous procedure, we achieve their approximate GCD:

$$v(:,1) = \begin{bmatrix} 0.1107407420000000.1110973271171260.110688204308178 \end{bmatrix}.$$

Thus we have computed the 8 of 9 elements of the filter:

$$F = \begin{bmatrix} 0.110740742 & 0.111097327117126 & 0.111097327117126 \\ 0.111097327117126 & * & 0.111097327117126 \\ 0.110740742 & 0.111097327117126 & 0.111097327117126 \end{bmatrix}.$$

The (2, 2) element is achieved by subtracting the some of the other element from 1: $F_{2,2} = 0.111934553297244$ and thus the filter is computed with an absolute error $6.194058675465041 \cdot 10^{-4}$, which is of the same order of the added noise.

A.2 Difficulties in partioning a small image

As already mentioned, in general, there is not a trivial way to partition the given blurred image in order to achieve restoration. In fact, if the image is too small, the solution may be impossible. To explain the above statement, let us consider that the initial image A is the following 5×5 matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

and the blurring function F is the following 3×3 matrix:

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Then, the blurred image 7×7 matrix B is presented in Table 6. We know that the first and last line ($[f_{11}, f_{12}, f_{13}]$ and $[f_{31}, f_{32}, f_{33}]$) of the blurring

