Blind Image Deconvolution via Fast Approximate GCD*

Zijia Li
Key Laboratory of Mathematics Mechanization
AMSS, Beijing 100190, China
lizijia@amss.ac.cn

Zhengfeng Yang
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University, Shanghai 200062, China
zfyang@sei.ecnu.edu.cn

Lihong Zhi
Key Laboratory of Mathematics Mechanization
AMSS, Beijing 100190, China
lzhi@mmrc.iss.ac.cn, http://www.mmrc.iss.ac.cn/~lzhi

ABSTRACT

The problem of blind image deconvolution can be solved by computing approximate greatest common divisors (GCD) of polynomials. The bivariate polynomials corresponding to the z-transforms of several blurred images have an approximate GCD corresponding to the z-transform of the original image. Since blurring functions as cofactors have very low degree in general, this GCD will be of high degree. On the other hand, if we only have one blurred image and want to identify the original scene, the blurred image can be partitioned such that each part completely contains the blurring function, hence the blurring function becomes the GCD which is of low degree. Therefore, we design a specialized algorithm for computing GCDs of polynomials to recover true images in two different cases. The new algorithm is based on the fast GCD algorithm for univariate polynomials and the Fast Fourier Transform (FFT) algorithm. The complexity of our specialized algorithm for identifying both the true image and the blurring functions from blurred images of size $n \times n$ is $O(n^2 \log(n))$ in the case of blurring functions of very low degree. The algorithm has been implemented in Maple and can extract true images of hundreds by hundreds pixel images from blurred images in a few seconds.

Categories and Subject Descriptors: G.1.6 [Numerical Analysis]: Approximation; I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; I.4.4 [Image Processing and Computer Vision]: Restoration
General Terms: algorithms, experimentation
Keywords: blind image deconvolution, approximate GCD, Bezout matrix, Sylvester matrix, Fast Fourier Transform.

1. INTRODUCTION

A grey image can be represented by a matrix whose dimension is equal to the size of the image, and a color image includes three parameters (Red, Green, Blue), and can be represented by three matrices respectively. Suppose the original image matrix is P, the distorted and polluted image matrix is F. The blurring function matrix and the additive noise matrix of F are U and N respectively. Then we have the following relation between the true image matrix and the distorted image matrix:

$$F = P * U + N. \tag{1}$$

The measurement for noise in an image is the signal-to-noise ratio, or SNR. In [17], it is defined as:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{\sigma^2_{P*U}}{\sigma^2_N} \right), \tag{2}$$

where σ^2_{P*U} and σ^2_N are the variances of the blurred image without noise and the additive noise respectively.

Blind image deconvolution is the process of identifying both the true image and the blurring function from the blurred images. Even if we assume the blurred picture is noise-free, which means N is a zero matrix or a matrix with very little entries (for example SNR ≥ 50), it is still very difficult and expensive to obtain P and U from F. For example, the complexity of algorithms based on the approximate factorization of polynomials for blind deconvolution is $O(n^8)$ for an $n \times n$ image [16, 28]. However, if we know multiple blurred versions of the true image, or the blurred image can be partitioned such that each part completely contains the same blurring function, blind image deconvolution can be transformed to computing approximate GCDs of polynomials by using the z-transforms [35].

Definition 1. *A two-dimensional z-transform maps the elements of an $m \times n$ matrix P to the coefficients of a bivariate polynomial $p(x, y)$:*

$$p(x, y) = \mathbf{x}^T \cdot P \cdot \mathbf{y}, \tag{3}$$

where $\mathbf{x} = [1, x, x^2, \ldots, x^{m-1}]^T$, $\mathbf{y} = [1, y, y^2, \ldots, y^{n-1}]^T$.

By the two-dimensional z-transform, we map the elements of matrices to coefficients of bivariate polynomials. Hence, (1) is transferred into

$$f(x, y) = p(x, y)\, u(x, y) + n(x, y), \tag{4}$$

where f, p, u, n are z-transforms of F, P, U, N respectively. Such a model is applicable in all scenarios where the distortion can be modeled as a linear filter acting on the original image. For example, camera motion and intermediate

*Zijia Li and Lihong Zhi are supported by the Chinese National Natural Science Foundation under Grants 60821002/F02, 60911130369 and 10871194. Zhengfeng Yang is supported by the Chinese National Natural Science Foundation under Grant 10901055 and Shanghai Natural Science Foundation under Grant 09ZR1408800.

medium in satellite photography can be modeled as (1) and (4) [7].

Suppose the original image matrix is P, matrices of two distorted images are F_1, F_2. The blurring functions and additive noises of F_1, F_2 are U, V and N_1, N_2 respectively. So we have the following relations between original images and distorted images:

$$F_1 = P * U + N_1, \quad F_2 = P * V + N_2. \tag{5}$$

By the two-dimensional z-transform, (5) can be transferred into polynomial forms:

$$\left.\begin{array}{l} f_1(x,y) = p(x,y)u(x,y) + n_1(x,y), \\ f_2(x,y) = p(x,y)v(x,y) + n_2(x,y), \end{array}\right\} \tag{6}$$

where $f_1, p, u, n_1, f_2, v, n_2$ are z-transforms of F_1, P, U, N_1, F_2, V, N_2 respectively. We can compute the approximate GCD of bivariate polynomials $f_1(x,y)$ and $f_2(x,y)$ to reconstruct the true image. In practice, $\deg(u)$ and $\deg(v)$ are very low compared with $\deg(p)$. Hence, the approximate GCD $p(x,y)$ of $f_1(x,y)$ and $f_2(x,y)$ will be of high degree.

If there is only one available blurred RGB image, we assume that three channels have the same blurring function. As well, we can get the matrix form corresponding to each channel

$$F_1 = P_1 * U + N_1, \quad F_2 = P_2 * U + N_2, \quad F_3 = P_3 * U + N_3,$$

where F_1, F_2, F_3 are the blurring image matrices of each channel, P_1, P_2, P_3 are original image matrices respectively, U is the blurring filter matrix and N_1, N_2, N_3 are additive noises. By z-transforms, we have:

$$\left.\begin{array}{l} f_1(x,y) = p_1(x,y)u(x,y) + n_1(x,y), \\ f_2(x,y) = p_2(x,y)u(x,y) + n_2(x,y), \\ f_3(x,y) = p_3(x,y)u(x,y) + n_3(x,y). \end{array}\right\} \tag{7}$$

Hence the blurring function $u(x,y)$ is now the approximate GCD of f_1, f_2, f_3 which is of very low degree.

