FLAT-FIELDING OF SOLAR Hα OBSERVATIONS BASED ON THE MAXIMUM CORRENTROPY CRITERION

GAO-GUI XU1, SHENG ZHENG1,2, GANG-HUA LIN2, and XIAO-FAN WANG2

1 College of Science, China Three Gorges University, Hubei, Yichang 443002, China
2 Key Laboratory of Solar Activity, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China

Received 2015 December 9; revised 2016 May 20; accepted 2016 May 24; published 2016 August 18

ABSTRACT

The flat-field CCD calibration method of Kuhn et al. (KLL) is an efficient method for flat-fielding. However, since it depends on the minimum of the sum of squares error (SSE), its solution is sensitive to noise, especially non-Gaussian noise. In this paper, a new algorithm is proposed to determine the flat field. The idea is to change the criterion of gain estimate from SSE to the maximum correntropy. The result of a test on simulated data demonstrates that our method has a higher accuracy and a faster convergence than KLL’s and Chae’s. It has been found that the method effectively suppresses noise, especially in the case of typical non-Gaussian noise. And the computing time of our algorithm is the shortest.

Key words: astroparticle physics – methods: observational – Sun: photosphere – techniques: image processing

1. INTRODUCTION

An ideal imaging system should not contain any unwanted signals that might arise from various sources, including electrical, mechanical, and optical. Fortunately, most effects can be accurately corrected, such as subtracted dark frame and bias. Thus, the real imaging system can be expressed as

\[ a = of + n, \]

where \( o \) is the object image, \( a \) is the observed image, \( f \) represents the non-uniform response (called “gain function”), and \( n \) the noise.

Flat-fielding is a process of obtaining the object image \( o \) from the observed image \( a \). In this process it is most important to determine the gain function \( f \). The easiest way of finding the gain function is to observe a very uniform target. Then the observed image reflects the gain function (Stetson & Harris 1988). It is very difficult to provide an absolutely uniform object image. Another approach, which does not require an absolutely uniform object, is to average multiple images taken while moving the CCD around an area near the center (Yang et al. 2003). Although this method is very simple and easy to implement, it is not carried out in real time. That means the gain function when observing the object is not the same as the calculated gain function when observing the uniform illumination, since the telescope will move after observing the uniform illumination.

Kuhn et al. (1991: KLL) proposed a method for computing a gain function from a set of relatively shifted images. Its solution is a Jacobi relaxation solution to Poisson’s equation. This method is simple and has been applied to solar observations. It is a convenient way of obtaining the gain function without uniform illumination. However, this method is sensitive to noise and its ability to suppress noise is poor. Moreover, It has slow convergence for large images (Denker et al. 1999).

Since shrinking the image is just a technique to improve the convergence, Toussaint et al. (2003) applied the standard technique of simultaneous overrelaxation to KLL’s method. The result shows that it improved the convergence of CCD gain calibration.

Chae (2004) proposed a more practical method of computing a gain function by using relatively shifted images. This assumes that the object images are variable during the observation. It can simultaneously determine the gain function, the object image, and the change in light level. But the accuracy may be lower than in KLL’s method.

Although there are many methods for improving the convergence or the accuracy of KLL’s method, they are based on the minimum of the sum of squares error (SSE). Recently, a new concept called the maximum correntropy criterion (MCC: Liu et al. 2006, 2007), which is derived from information theoretic learning (ITL: Principe 2010), has received a lot of attention. Unlike SSE, MCC is very suitable for non-Gaussian noise and impulsive noise. We therefore propose a new method. Like KLL’s method, our method uses a number of relatively shifted images of the same object. But the criterion for the best optimization function is different. We change the criterion of the gain estimate from SSE to MCC. Our method has the highest accuracy and the fastest convergence. Moreover, it can effectively suppress noise, especially non-Gaussian noise.

The paper is organized as follows: Section 2 introduces the formalism of KLL’s method. Section 3 describes the method of flat-fielding based on MCC and a flowchart of the algorithm. Section 4 describes the data used, shows the performance of our algorithm from several aspects, and compares with other flat-fielding methods. Section 5 presents the results and the analysis of applying our algorithm to real Hα observations. We conclude this study in Section 6.