Table 6 Mathematical view of blurred image B

$a_{11} \cdot f_{11}$	$a_{11} \cdot f_{12}+$ $a_{12} \cdot f_{11}$	$a_{11} \cdot f_{13}+$ $a_{12} \cdot f_{12}+$ $a_{13} \cdot f_{11}$	$a_{12} \cdot f_{13}+$ $a_{13} \cdot f_{12}+$ $a_{14} \cdot f_{11}$	$a_{13} \cdot f_{13}+$ $a_{14} \cdot f_{12}+$ $a_{15} \cdot f_{11}$	$a_{14} \cdot f_{13}+$ $a_{15} \cdot f_{12}$	$a_{15} \cdot f_{13}$
$a_{11} \cdot f_{21}+$ $a_{21} \cdot f_{11}$	$a_{11} \cdot f_{22}+$ $a_{12} \cdot f_{21}+$ $a_{21} \cdot f_{12}+$ $a_{22} \cdot f_{11}$	$a_{11} \cdot f_{23}+$ $a_{12} \cdot f_{22}+$ $a_{13} \cdot f_{21}+$ $a_{21} \cdot f_{13}+$ $a_{22} \cdot f_{12}+$ $a_{23} \cdot f_{11}$	$a_{12} \cdot f_{23}+$ $a_{13} \cdot f_{22}+$ $a_{14} \cdot f_{21}+$ $a_{22} \cdot f_{13}+$ $a_{23} \cdot f_{12}+$ $a_{24} \cdot f_{11}$	$a_{13} \cdot f_{23}+$ $a_{14} \cdot f_{22}+$ $a_{15} \cdot f_{21}+$ $a_{23} \cdot f_{13}+$ $a_{24} \cdot f_{12}+$ $a_{25} \cdot f_{11}$	$a_{14} \cdot f_{23}+$ $a_{15} \cdot f_{22}+$ $a_{24} \cdot f_{13}+$ $a_{25} \cdot f_{12}$	$a_{15} \cdot f_{23}+$ $a_{25} \cdot f_{13}$
$a_{11} \cdot f_{31}+$ $a_{21} \cdot f_{21}+$ $a_{31} \cdot f_{11}$	$a_{11} \cdot f_{32}+$ $a_{12} \cdot f_{31}+$ $a_{21} \cdot f_{22}+$ $a_{22} \cdot f_{21}+$ $a_{31} \cdot f_{12}+$ $a_{32} \cdot f_{11}$	$a_{11} \cdot f_{33}+$ $a_{12} \cdot f_{32}+$ $a_{13} \cdot f_{31}+$ $a_{21} \cdot f_{23}+$ $a_{22} \cdot f_{22}+$ $a_{23} \cdot f_{21}+$ $a_{31} \cdot f_{13}+$ $a_{32} \cdot f_{12}+$ $a_{33} \cdot f_{11}$	$a_{12} \cdot f_{33}+$ $a_{13} \cdot f_{32}+$ $a_{14} \cdot f_{31}+$ $a_{22} \cdot f_{23}+$ $a_{23} \cdot f_{22}+$ $a_{24} \cdot f_{21}+$ $a_{32} \cdot f_{13}+$ $a_{33} \cdot f_{12}+$ $a_{34} \cdot f_{11}$	$a_{13} \cdot f_{33}+$ $a_{14} \cdot f_{32}+$ $a_{15} \cdot f_{31}+$ $a_{23} \cdot f_{23}+$ $a_{24} \cdot f_{22}+$ $a_{25} \cdot f_{21}+$ $a_{33} \cdot f_{13}+$ $a_{34} \cdot f_{12}+$ $a_{35} \cdot f_{11}$	$a_{14} \cdot f_{33}+$ $a_{15} \cdot f_{32}+$ $a_{24} \cdot f_{23}+$ $a_{25} \cdot f_{22}+$ $a_{34} \cdot f_{13}+$ $a_{35} \cdot f_{12}$	$a_{15} \cdot f_{33}+$ $a_{25} \cdot f_{23}+$ $a_{35} \cdot f_{13}$
$a_{21} \cdot f_{31}+$ $a_{31} \cdot f_{21}+$ $a_{41} \cdot f_{11}$	$a_{21} \cdot f_{32}+$ $a_{22} \cdot f_{31}+$ $a_{31} \cdot f_{22}+$ $a_{32} \cdot f_{21}+$ $a_{41} \cdot f_{12}+$ $a_{42} \cdot f_{11}$	$a_{21} \cdot f_{33}+$ $a_{22} \cdot f_{32}+$ $a_{23} \cdot f_{31}+$ $a_{31} \cdot f_{23}+$ $a_{32} \cdot f_{22}+$ $a_{33} \cdot f_{21}+$ $a_{41} \cdot f_{13}+$ $a_{42} \cdot f_{12}+$ $a_{43} \cdot f_{11}$	$a_{22} \cdot f_{33}+$ $a_{23} \cdot f_{32}+$ $a_{24} \cdot f_{31}+$ $a_{32} \cdot