In [35], they described a fast algorithm for computing the approximate GCD of bivariate polynomials based on the Sylvester-type GCD algorithm for univariate polynomials and the Discrete Fourier Transform (DFT) algorithm. Their idea is to sample polynomials f_1 and f_2 in one variable at the DFT points on the unit circle:

$$x = e^{-\frac{2k\pi \mathbf{i}}{m}}, \; k = 0, 1, \ldots, m-1,$$

and

$$y = e^{-\frac{2l\pi \mathbf{i}}{n}}, \; l = 0, 1, \ldots, n-1,$$

where $\mathbf{i} = \sqrt{-1}$, m and n are the number of rows and columns of the matrix of a blurred image. The GCD of the resulting univariate polynomials is found by using the Sylvester-type GCD method. These GCD polynomials are again sampled at the DFT points, eventually generating two matrices that are scaled versions of the two dimensional DFT of the original image. They obtained an approximation of $p(x,y)$ by equalizing these two matrices and taking the two-dimensional inverse DFT. They assumed that DFT points are free of these points which will change the degree of the GCD or be the zeros of the GCD or the cofactors [35]. For images of size $n \times n$, the complexity of their algorithm is $O(n^4)$, which is a substantial saving compared with the direct generalized Sylvester-type procedures for bivariate polynomials [14, 23, 42], which require about $O(n^6)$ operations.

The computation of the approximate GCD of univariate polynomials has been extensively studied with various approaches including the Euclidean method on the polynomial remainder sequence [4, 20, 32, 36], the singular value decomposition or QR decomposition of the Sylvester matrix [8, 9, 13], Padé approximation [34], iterative method [39] and other optimization strategies [25, 26, 27, 31]. Fast GCD algorithms for univariate polynomials based on displacement structure of the Sylvester matrix have also been proposed in [5, 29, 30, 41, 43].

It is well known that the Bezout matrix can also be used to compute GCDs of univariate polynomials [1, 2, 3, 5, 10, 11, 15, 38]. Compared with the Sylvester matrix, the Bezout matrix has smaller size. For example, the size of the Sylvester matrix of two polynomials with the same degree n is $2n \times 2n$, while the size of the Bezout matrix is $n \times n$. Moreover, since the degree of the blurring function is usually very low, the GCD polynomial $p(x,y)$ has high degree. We see below that, in this case (Example 1 and Example 2 in Section 4), it is more appealing to use the Bezout matrix. Because only a very small submatrix of the Bezout matrix is constructed and used for computing approximate GCDs.

On the other hand, although the Bezout matrix has smaller size, entries of the Bezout matrix are bilinear in the coefficients of polynomials. For polynomials of degree m, the cost to generate a full size Bezout matrix is already $O(m^2)$. Hence, when the degree of the cofactor is high, for instance, in recovering true images from one blurred grey or RGB images (Example 3 and Example 4 in Section 4), it becomes more efficient to use the Sylvester matrix to compute the GCD of univariate polynomials.

Section 2 is devoted to recall some notations and well-known facts about the Bezout matrix and the Sylvester matrix. We also present a fast algorithm for computing the approximate GCD of univariate polynomials. In Section 3, we describe an algorithm for computing the approximate GCD of bivariate polynomials based on the univariate GCD computation and FFT. We prove that the complexity of our new algorithm for blind image deconvolution of size $n \times n$ is $O(n^2 \log(n))$ in the case of blurring functions of very low degree. Experiments are done in Section 4. Some remarks for future work will be given in Section 5.

2. AN APPROXIMATE UNIVARIATE POLYNOMIAL GCD ALGORITHM

Suppose we are given two univariate polynomials $f_1, f_2 \in \mathbb{C}[x] \backslash \{0\}$ with $\deg(f_1) = m$ and $\deg(f_2) = n$, assume $m \geq n$,

$$\left.\begin{array}{ll} f_1 = u_m x^m + u_{m-1}x^{m-1} + \cdots + u_1 x + u_0, & u_m \neq 0, \\ f_2 = v_n x^n + v_{n-1}x^{n-1} + \cdots + v_1 x + v_0, & v_n \neq 0. \end{array}\right\} \tag{8}$$

The Bezout matrix $\hat{B}(f_1, f_2) = (\hat{b}_{ij})$ is defined by

$$\hat{b}_{ij} = |u_0 v_{i+j-1}| + |u_1 v_{i+j-2}| + \cdots + |u_k v_{i+j-k-1}|,$$

where $|u_r v_s| = u_s v_r - u_r v_s$, $k = \min(i-1, j-1)$ and $v_r = 0$

if $r > n$ [3]. It satisfies that

$$\frac{f_1(x)f_2(y) - f_1(y)f_2(x)}{x - y} = [1, x, x^2, \ldots, x^{m-1}]\hat{B}(f_1, f_2)[1, y, y^2, \ldots, y^{m-1}]^T.$$

Notice that the Bezout matrix $B(f_1, f_2)$ defined in Maple is as follows:

$$B(f_1, f_2) = -J\hat{B}(f_1, f_2)J, \tag{9}$$

where J is an anti-diagonal matrix with 1 as its nonzero entries.