2. KLL FORMALISM

The basic steps of KLL’s method are as follows. Consider an image that is shifted by a vector \( a_i \) from an invariant source \( s \),

\[ s_i(x) = s(x - a_i), \]

where \( x \) represents a pixel in the CCD frame and \( i \) represents the \( i \)th frame in a series \( i = 1, \ldots, N \). After subtracting the dark current and amplifier bias at each pixel, we have

\[ d_i(x) = g(x)s_i(x), \]

where \( d_i \) and \( g \) are the observed image and pixel gain at \( x \) on the CCD. Thus, we assume that the source image does not

3 Corresponding author.
change between observations. So

$$
\frac{d_i(x + a_i)}{d_j(x + a_j)} = \frac{g(x + a_i)}{g(x + a_j)},
$$

(4)

The logarithmic form of the above equation is

$$
D_i(x + a_i) - D_j(x + a_j) - G(x + a_i) + G(x + a_j) = 0,
$$

(5)

where \( \log d_i \equiv D_i \) and \( \log g \equiv G \). Then a solution for \( G(x) \) is to minimize

$$
\sum_{i<j,x} [D_i(x + a_i) - D_j(x + a_j) - G(x + a_i) + G(x + a_j)]^2.
$$

(6)

Since the function \( G \) is independent of the observation, the problem is that there is no exact solution. But we can seek a least-squares solution for the gain function \( G \), that is

$$
\sum_{i<j} [G(x) - G(x - \delta_{ij}) - D_i(x) + D_j(x - \delta_{ij})]
+ \sum_{i<j} [G(x) - G(x + \delta_{ij}) - D_i(x) + D_j(x + \delta_{ij})] = 0,
$$

(7)

where \( \delta_{ij} = a_i - a_j \) and the first sum on the left-hand side of the above equation is over all distinct \( i \) and \( j \). Since finding an exact solution to Equation (7) is difficult, we construct solutions by iterating from an initial guess \( G^0(x) = 0 \). The solution at iteration \( r + 1 \) is given by

$$
G^{r+1}(x) = K(x) + \frac{1}{n(x)} \sum_{i<j} [G^r(x - \delta_{ij}) + G^r(x + \delta_{ij})],
$$

(8)

where

$$
K(x) = \frac{1}{n(x)} \left\{ \sum_{i<j} [D_i(x) - D_j(x - \delta_{ij})] + \sum_{i<j} [D_i(x) - D_i(x + \delta_{ij})] \right\}
$$

(9)

and \( n(x) \) is the total number of terms. This has the form of a relaxation solution to Poisson’s equation. The average of surrounding pixels from the previous iteration determined the iterated solution at a given pixel. This is the basis of KLL’s method (Kuhn et al. 1991).

3. METHOD BASED ON MCC

3.1. Maximum Correntropy Criterion

Entropy is a way of quantifying information by using mathematical methods. The concept of correntropy was proposed for ITL (Liu et al. 2007). It dealt with non-Gaussian noise and impulsive noise. The correntropy is a generalized similarity measure between two arbitrary random variables \( A \) and \( B \), defined as

$$
V_c(A, B) = E[k_c(A - B)],
$$

(10)

where \( k_c(.) \) is the kernel function and it satisfies Mercer’s theory (Vapnik 1995), and \( E[. \] denotes the mathematical expectation. It maps the input space to a higher-dimensional space by using the kernel technique.

In practice, the joint probability density function (PDF) of variables \( A \) and \( B \) is unknown, but there are a finite number of samples \( \{(a_i, b_i)\}_{i=1}^{n} \). We can get the sample estimator of correntropy as follows

$$
\hat{V}_c(A, B) = \frac{1}{n} \sum_{i=1}^{n} g(a_i - b_i, \sigma),
$$

(11)

where \( g(x) \) is the Gaussian kernel, \( g(a - b, \sigma) = \exp \left( -\frac{||a - b||^2}{\sigma^2} \right) \).