f_{23}+$ $a_{33} \cdot f_{22}+$ $a_{34} \cdot f_{21}+$ $a_{42} \cdot f_{13}+$ $a_{43} \cdot f_{12}+$ $a_{44} \cdot f_{11}$	$a_{23} \cdot f_{33}+$ $a_{24} \cdot f_{32}+$ $a_{25} \cdot f_{31}+$ $a_{33} \cdot f_{23}+$ $a_{34} \cdot f_{22}+$ $a_{35} \cdot f_{21}+$ $a_{43} \cdot f_{13}+$ $a_{44} \cdot f_{12}+$ $a_{45} \cdot f_{11}$	$a_{24} \cdot f_{33}+$ $a_{25} \cdot f_{32}+$ $a_{34} \cdot f_{23}+$ $a_{35} \cdot f_{22}+$ $a_{44} \cdot f_{13}+$ $a_{45} \cdot f_{12}$	$a_{25} \cdot f_{33}+$ $a_{35} \cdot f_{23}+$ $a_{45} \cdot f_{13}$
$a_{31} \cdot f_{31}+$ $a_{41} \cdot f_{21}+$ $a_{51} \cdot f_{11}$	$a_{31} \cdot f_{32}+$ $a_{32} \cdot f_{31}+$ $a_{41} \cdot f_{22}+$ $a_{42} \cdot f_{21}+$ $a_{51} \cdot f_{12}+$ $a_{52} \cdot f_{11}$	$a_{31} \cdot f_{33}+$ $a_{32} \cdot f_{32}+$ $a_{33} \cdot f_{31}+$ $a_{41} \cdot f_{23}+$ $a_{42} \cdot f_{22}+$ $a_{43} \cdot f_{21}+$ $a_{51} \cdot f_{13}+$ $a_{52} \cdot f_{12}+$ $a_{53} \cdot f_{11}$	$a_{32} \cdot f_{33}+$ $a_{33} \cdot f_{22}+$ $a_{34} \cdot f_{31}+$ $a_{42} \cdot f_{23}+$ $a_{43} \cdot f_{22}+$ $a_{44} \cdot f_{21}+$ $a_{52} \cdot f_{13}+$ $a_{53} \cdot f_{12}+$ $a_{54} \cdot f_{11}$	$a_{33} \cdot f_{33}+$ $a_{34} \cdot f_{22}+$ $a_{35} \cdot f_{31}+$ $a_{43} \cdot f_{23}+$ $a_{44} \cdot f_{22}+$ $a_{45} \cdot f_{21}+$ $a_{53} \cdot f_{13}+$ $a_{54} \cdot f_{12}+$ $a_{55} \cdot f_{11}$	$a_{34} \cdot f_{33}+$ $a_{35} \cdot f_{32}+$ $a_{44} \cdot f_{23}+$ $a_{45} \cdot f_{22}+$ $a_{54} \cdot f_{13}+$ $a_{55} \cdot f_{12}$	$a_{35} \cdot f_{33}+$ $a_{45} \cdot f_{23}+$ $a_{55} \cdot f_{13}$
$a_{41} \cdot f_{31}+$ $a_{51} \cdot f_{21}$	$a_{41} \cdot f_{32}+$ $a_{42} \cdot f_{31}+$ $a_{51} \cdot f_{22}+$ $a_{52} \cdot f_{21}$	$a_{41} \cdot f_{33}+$ $a_{42} \cdot f_{32}+$ $a_{43} \cdot f_{31}+$ $a_{51} \cdot f_{23}+$ $a_{52} \cdot f_{22}+$ $a_{53} \cdot f_{21}$	$a_{42} \cdot f_{33}+$ $a_{43} \cdot f_{32}+$ $a_{44} \cdot f_{31}+$ $a_{52} \cdot f_{23}+$ $a_{53} \cdot f_{22}+$ $a_{54} \cdot f_{21}$	$a_{43} \cdot f_{33}+$ $a_{44} \cdot f_{32}+$ $a_{45} \cdot f_{31}+$ $a_{53} \cdot f_{23}+$ $a_{54} \cdot f_{22}+$ $a_{55} \cdot f_{21}$	$a_{44} \cdot f_{33}+$ $a_{45} \cdot f_{32}+$ $a_{54} \cdot f_{23}+$ $a_{55} \cdot f_{22}$	$a_{45} \cdot f_{33}+$ $a_{55} \cdot f_{23}$
$a_{51} \cdot f_{31}$	$a_{51} \cdot f_{32}+$ $a_{52} \cdot f_{31}$	$a_{51} \cdot f_{33}+$ $a_{52} \cdot f_{32}+$ $a_{53} \cdot f_{31}$	$a_{52} \cdot f_{33}+$ $a_{53} \cdot f_{32}+$ $a_{54} \cdot f_{31}$	$a_{53} \cdot f_{33}+$ $a_{54} \cdot f_{32}+$ $a_{55} \cdot f_{31}$	$a_{54} \cdot f_{33}+$ $a_{55} \cdot f_{32}$	$a_{55} \cdot f_{33}$