Theorem 1. *[3] Given univariate polynomials $f_1, f_2 \in \mathbb{C}[x]$ with $\deg(f_1) = m$, $\deg(f_2) = n, m \geq n$, then we have*

$$\dim \text{NullSpace}(B(f_1, f_2)) = \deg(\gcd(f_1, f_2)).$$

Theorem 2. *[37] Given univariate polynomials $f_1(x), f_2(x)$ with $\deg(f_1) = m$, $\deg(f_2) = n, m \geq n$. Let $p(x) = \gcd(f_1, f_2)$ with $\deg(p) = r$, then we have*

1. *$\text{rank}(B(f_1, f_2)) = m - r$, and $\det(B(f_1, f_2)_k) \neq 0$ for $k \leq m - r$, where $B(f_1, f_2)_k$ is the $k \times k$ leading principal submatrix of $B(f_1, f_2)$, but $\det(B(f_1, f_2)_k) = 0$ for $k > m - r$.*

2. *Suppose $\mathbf{y} = [y_0, y_1, \ldots, y_{m-r-1}]^T$ satisfies $C\mathbf{y} = \mathbf{b}$, where $C = B(f_1, f_2)_{m-r}$ and $-\mathbf{b}$ is a vector formed from the first $m-r$ entries of the $m-r+1$-th column of $B(f_1, f_2)$. Let $\mathbf{u} = (u_i)_{i=0,\ldots,m-r} = (JB(f_1, 1))_{m-r+1} \cdot [y_0, \ldots, y_{m-r}]^T$, $y_{m-r} = 1$, then $f_1(x) = p(x)u(x)$, where $u(x) = \sum_{i=0}^{m-r} u_i x^i$.*

According to Theorem 2, we can estimate efficently the degree r of $\gcd(f_1, f_2)$ by checking whether the first $1 \times 1, 2 \times 2, 4 \times 4, \ldots, 2^{\lceil \log_2(m-r+1) \rceil} \times 2^{\lceil \log_2(m-r+1) \rceil}$ leading principal submatrices are singular. Suppose $r = \deg(\gcd(f_1, f_2))$, we compute the cofactor $u(x)$ by solving an $(m - r) \times (m - r)$ linear system and the GCD can be found using the approximate polynomial division based on FFT. It is clear that we only need to form the first $(m - r + 1) \times (m - r + 1)$ submatrix of $B(f_1, f_2)$. This will save a lot of time and space when $m - r \ll m$. For example, since the blurring function has very low degree, the GCD problem arising from reconstructing an image from distorted images will satisfy $m - r \ll m$.

We recall how to compute the unknown factor $p(x)$ from polynomials $f(x)$ and $u(x)$ by the DFT algorithm. The associated convolution form of $f(x) = p(x)u(x)$ is

$$\mathbf{f} = \mathbf{p} * \mathbf{u},$$

where $\mathbf{f}, \mathbf{p}, \mathbf{u} \in \mathbb{C}^m$ are coefficient vectors of polynomials f, p, u respectively and $m = \deg(f) + 1$. Evaluating of $f(x)$ and $u(x)$ at $x_k = e^{-\frac{2k\pi \mathrm{i}}{m}}, \quad 0 \leq k \leq m - 1$, we obtain the evaluation vector

$$[p(x_0), p(x_1) \ldots, p(x_{m-1})]^T,$$

where $p(x_k) = f(x_k)/u(x_k)$, for $0 \leq k \leq m - 1$. The coefficient vector $\mathbf{p}$ associated with $p(x)$ is obtained by applying inverse DFT to $[p(x_0), p(x_1) \ldots, p(x_{m-1})]^T$. The complexity of the polynomial division based on DFT for univariate polynomials is $O(m^2)$. The complexity of the fast algorithm using FFT for the univariate polynomial division is $O(m \log(m))$ [33, 34].

Now let us consider the general case, i.e., the condition $m - r \ll m$ does not hold. It is well known that the displacement rank of an $m \times m$ Bezout matrix is 2:

$$\nabla B = B - Z_1 B Z_0^T, \text{ and } \text{rank}(\nabla B) = 2, \tag{10}$$

where Z_a is the unit a-circulant matrix [34],

$$Z_a = \begin{bmatrix} 0 & & & a \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}.$$

Hence, it is also possible to apply fast algorithms to compute the GCD of univariate polynomials. The complexity of the fast algorithm given in [5] is $O(m^2)$. A more extensive description about fast algorithms for matrices with low displacement rank can be found in [21, 34].

Although we have a fast Bezout-type GCD algorithm, it has been pointed out in Section 1, the cost to generate the $m \times m$ matrix $B(f_1, f_2)$ is $O(m^2)$, which will be expensive if m is over hundreds. On the contrary, the Sylvester matrix $S(f_1, f_2)$ is:

$$\begin{bmatrix} u_m & u_{m-1} & \cdots & u_1 & u_0 & & & \\ & u_m & u_{m-1} & \cdots & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & u_m & u_{m-1} & \cdots & u_1 & u_0 \\ v_n & v_{n-1} & \cdots & v_1 & v_0 & & & \\ & v_n & v_{n-1} & & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & v_n & v_{n-1} & \cdots & v_1 & v_0 \end{bmatrix}.$$

It costs almost nothing to generate the Sylvester matrix. Moreover, we know that the Sylvester matrix S is a quasi-Toeplitz matrix, i.e., the matrix $S - Z_0 S Z_0^T$ has rank at most 2.

Fast GCD algorithms with complexity $O(m^2)$ have been given in [5, 29, 30, 41, 43]. Hence, when the condition $m - r \ll m$ does not hold, it would be more efficient to construct the Sylvester matrix of f_1 and f_2 and use the fast Sylvester-type GCD algorithms.

Algorithm Approximate Univariate Polynomial GCD

Input:
- $f_1(x), f_2(x) \in \mathbb{C}[x]$ with $m = \deg(f_1), n = \deg(f_2)$ and $m \geq n$.
- $\epsilon \in \mathbb{R}_{>0}$: the given tolerance.

Output:
- $p(x) \in \mathbb{C}[x]$: an approximate GCD of f_1 and f_2.
- $u(x), v(x) \in \mathbb{C}[x]$: approximate cofactors corresponding to f_1, f_2.

Case 1: $m - r \ll m$ holds.

UH1 Estimate the degree r of $\gcd(f_1, f_2)$ for the given tolerance ϵ, and form C and b as shown in Theorem 2.

UH2 Obtain $u(x)$ by solving $C\mathbf{y} = b$.

UH3 Compute $p(x)$ and $v(x)$ by applying the fast polynomial division based on FFT to $f_1(x)$ and $u(x)$; $f_2(x)$ and $p(x)$ respectively.

Case 2: $m - r \ll m$ does not hold.

UL1 Compute $u(x)$ via a fast GCD algorithm based on the displacement structure of the Sylvester matrix.