Intuitively, correntropy reflects the degree of similarity between \( A \) and \( B \); if \( A \) and \( B \) have a large correntropy then they are similar. Thus, the maximum of the correntropy of error in Equation (11) is called the MCC (Liu et al. 2007).

However, it is difficult to solve Equation (11) because of its form. But there is a proposition for converting it to a more straightforward form to simplify this problem.

Proposition 1. There exists a convex function \( \varphi: R \to R \), such that

$$
g(x, \sigma) = \max_{p < 0} \left( p \frac{||x||^2}{\sigma^2} - \varphi(p) \right),
$$

and for a fixed \( x \), the maximum is reached at \( p = -g(x, \sigma) \).

The proof of Proposition 1 is based on convex conjugated function theory (Rockafellar 1997) and can be found in Yan et al. (2011).

3.2. Flat-fielding Based on MCC

The central idea of the KLL method is to find a least-squares solution, which means that two variables are almost the same at each pixel when the least-squares solution is found. In that situation, they have a large correntropy, so we can change the least-squares criterion to MCC. Rewrite Equation (5) as

$$
D_i(x + a_i) - G(x + a_i) - D_j(x + a_j)
+ G(x + a_j) = \varepsilon_{ij}(x).
$$

(13)

Writing \( \delta_{ij} = a_i - a_j \), we have

$$
\varepsilon_{ij}(x - \delta_{ij}) = D_i(x) - G(x) - D_j(x - \delta_{ij}) + G(x - \delta_{ij}),
$$

(14)

$$
\varepsilon_{ij}(x + \delta_{ij}) = D_i(x + \delta_{ij}) - G(x + \delta_{ij}) - D_j(x) + G(x),
$$

(15)

Integrating MCC, we have

$$
\max_{G(x)} \frac{1}{n} \sum_{i<j} g([D_i - G(x)] - [D_j(x - \delta_{ij}) - G(x - \delta_{ij})], \sigma),
$$

(16)

$$
\max_{G(x)} \frac{1}{n} \sum_{i<j} g([D_i(x + \delta_{ij}) - G(x + \delta_{ij})] - [D_j(x) - G(x)], \sigma).
$$

(17)

Since the above optimization function is difficult to solve, we construct the following optimization problem of flat-fielding by
applying Proposition 1 (see Section 3.1):

\[
\max_{G(x)} \frac{C}{2} \sum_{i<j} p_{ij}(x - \delta_{ij}) \frac{\varepsilon_{ij}^2(x - \delta_{ij})}{\sigma^2},
\]

(18)

\[
\max_{G(x)} \frac{C}{2} \sum_{i<j} p_{ij}(x + \delta_{ij}) \frac{\varepsilon_{ij}^2(x + \delta_{ij})}{\sigma^2},
\]

(19)

where \( p_{ij} = -g(\varepsilon_{ij}, \sigma) = -\exp\left(-\frac{\varepsilon_{ij}^2}{\sigma^2}\right) \).

Concretely, Equation (18) is the best optimization function of Equations (14) and (19) is the best optimization function of Equation (15). Then, we have two optimization functions and their form is similar. For the first optimization function, (18), we obtain the following equivalent problem by introducing the variable \( q_{ij} = -p_{ij} \):

\[
\begin{align*}
\min_{G(x)} & \frac{C}{2} \sum_{i<j} \left( q_{ij}(x - \delta_{ij}) \frac{\varepsilon_{ij}^2(x - \delta_{ij})}{\sigma^2} \right) \\
\text{s.t.} & \\
& D_i(x) - G(x) - D_j(x - \delta_{ij}) + G(x - \delta_{ij}) = \varepsilon_{ij}(x - \delta_{ij}).
\end{align*}
\]

(20)