function are equal. Taking into account the previous observation, we notice that the first and last line of the blurred image are completely symmetric as far as the initial image coefficients are concerned ($[a_{11}, a_{12}, a_{13}, a_{14}, a_{15}]$ and

$[a_{51}, a_{52}, a_{53}, a_{54}, a_{55}]$). This means that, the great common devisor of the pre-mentioned lines is vector $[f_{11}, f_{12}, f_{13}]$ which is equal to vector $[f_{31}, f_{32}, f_{33}]$. The same thing holds for the first and last columns of the blurred image.

It is pretty obvious, that the above does not hold for any other pair of subimages of the blurred image. There are not any other pair of subimages, that contain the whole blurring function in full symmetry with the initial image coefficients a_{ij}, in order to produce the blurring function as their greater common devisor. That is the reason why for small images there may be no solution to our problem and why image partitioning is not trivial in general.

References

1. Beck, A., Eldar, Y.C.: Regularization in regression with bounded noise: a Chebyshev center approach. SIAM J. Matrix Anal. Appl. **29**, 606–625 (2007)
2. Ayers, G.R., Dainty, J.C.: Iterative blind deconvolution method and its applications. Opt. Lett. **13**, 547–549 (1988)
3. Barnett, S.: Greatest common divisor from generalized Sylvester resultant matrices. Linear Multilinear Algebra **8**, 271–279 (1980)
4. Chatterjee, P.: Denoising using the K-SVD method. Course Web Pages, EE 264: Image Processing and Reconstruction, pp. 1–12 (2007)
5. Cooley, J.W., James W., Tukey W.J.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**, 297–301 (1965)
6. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising with block-matching and 3D filtering. In: Proceedings of Electronic Imaging, Proc. SPIE 6064, No 6064A-30, vol. 14. San Jose, USA (2006)
7. Datta, N.B.: Numerical Linear Algebra and Applications, 2nd edn. SIAM, Philadelphia (2010)
8. El-Khamy, S.E., Hadhood, M.M., Dessouky, M.I., Salam, B.M., Abd El-Samie, F.E.: A Greatest common divisor approach to blind super-resolution reconstruction of images. J. Mod. Opt. **53**, 1027–1039 (2006)
9. Fiori, S.: A fast fixed-point neural blind deconvolution algorithm. IEEE Trans. Neural Netw. **15**, 455–459 (2004)
10. Foi, A., Katkovnik, V., Egiazarian, K.: Pointwise shape-adaptive DCT for high-quality deblocking of compressed color images. IEEE Trans. Image Process. **16**, 1057–7149 (2007)
11. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proc. IEEE **93**, 216–231 (2005)
12. Gentleman, W.M., Sande, G.: Fast fourier transforms-for fun and profit. Proc. AFIPS **29**, 563–578 (1966)
13. Gonzalez, R., Woods, R.: Digital Image Processing. Addison-Wesley, New York (1992)
14. He, L., Marquina, A., Osher, S.J.: Blind deconvolution using TV regularization and bregman iteration. Int. J. Imag. Syst. Technol. **15**, 74–83 (2005)
15. Heindl, A.R.: Fourier transform, polynomial GCD, and image restoration. Master Thesis for Clemson University, Department of Mathematical Sciences (2005)
16. Heideman, M.T., Burrus, C.S.: On the number of multiplications necessary to compute a length-2n DFT. IEEE Trans. Acoust. Speech. Signal Process. **34**, 91–95 (1986)
17. Kervrann, C., Boulanger, J.: Local adaptivity to variable smoothness for exemplar-based image regularization and representation. Int. J. Comput. Vis. **79**, 45–69 (2006)
18. Kundur, D., Hatzinakos, D.: Blind image deconvolution. IEEE Signal Process. Mag. **13**(3), 43–64 (1996)
19. Kundur, D., Hatzinakos, D.: A novel blind deconvolution scheme for image restoration using recursive filtering. IEEE Trans. Signal Process. **46**, 375–390 (1998)
20. Mathis, H., Douglas, C.S.: Bussgang blind deconvolution for impulsive signals. IEEE Trans. Signal Process. **51**, 1905–1915 (2003)

21. Karcanias, N., Mitrouli, M., Triantafyllou, D.: Matrix pencil methodologies for computing the greatest common divisor of polynomials: hybrid algorithms and their performance. Int. J. Control **79**, 1447–1461 (2006)
22. Kim, Y., Kim, S., Kim, M.: The derivation of a new blind equalization algorithm. ETRI J. **18**, 53–60 (1996)
23. Lundy, T., Van Buskirk, J.: A new matrix approach to real FFTs and convolutions of length 2k. Computing **80**, 23–45 (2007)
24. Triantafyllou, D., Mitrouli, M.: On rank and null space computation of the generalized Sylvester matrix. Numer. Algorithms **54**, 297–324 (2009)
25. Nagy, G.J., Palmer, K., Perrone, L.: Iterative methods for image deblurring: a Matlab object-oriented approach. Numer. Algorithms **36**, 73–93 (2003)
26. Pillai, S.U., Liang, B.: Blind image deconvolution using a robust GCD approach. IEEE Trans. Image Process. **8**, 295–301 (1999)
27. Abad, J.O., Morigi, S., Reichel, L., Sgallari, F.: Alternating Krylov subspace image restoration methods. J. Comput. Appl. Math. **236**(8), 2049–2062 (2012)
28. Schatzman, C.J.: Accuracy of the discrete fourier transform and the fast fourier transform. SIAM J. Sci. Comput. **17**, 1150–1166 (1996)
29. van der Veen, A.J., Paulraj, A.: An analytical constant modulus algorithm. IEEE Trans. Signal Process. **44**, 1–19 (1999)
30. Vermon, D.: Machine Vision. Prentice-Hall, Englewood Cliffs (1991)
31. Chan, F.T., Wong, C.K.: Total variation blind deconvolution. IEEE Trans. Image Process. **7**, 370–375 (1998)