UL2 Compute $p(x)$ and $v(x)$ by applying the fast polynomial division based on FFT to $f_1(x)$ and $u(x)$; $f_2(x)$ and $p(x)$ respectively.

□

Theorem 3. *Given two univariate polynomials $f_1, f_2 \in \mathbb{C}[x]$ with $m = \max(\deg(f_1), \deg(f_2))$, $r = \deg(\gcd(f_1, f_2))$. In Case 1, if $m - r = O(m^{1/3})$, the total number of operations for computing the approximate GCD of univariate polynomials f_1 and f_2 based on the leading $(m - r + 1) \times (m - r + 1)$ principal submatrix of the Bezout matrix and FFT is $O(m \log(m))$. In Case 2, using the fast GCD algorithms based on the displacement structure of the Sylvester matrix, the total number of operations for computing the GCD is $O(m^2)$.*

Proof.

- Case 1. According to Theorem 2, we need to check whether the $k \times k$ leading principal submatrices are singular for $k \leq m - r + 1$. The number of operations involved in step UH1 is bounded by $O((m - r + 1)^3 \log(m - r + 1))$. It takes $O((m - r + 1)^3)$ operations to solve the linear system in step UH2. The fast polynomial divisions based on FFT in step UH3 cost $O(m \log(m))$. If $m - r = O(m^{1/3})$, then the operations in all three steps are bounded by $O(m \log(m))$.

- Case 2. The number of operations need in step UL1 is bounded by $O(m^2)$ since the Sylvester matrix has displacement rank 2.

□

3. AN APPROXIMATE BIVARIATE POLYNOMIAL GCD ALGORITHM

Suppose $f_1(x, y)$ and $f_2(x, y)$ are given in (4). We assume $\deg_x(f_1) = \deg_x(f_2) = m$ and $\deg_y(f_1) = \deg_y(f_2) = n$. In [35], they substituted

$$x = e^{-\frac{2k\pi \mathbf{i}}{m}},\ k = 0, 1, \ldots, m - 1$$

into f_1 and f_2. For each k, this results in two univariate polynomials $f_1(e^{-\frac{2k\pi \mathbf{i}}{m}}, y)$ and $f_2(e^{-\frac{2k\pi \mathbf{i}}{m}}, y)$, applying the approximate GCD algorithm to these univariate polynomials, they obtained scaled quantity $c_0(e^{-\frac{2k\pi \mathbf{i}}{m}})p(e^{-\frac{2k\pi \mathbf{i}}{m}}, y)$. They proceeded further by substituting

$$y = e^{-\frac{2l\pi \mathbf{i}}{n}},\ l = 0, 1, \ldots, n - 1$$

and formed a matrix of discrete Fourier transform elements,

$$A(k, l)a(k) = p(e^{-\frac{2k\pi \mathbf{i}}{m}}, e^{-\frac{2l\pi \mathbf{i}}{n}}), \tag{11}$$

where $A(k, l)$ is the evaluation of the GCD of $f_1(e^{-\frac{2k\pi \mathbf{i}}{m}}, y)$ and $f_2(e^{-\frac{2k\pi \mathbf{i}}{m}}, y)$. Carrying out similar operations by substituting $y = e^{-\frac{2l\pi \mathbf{i}}{n}}$ into f_1 and f_2, taking their univariate GCD and further substituting $x = e^{-\frac{2k\pi \mathbf{i}}{m}}$, they obtained another matrix

$$B(k, l)b(l) = p(e^{-\frac{2k\pi \mathbf{i}}{m}}, e^{-\frac{2l\pi \mathbf{i}}{n}}). \tag{12}$$

The vectors $a(k)$ and $b(l)$ are obtained by solving the least squares problem

$$A(k, l)a(k) - B(k, l)b(l) = 0, \tag{13}$$

and therefore, $p(x, y) = \gcd(f_1, f_2)$ can be computed by applying inverse DFT to

$$p(e^{-\frac{2k\pi \mathbf{i}}{m}}, e^{-\frac{2l\pi \mathbf{i}}{n}}) = \frac{1}{2}(A(k, l)a(k) + B(k, l)b(l)). \tag{14}$$

For images of size $n \times n$, their algorithm requires $O(n^4)$ operations.

For polynomials arising from blurred images, as we have mentioned in Section 1, the blurring function always has very low degree. Hence the degrees in variables x and y of the approximate GCD of $f_1(x, y)$ and $f_2(x, y)$ are almost as large as m and n. It would be much cheaper if we interpolate cofactor $u(x, y)$ or $v(x, y)$ by the above procedure, and then compute the GCD $p(x, y)$ by applying the fast polynomial division to $f_1(x, y)$ or $f_2(x, y)$. The division of two bivariate polynomials can also be done by the fast algorithm based on FFT. Hence, we can reduce the cost of the algorithm to $O(n^2 \log(n))$ for identifying both the true image and the blurring functions from blurred images of size $n \times n$ when the blurring functions have very low degree.

Algorithm Approximate Bivariate Polynomial GCD

Input: ▸ $f_1(x, y), f_2(x, y) \in \mathbb{C}[x, y]$ with $\deg_x(f_1) = \deg_x(f_2) = m$ and $\deg_y(f_1) = \deg_y(f_2) = n$.
▸ $\epsilon \in \mathbb{R}_{>0}$: the given tolerance.

Output: ▸ $p(x, y) \in \mathbb{C}[x, y]$: an approximate GCD of f_1 and f_2.
▸ $u(x, y), v(x, y) \in \mathbb{C}[x, y]$: approximate cofactors of f_1 and f_2 respectively.

I. Estimate $r = \deg_x(p)$ and $s = \deg_y(p)$ for the given tolerance ϵ.

II. **Case 1:** $m - r + n - s \ll m + n$.

BH1 Apply Algorithm Approximate Univariate Polynomial GCD (Case 1) to compute the evaluation matrix $\big[u(x_k, y_l)\big] \in \mathbb{C}^{(m-r+1)\times(n-s+1)}$, where

$$x_k = e^{-\frac{2k\pi \mathbf{i}}{m-r+1}}, \quad 0 \leq k \leq m - r,$$
$$y_l = e^{-\frac{2l\pi \mathbf{i}}{n-s+1}}, \quad 0 \leq l \leq n - s.$$

BH2 Apply inverse FFT to $\big[u(x_k, y_l)\big]$ to compute $u(x, y)$.