In order to solve the above problem, we introduce the Lagrangian function as follows:

\[
L = \frac{C}{2} \sum_{i<j} q_{ij}(x - \delta_{ij}) \frac{\varepsilon_{ij}^2(x - \delta_{ij})}{\sigma^2}
- \sum_{i<j} \alpha_{ij} \left[ D_i(x) - G(x) - D_j(x - \delta_{ij}) + G(x - \delta_{ij}) - \varepsilon_{ij}(x - \delta_{ij}) \right].
\]

(21)
where $\alpha_{ij}$ is a Lagrangian multiplier. Using the Karush–Kuhn–Tucker (KKT) condition (Boyd & Vandenberghe 2004), we have

$$\frac{\partial L}{\partial \varepsilon_{ij}(x - \delta_{ij})} = C_{ij}(x - \delta_{ij})\frac{\varepsilon_{ij}(x - \delta_{ij})}{\sigma^2} + \alpha_{ij} = 0, \quad (22)$$

$$\frac{\partial L}{\partial \alpha_{ij}} = D_i(x) - G(x) - D_j(x - \delta_{ij}) + G(x - \delta_{ij}) = \varepsilon_{ij}(x - \delta_{ij}) = 0, \quad (23)$$

$$\frac{\partial L}{\partial G(x)} = \sum_{i<j} \alpha_{ij} = 0, \quad (24)$$

By introducing the variable $y_{ij} = D_j(x - \delta_{ij}) - D_i(x) - G(x - \delta_{ij})$ and combining Equations (22)–(24), we obtain

$$\begin{bmatrix} Q & e^T \\ e & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \sigma^2 \end{bmatrix} - \begin{bmatrix} 0 \\ G(x) \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix}$$

Equation (25) is

$$\begin{cases} G(x) = -\frac{e^T Q^{-1} e}{\sigma^2} = -\frac{\sum_{i<j} C_{ij}(x - \delta_{ij})y_{ij}}{\sigma^2} \\ \alpha_{ij} = \frac{C_{ij}(x - \delta_{ij})}{\sigma^2} (y_{ij} + G(x)) \end{cases} \quad (26)$$

Up to now, we have solved the optimization function Equation (18). Since it is difficult to find the exact solution, we construct solutions by iterating from an initial guess $G^0 = 0$. The solution at iteration $r + 1$ is given by

$$G^{r+1}(x) = \frac{\sum q_{ij}(x - \delta_{ij})[D_i(x) + G^r(x - \delta_{ij}) - D_j(x - \delta_{ij})]}{\sum q_{ij}(x - \delta_{ij})} \quad (27)$$
where

\[ \begin{align*}
\varepsilon_i (x - \delta_i) &= D_i (x) - D_j (x - \delta_j) - G' (x) + G' (x - \delta_j) \\
q_j (x - \delta_j) &= \exp \left( -\frac{\varepsilon_i (x - \delta_i)}{\sigma^2} \right). \\
\end{align*} \]

(28)

Since the optimization function Equation (19) has a similar form to the optimization function Equation (18), an iterative solution of the former can be given as

\[
G^{r+1} (x) = \frac{\sum q_i (x + \delta_i) [D_i (x + \delta_i) - D_j (x) - G' (x + \delta_i)]}{\sum q_i (x + \delta_i)}. 
\]

(29)

Combining Equations (27) and (29), we construct solutions of flat-fielding based on MCC by iterating from the initial guess \( G^0 = 0 \). The solution at iteration \( r + 1 \) is given by

\[
G^{r+1} (x) = \frac{1}{2} \left\{ \frac{\sum q_i (x + \delta_i) [D_i (x + \delta_i) - D_j (x) - G' (x + \delta_i)]}{\sum q_i (x + \delta_i)} + \sum q_i (x - \delta_i) [D_i (x) + G' (x - \delta_i) - D_j (x - \delta_i)] \right\}.
\]

(30)

Considering the effects of the boundary, Equation (30) can be written as

\[
G^{r+1} (x) = \frac{1}{2} \left\{ \frac{\sum q_i (x + \delta_i) [D_i (x + \delta_i) - D_j (x) - G' (x + \delta_i)] w_j (x + \delta_i)}{\sum q_i (x + \delta_i) w_j (x + \delta_i)} + \frac{\sum q_i (x - \delta_i) [D_i (x) + G' (x - \delta_i) - D_j (x - \delta_i)] w_j (x - \delta_i)}{\sum q_i (x - \delta_i) w_j (x - \delta_i)} \right\}.
\]

(31)
where the boundary \( w \) is defined as

\[
\begin{cases}
1, & 0 \leq x \leq \text{rows} - 1 \\
0, & \text{otherwise}
\end{cases}
\]

then the solution of flat-fielding based on MCC is given after many iterations.

### 3.3 Kernel Size \( \sigma \)

As in any kernel method, the selection of kernel size will affect the performance of the proposed technique, and it controls all robust properties of correntropy (Liu et al. 2007). The kernel size is the window within which the similarity of the two random variables is computed. A large kernel size will therefore yield a similarity measure close to the SSE value. A correctly tuned value of kernel size can eliminate the effects of outliers and noise. Adaptation of the kernel size is proposed in Principe (2010), where the criterion of adapting the kernel size parameter online is to use the Kullback–Leibler divergence between the true error PDF and the estimated PDF.

Let \( \hat{p}_s(e) \) be the estimated density from a window of \( N \) samples of the error, and it is evaluated by using the Gaussian kernel \( G_\sigma \) with width \( \sigma \):

\[
\hat{p}_s(e) = \frac{1}{N} \sum_{i=1}^{N} G_\sigma(e - e_i).
\]

When the discriminant information between the estimated density and the true density \( p_s(e) \) of the errors is minimum, the value of \( \sigma \) is optimal. Thus, the cost function for optimizing the kernel size is

\[
D_{KL}(p||\hat{p}) = \int p(e) \log \left( \frac{p(e)}{\hat{p}_s(e)} \right) de
\]

\[
= \int p(e) \log p(e) de - \int p(e) \log \hat{p}_s(e) de.
\]

(33)

Since the first term in Equation (33) is independent of \( \sigma \), an optimal value of \( \sigma \) is therefore one that maximizes the second term. Then the cost function can be written as

\[
J_{KL}(\sigma) = E[\log \hat{p}_s(e)].
\]

(34)

We can get the sample estimator of the cost function as follows:

\[
\hat{J}_{KL}(\sigma) = \frac{1}{N} \sum_{i=1}^{N} \log \left( \frac{1}{N} \sum_{j=1}^{N} G(e_i - e_j) \right).
\]

(35)

Taking the derivative of \( J_{KL}(\sigma) \) with respect to \( \sigma \) yields

\[
\frac{\partial J_{KL}(\sigma)}{\partial \sigma} = E\left[ \sum_{i=1}^{N} \exp\left(\frac{-(e - e_i)^2}{2\sigma^2} - \frac{1}{\sigma} \right) \right].
\]

(36)

Using the above equation and a stochastic approximation of the gradient by dropping the expectation operator, we can obtain the iterative approximate solution of kernel size as

\[
\sigma(n + 1) = \sigma(n) + \eta \frac{\partial J_{KL}(\sigma(n))}{\partial \sigma(n)}
\]

\[
+ \eta \left[ \sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)} - \frac{1}{\sigma(n)} \right) \right] \sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)} \right)
\]

(37)

where \( L \) is the window size for estimating \( J \) and \( n \) is the current time index. In this study, we take \( L = 1 \). Therefore the adaptation of the kernel size can be updated by the simple form of a stochastic gradient as

\[
\sigma(n + 1) = \sigma(n) + \eta \left(\frac{(e(n) - e(n - 1))^2}{\sigma^2(n)} - \frac{1}{\sigma(n)} \right). \]

(38)

During iterative solution of the gain function \( (G) \), the ideal situation is that the value of each element of the gain function is no longer changed. Based on statistical theory, this will lead to the minimum of the difference in the standard deviations of gain functions \( G' \) and \( G' + 1 \). In this study, the condition for terminating the iteration is that the difference in the standard

\[
\]
deviations of gain functions $G^r$ and $G^{r+1}$ is less than the tolerance $\gamma = 10^{-5}$. Thus, in the iterative solution of the gain function ($G$), the standard deviation of $G$ can be used as the error $\varepsilon$ in Equation (38).