BH3 Compute $p(x, y)$ by applying the fast polynomial division to $f_1(x, y)$ and $u(x, y)$.

BH4 Compute $v(x, y)$ by applying the fast polynomial division to $f_2(x, y)$ and $p(x, y)$.

Case 2: $r + s \ll m + n$.

BL1 Apply Algorithm Approximate Univariate Polynomial GCD (Case 2) to compute the evaluation matrix $\big[p(x_k, y_l)\big] \in \mathbb{C}^{(r+1)\times(s+1)}$, where

$$x_k = e^{-\frac{2k\pi \mathbf{i}}{r+1}}, \quad 0 \leq k \leq r,$$
$$y_l = e^{-\frac{2l\pi \mathbf{i}}{s+1}}, \quad 0 \leq l \leq s.$$

BL2 Apply inverse FFT to the matrix $\big[p(x_k, y_l)\big]$ to get $p(x, y) = \gcd(f_1, f_2)$.

BL3 Compute $u(x,y), v(x,y)$ by applying the fast polynomial division to polynomials $f_1(x,y), p(x,y)$ and $f_2(x,y), p(x,y)$ respectively.

Theorem 4. *Given bivariate polynomials $f_1, f_2 \in \mathbb{C}[x,y]$, we assume that $n = \deg_x(f_1) = \deg_x(f_2) = \deg_y(f_1) = \deg_y(f_2)$, $r = \deg_x(\gcd(f_1,f_2)) = \deg_y(\gcd(f_1,f_2))$. The total number of operations for computing the approximate GCD of polynomials f_1 and f_2 is bounded by $O(n^2\log(n))$ when $n-r = O(n^{1/2})$ or $r = O(\log(n))$ which corresponding to Case 1 and Case 2 respectively.*

Proof.

- According to Theorem 3, the number of operations for estimating r and s in step I is bounded by $O(n^2)$.
- Case 1. It takes $O(n^2\log(n))$ operations to get the evaluations of polynomials f_1 and f_2 at the DFT points. We need $O((n-r)^4)$ operations to get the evaluation matrix $[u(x_k,y_l)]$ by step UH2 of Algorithm Approximate Univariate Polynomial GCD (Case 1). If $n-r = O(n^{1/2})$, then the number of operations involved in step BH1 is bounded by $O(n^2\log(n)+n^2)$. Moreover, if we apply the fast polynomial division based on FFT in steps BH2, BH3 and BH4, the total number of operations involved in these three steps is bounded by $O(n^2\log(n))$.
- Case 2. We compute the evaluation matrix $[p(x_k,y_l)]$ via the fast GCD algorithm based on the displacement structure of the Sylvester matrix. Hence, the complexity of BL1 is bounded by $O(rn^2+n^2\log(n))$. If $r = O(\log(n))$, then the complexity of BL1 is bounded by $O(n^2\log(n))$. The total number of operations of steps BL2 and BL3 is bounded by $O(n^2\log(n))$ if we apply the fast polynomial division based on FFT.

$\square$

4. EXPERIMENTS

The following examples come from literatures on image deconvolution. We show that our new algorithm implemented in Maple can reconstruct true images from blurred images successfully in few seconds. The algorithm in [35] has also been implemented in Maple. All experiments are run on an Inter(R) Core(TM) 2 Quad Cpu at 2.40 GHz for Digits=14 in Maple 13 under Windows.

Example 1. *(Reconstructing Image From Three Distorted Images)*
In Figure 1, Figure 1.(a) is an image of size 250 × 250 scanned from [7]. Figure 1.(b) contains three distorted images built by convolving Figure 1.(a) with three 7 × 7 coprime distortion filters. Since we can not add white noise in Maple directly, for simplicity, we add random noise with SNR $= 52dB$.
Figure 1.(c) is the image reconstructed by running Algorithm Approximate Bivariate Polynomial GCD (Case 1, $7+7 \ll 250+250$) in about 1.10 seconds; whereas the time obtained by running the algorithm [35] in Maple is 46.90 seconds.

Example 2. *(Reconstructing RGB Image From Two Distorted Images)*

Size	Three Grey		Two RGB	
	Time(s) (Bezout)	*Time(s) (Liang)*	*Time(s) (Bezout)*	*Time(s) (Liang)*
256 × 256	1.23	49.34	1.66	152.40
512 × 512	5.72	317.15	7.66	982.96
1024 × 1024	28.63	2995.16	45.42	7914.44

Table 1: Algorithm performance on benchmark

In Figure 2, Figure 2.(a) is an image of size 128 × 170 scanned from [35]. Figure 2.(b) and Figure 2.(c) are two distorted images built by convolving Figure 2.(a) with two 7 × 7 co-prime distortion filters and with the additive noise SNR $= 52dB$.
Figure 2.(d) is the image reconstructed successfully by running Algorithm Approximate Bivariate Polynomial GCD (Case 1, $7+7 \ll 128+170$) in about 0.73 seconds; whereas the time obtained by running the algorithm in [35] is 47.26 seconds.

In Table 1, we show the performance of our algorithm for recovering large images obtained by convolving three original images downloaded from http://sipi.usc.edu/database/ with distortion filters of sizes 7×7,13×13,19×19 respectively. The additive noises are the same: SNR $= 63dB$. Here *Size* denote the size of the original images; *Time(Bezout)* and *Time(Liang)* contain the timings to solve blind image deconvolution problem by running our new algorithm based on Bezout-type GCD algorithm and using the algorithm in [35], respectively.