3.4. Flowchart of the Algorithm

The detailed description of our algorithm is as follows:

Initialization. Set the initial kernel size $\sigma = 1.5$, tolerance $\gamma = 10^{-5}$, gain function $G^0 = 0$, and boundary $w$.

Step 1. Calculate $\varepsilon_i(x - \delta_i), q_i(x - \delta_i)$ from Equation (28) and calculate $\varepsilon_i(x + \delta_i), q_i(x + \delta_i)$ in the same way.

Step 2. Solve Equation (31) to obtain the gain function $G^r$.

Step 3. If the difference in standard deviation between $G^r$ and $G^{r+1}$ is less than $\gamma$ go to Step 5, else go to Step 4.

Step 4. Calculate the standard deviation of $G^{r+1}$ and update the kernel size using Equation (38), then go to Step 1.

Step 5. Obtain the final gain function $G$.

4. RESULTS FROM SIMULATED DATA

In order to test our algorithm, we generate a set of simulated images of 513 × 513 pixels and compare with the methods of Chae and KLL. We use an image4 (Figure 1(a)) of a solar active region at the Hα-0.25 Å on 2015 September 27 from the Big Bear Solar Observatory (BBSO) as the raw image, which has a size of 2048 × 2048. The central part of the raw image is

Figure 6. Nine images of a solar active region at the Hα-0.25 Å from the month of 2015 September from the BBSO. The sizes of images are reduced to 256 × 256.

4 http://bbsoweb.bbso.njit.edu/pub/archive/2015/09/27/bbso_halph_fr_20150927_220002.fts
Figure 7. Left: the accuracy of the three algorithms with a month’s images of a solar active region at the Hα-0.25 Å in 2015 September from the BBSO. The mean of the standard error of KLL’s method, Chae’s method, and our method is 0.2461, 0.2889, and 0.2268 respectively. Right: the computing time for the three algorithms with these data. The mean computing times of KLL’s method, Chae’s method, and our method are 4.9384, 1.6194, and 0.8532 s respectively. The “Date” axis represents the date of 2015 September. In this test, there is no added noise and the initial value of kernel size in our algorithm is 1.5. The prime displacements are (0, 0), (50, −14), (40, 17), (21, 43), (−5, 56), (−28, 35), (−44, 7), (−51, −24), (−27, 40), (5, −43), (37, −36).

used as the object image (Figure 1(b)) and it has a size of 513 × 513. The gain function (flat field) (Figure 1(d)) is similarly simulated by using the image5 (Figure 1(c)) of flat field frames on 2000 June 19 from the BBSO, and the size of the gain function is 513 × 513.

Then we choose the spatial distribution of the relative displacements. The displacements are (0, 0), (0, 30), (0, −30), (30, 0), (−30, 0), (0, 50), (0, −50), (50, 0), (−50, 0), (0, 70), (0, −70). We generate multiple observed images with the displacements by integrating shifts of the object image (Figure 1(b)) and multiplying the shifted object and the gain function as in Equation (3). Figure 2 shows four of the observed images with four different displacements. One measure of the relative success of the algorithm is the standard deviation

\[
\text{std} = \sqrt{\frac{\sum (G'(x) - G_{\text{true}}(x))^2}{N}}
\]  

(39)

where \(G'(x)\) is the solution of our algorithm for the iteration \(r\), \(G_{\text{true}}(x)\) is the logarithm of the true gain function, and \(N\) is the number of pixels.