Remark 1. In [35], they also considered the case where there was only one available grey image. They assumed the blurring function is one dimension. By z-transforms, the polynomial form of (1) can be written as :

$$f(x,y) = p(x,y)u(y) + n(x,y), \tag{15}$$

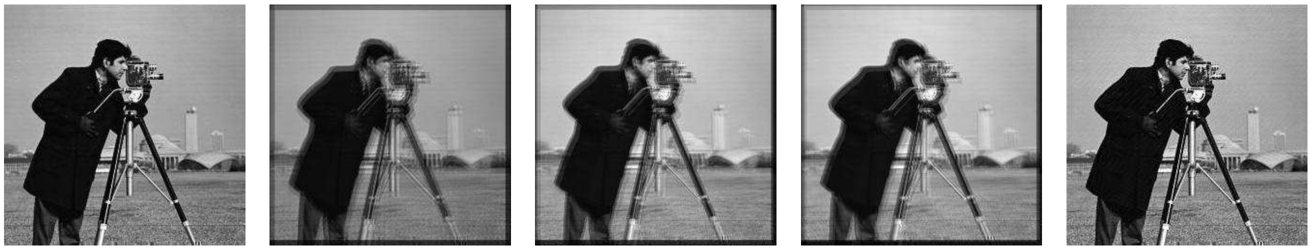
where f, p, u, n are z-transforms of F, P, U, N respectively. Suppose $f(x,y)$ can be represented as follows:

$$f(x,y) = \sum_{i=0}^{m} a_i(y)x^i,$$

where $m = \deg_x(f)$. Assuming $p(x,y)$ is primitive with respect to x, then $u(y)$ is the common divisor of $a_0(y), \ldots, a_m(y)$. The GCD polynomial $p(x,y)$ can be obtained by performing the polynomial division of $f(x,y)$ and $u(y)$. In the approximate case, we need to compute approximate univariate GCDs of several polynomials (more than 2), which can be solved by the following two options:

1. Choose random values $s_0, s_1, \ldots s_m, t_0, t_1, \ldots t_m$ and get two polynomials
$$f_1 = \sum_{i=0}^{m} s_i a_i(y), \quad \text{and} \quad f_2 = \sum_{i=0}^{m} t_i a_i(y).$$
Compute the approximate GCD $u'(y)$ of f_1 and f_2, with high probability, we have $u'(y) = \gcd(a_0, \ldots, a_m)$.
2. Construct the generalized Bezout matrix $B(a_0, \ldots, a_m)$ or the generalized Sylvester matrix $S(a_0, \ldots, a_m)$ to compute the approximate GCD.

$\square$



(a) An original Image (b) Three Blurred Images (c) Reconstructed Image

Figure 1: Blind deblurring from three distorted and noisy (SNR $= 52dB$) images

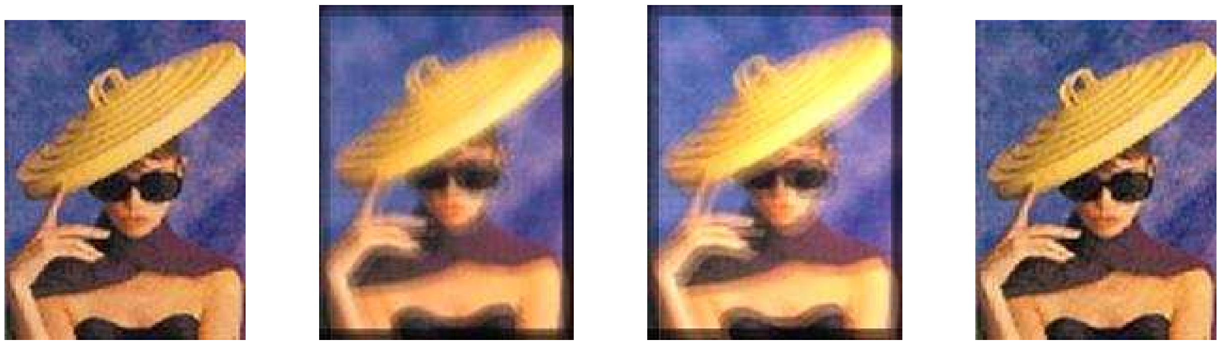


Fig (a) An original Image Fig (b) Blurred Image 1 Fig (c) Blurred Image 2 Fig (d) Reconstructed Image

Figure 2: Blind deblurring from two distorted and noisy (SNR $= 52dB$) RGB images

Example 3. *(One Blurred Grey Image)*
In Figure 3, Figure 3.(a) is an image of size 337×77 scanned from [35]. Figure 3.(b) is an image of size 312×77 reconstructed in about 3.17 seconds by using the Sylvester-type GCD algorithm for univariate polynomials and option 1. Notice that for this example, we need to generate a 336×336 Bezout matrix. It takes already more than 5 seconds in Maple to generate the matrix. Whereas the time obtained by running the algorithm in [35] is 3.46 seconds.

Example 4. (*Reconstructing RGB Image from One Blurred Image*)
Figure 4.(a) is an image of size 128×170 which is scanned from [35]. The distorted image of Figure 4.(b) is built by convolving Figure 4.(a) with a 7×7 distortion filter and with the additive noise SNR $= 53dB$. Figure 4.(c) is the image reconstructed by running Algorithm Approximate Bivariate Polynomial GCD (Case 2, $7 + 7 \ll 128 + 170$) in about 5.85 seconds if we are using the Sylvester-type GCD algorithm for univariate polynomials. Whereas the time obtained by running the algorithm in [35] is 118.47 seconds. $\square$

5. CONCLUSION

In this paper, we present a specialized algorithm for computing approximate GCDs of univariate or bivariate polynomials arising from blind images deconvolution problem. To recover images from the blurred ones of size $n \times n$, we are able to reduce the complexity to $O(n^2 \log(n))$ in the case of blurring functions of very low degree.

We have implemented both Bezout-type and Sylvester-type univariate GCD algorithms, the fast polynomial division and interpolation based on FFT in Maple. We show that our algorithm is efficient and quite robust. When the additive noise satisfies SNR $\geq 50dB$, i.e., the relative errors of polynomials corresponding to the distorted images are within 10^{-4}, our new algorithm can recover successfully true images from blurred and distorted images. However, we also notice that when the additive noise is smaller than $50dB$, it is still hard to use our algorithm to recover original images from distorted ones. Moreover, although our new algorithm is much faster than the one in [23] for computing GCDs of bivariate polynomials, unfortunately, the backward error becomes much bigger. We may have to sacrifice the efficiency to have a more robust GCD algorithm. Some new techniques in computing GCDs [5, 23, 25, 30, 39] of polynomials and multivariate polynomial interpolation [18, 22, 24] may be used to improve the efficiency and stability of our algorithm.

Acknowledgments

We thank John May for valuable suggestions that improve our Maple code significantly. The authors also wish to thank the reviewers for their helpful comments.

6. REFERENCES

[1] Barnett, S. Greatest common divisor of two polynomials. *Linear Algebra Appl. 3* (1970), 7–9.

[2] Barnett, S. Greatest common divisor of several polynomials. *Proc. Camb. Phil. Soc 70* (1971), 263–268.

[3] Barnett, S. A note on the Bezoutian matrix. *SIAM J.APPL.MATH. 22*, 1 (January 1972), 84–86.

[4] Beckermann, B., and Labahn, G. When are two polynomials relatively prime? *J. Symbolic Comput. 26* (1998), 677–689.

[5] Bini, D. A., and Boito, P. Structured matrix-based methods for polynomial ϵ-gcd: analysis and comparisons. In Brown [6].

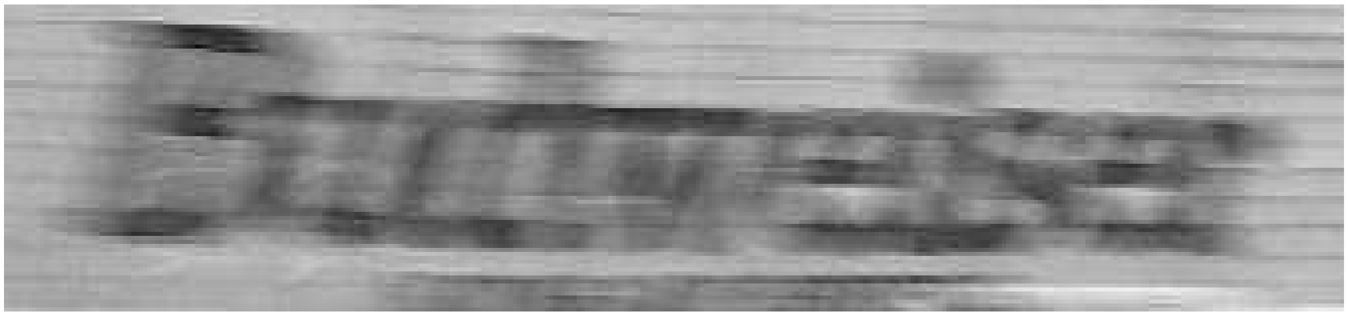


Figure (a): Blurred Image

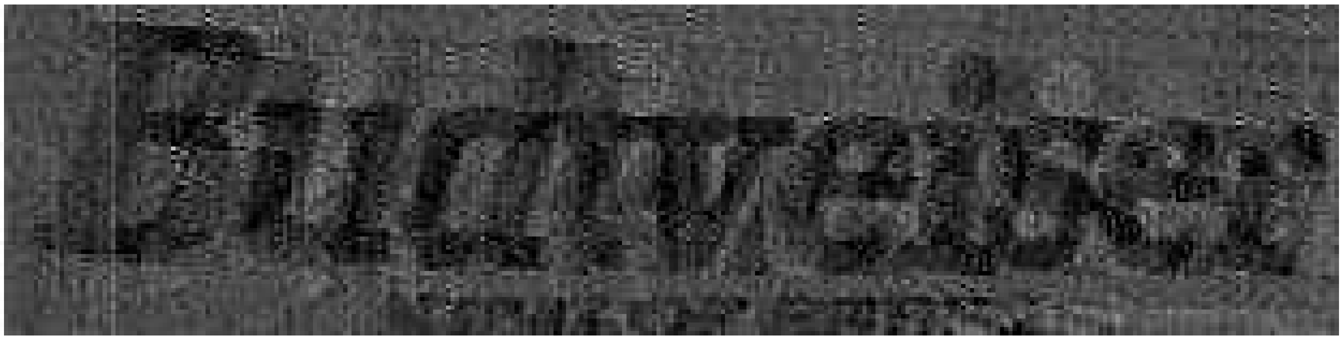
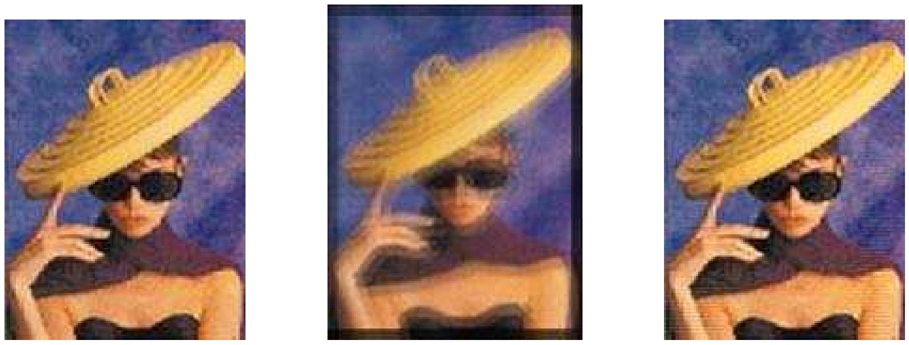


Figure (b): Reconstructed Image

Figure 3: Blind deblurring of linear motion blurred image



(a) An original Image (b) Three Blurred Images (c) Reconstructed Image

Figure 4: Blind deblurring from one distorted and noisy (SNR $= 53dB$) RGB images

[6] Brown, C. W., Ed. *ISSAC 2007 Proc. 2007 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2007), ACM Press.

[7] Campisi, P., and Egiazarian, K. *Blind image deconvolution: theory and applications.* CRC Press, 2007.

[8] Corless, R., Gianni, P., Trager, B., and Watt, S. The singular value decomposition for polynomial systems. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95* (New York, 1995), A. Levelt, Ed., ACM Press, pp. 96–103.

[9] Corless, R., Watt, S., and Zhi, L. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Transactions on Signal Processing 52* (Dec 2004), 3394–3402.

[10] Diaz-Toca, G., and Gonzalez-Vega, L. Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases. *Linear Algebra Appl. 412* (2006), 222–246.

[11] Diaz-Toca, G. M., and Gonzalez-Vega, L. Barnett's theorems about the greatest common divisor of several univariate polynomials through Bezout-like matrices. *Journal of Symbolic Computation 34*, 1 (2002), 59 – 81.

[12] Dumas, J.-G., Ed. *ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2006), ACM Press.

[13] Emiris, I., Galligo, A., and Lombardi, H. Certified approximate univariate GCDs. *J. Pure Applied Algebra. 117 & 118* (May 1996), 229–251. Special Issue on Algorithms for Algebra.

[14] Gao, S., Kaltofen, E., May, J., Yang, Z., and Zhi, L. Approximate factorization of multivariate polynomials via differential equations. In Gutierrez [19], pp. 167–174.

[15] Gemignani, L. Gcd of polynomials and Bezout matrices. In *Proc. 1997 Internat. Symp. Symbolic*

Algebraic Comput. ISSAC'97 (New York, 1997), Küchlin, Ed., ACM Press, pp. 271–277.

[16] Ghiglia, D. C., Romero, L. A., and Mastin, G. A. Systematic approach to two-dimensional blind deconvolution by zero-sheet separation. *J. Opt. Soc. Am. A 10* (1993), 1024–1036.

[17] Giannakis, G., and Heath, R. Blind identification of multichannel fir blurs and perfect image restoration. *IEEE Trans. Image Processing 9* (2000), 1877–1896.

[18] Giesbrecht, M., Labahn, G., and Lee, W. Symbolic-numeric sparse interpolation of multivariate polynomials. In Dumas [12], pp. 116–123.

[19] Gutierrez, J., Ed. *ISSAC 2004 Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2004), ACM Press.

[20] Hribernig, V., and Stetter, H. Detection and validation of clusters of polynomials zeros. *J. Symbolic Comput. 24* (1997), 667–681.

[21] Kailath, T., and Sayed, A. H. Displacement structure: theory and applications. *SIAM Review 37*, 3 (1995), 297–386.

[22] Kaltofen, E., and Yang, Z. On exact and approximate interpolation of sparse rational functions. In Brown [6], pp. 203–210.

[23] Kaltofen, E., Yang, Z., and Zhi, L. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In Dumas [12], pp. 169–176. Full version, 21 pages. Submitted, December 2006.

[24] Kaltofen, E., Yang, Z., and Zhi, L. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In *SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput.* (New York, N. Y., 2007), J. Verschelde and S. M. Watt, Eds., ACM Press, pp. 11–17.

[25] Kaltofen, E., Yang, Z., and Zhi, L. Structured low rank approximation of a Sylvester matrix. In *Symbolic-Numeric Computation*, D. Wang and L. Zhi, Eds., Trends in Mathematics. Birkhäuser Verlag, Basel, Switzerland, 2007, pp. 69–83. Preliminary version in [40], pp. 188–201.

[26] Karmarkar, N., and Lakshman Y. N. Approximate polynomial greatest common divisors and nearest singular polynomials. In *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 1996), Lakshman Y. N., Ed., ACM Press, pp. 35–42.

[27] Karmarkar, N. K., and Lakshman Y. N. On approximate GCDs of univariate polynomials. *J. Symbolic Comput. 26*, 6 (1998), 653–666. Special issue on Symbolic Numeric Algebra for Polynomials S. M. Watt and H. J. Stetter, editors.

[28] Lane, R. G., and Bates, R. H. T. Automatic multidimensional deconvolution. *J. Opt. Soc. Am. A 4* (1987), 180–188.

[29] Li, B., Liu, Z., and Zhi, L. A structured rank-revealing method for sylvester matrix. *J. Comput. Appl. Math. 213*, 1 (2008), 212–223.

[30] Li, B., Yang, Z., and Zhi, L. Fast low rank approximation of a Sylvester matrix by structured total least norm. *J. JSSAC (Japan Society for Symbolic and Algebraic Computation) 11*, 3,4 (2005), 165–174.

[31] Markovsky, I., and Huffel, S. V. An algorithm for approximate common divisor computation. Internal Report 05-248, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2005.

[32] Noda, M., and Sasaki, T. Approximate GCD and its application to ill-conditioned algebraic equations. *J.Comput. Appl.Math. 38* (1991), 335–351.

[33] P. Duhamel, M. V. Fast Fourier Transforms: A tutorial review and a state of the art. *Signal Processing 19* (April 1990), 259–299.

[34] Pan, V. Numerical computation of a polynomial GCD and extensions. *Information and computation 167* (2001), 71–85.

[35] Pillai, S. U., and Liang, B. Blind image deconvolution using a robust GCD approach. *IEEE Transactions on Image Processing 8*, 2 (1999), 295–301.

[36] Schönhage, A. Quasi-gcd computations. *Journal of Complexity 1* (1985), 118–137.

[37] Sun, D., and Zhi, L. Structured low rank approximation of a Bezout matrix. *MM Research Preprints 25* (December 2006), 207–218.

[38] Sun, D., and Zhi, L. Structured low rank approximation of a Bezout matrix. *Mathematics in Computer Science 1*, 2 (2007), 427–437.

[39] Terui, A. An iterative method for calculating approximate GCD of univariate polynomials. In *ISSAC 2009 Proc. 2009 Internat. Symp. Symbolic Algebraic Comput.* (New York, NY, USA, 2009), J. May, Ed., ACM, pp. 351–358.

[40] Wang, D., and Zhi, L., Eds. *Internat. Workshop on Symbolic-Numeric Comput. SNC 2005 Proc.* (2005). Distributed at the Workshop in Xi'an, China, July 19–21.

[41] Zarowski, C. J., Ma, X., and Fairman, F. W. A QR-factorization method for computing the greatest common divisor of polynomials with real-valued coefficients. *IEEE Trans. Signal Processing 48* (2000), 3042–3051.

[42] Zeng, Z., and Dayton, B. The approximate GCD of inexact polynomials part II: a multivariate algorithm. In Gutierrez [19], pp. 320–327.

[43] Zhi, L. Displacement structure in computing approximate GCD of univariate polynomials. In *Proc. Sixth Asian Symposium on Computer Mathematics (ASCM 2003)* (Singapore, 2003), Z. Li and W. Sit, Eds., vol. 10 of *Lecture Notes Series on Computing*, World Scientific, pp. 288–298.