Before the flat-fielding, it is worth paying close attention to the displacements. It is important to not share any common factors because that may introduce a geometric artifact (Kuhn et al. 1991). But what happens if we choose the displacements with common factors? Figure 3 shows the difference between the flat field calculated by using the relative prime displacements (guaranteeing no common multiples) (left) Toussaint et al. 2003 and the flat field calculated by using the displacements that shared common factors (right). The geometric artifact is clearly visible on the right. But it is unacceptable to introduce a geometric artifact into the data with a common factor. Luckily, we can easily avoid this by carefully choosing the displacements. For our test, we use the resulting points of the relative displacements, which are primes in pixel units: (50, −14), (40, 17), (21, 43), (−5, 56), (−28, 35), (−44, 7), (−51, −24), (−27, 40), (5, −43), (37, −36) (Chae 2004).

Figure 4 shows the differences in the convergence of the three tested algorithms. KLL’s method has higher accuracy and faster convergence than Chae’s, and our algorithm is better than KLL’s. This means that, with the same accuracy, the number of iterations required by our algorithm is the fewest. With the same number of iterations, the accuracy of our algorithm is the highest. Obviously, increasing the number of iterations will increase accuracy.

Figure 5 demonstrates the anti-noise capability of the three algorithms. The left panel of Figure 5 shows the anti-noise capability with Gaussian noise. With increasing noise intensity, the performance of these algorithms decreases. And KLL’s method is better than Chae’s method. Furthermore, our algorithm has the best anti-noise capability with Gaussian noise. The solution of KLL’s method and Chae’s method is an approximate SSE solution. As we know, SSE has a strong anti-noise capability with Gaussian noise, but a correctly tuned MCC is not inferior to the SSE for large amounts of Gaussian noise. The right panel of Figure 5 shows the anti-noise capability with salt-and-pepper noise. The performance of KLL’s algorithm is almost the same as that of Chae’s algorithm. With increasing noise intensity, the performance of KLL’s algorithm and Chae’s algorithm decreases. And the noise has little effect on our algorithm. Moreover, it is clearly seen that the performance of our algorithm is very good. We also test the anti-Poisson noise capability of the three algorithms. The standard deviations of KLL’s method, Chae’s method, and our method are 0.0587, 0.0602, and 0.0551 respectively. Obviously, our method can be applied for Poisson noise. In summary, our method has a strong anti-noise capability with typical Gaussian and non-Gaussian noises.

In the above experiments, the simulated object images are the central parts of the raw images. If the whole raw image is used as the object image, computing the gain function from the full-sized flat-field frames would take a lot of computing time. Therefore, we reduce the size of the centered and shifted

5 http://www.bbso.njit.edu/Research/FDHA/data/hd61900f.jpg
Figure 8. (a) One centered solar image on 2000 June 19 from the BBSO. (b) One of the eight solar images shifted with respect to the center of the camera. The shifted frames are obtained by off-setting the telescope from the center of the Sun. (c) The calculated flat field and (d) the flat-field image corrected by using our method. (e) The calculated flat field and (f) the flat-field image corrected by using KLL’s method. The number of iterations in KLL’s method is 15.
calibration frames, which are 256 × 256. In this test, we use a month’s images of a solar active region at the Hα-0.25 Å in 2015 September from the BBSO as the object images. Figure 6 shows nine of these images. The difference among these images is clear. Affected by weather and other factors, we have only 22 sets of available full-disk Hα FITS images with valid data. Thus, the data distribution in Figure 7 is not uniform. The left panel of Figure 7 shows the accuracy of the three algorithms with these images. The mean of the standard error of KLL’s method, Chae’s method, and our method is 0.2461, 0.2889, and 0.2268 respectively. This shows that our algorithm has the highest accuracy. Moreover, our algorithm is stable. The right panel of Figure 7 shows the computing times of the three algorithms with these data. The mean computing times of KLL’s method, Chae’s method, and our method are 4.9384, 1.6194, and 0.8532 s respectively. This clearly shows that our algorithm saves computing time.

5. RESULTS FROM REAL DATA

We applied our algorithm to a set of images of nine displacements at the BBSO on 2000 June 19. Note that the typical exposure time for obtaining the Hα full-disk images at BBSO is 30 ms. A full-disk image usually remains the same during the observation. Figure 8(a) shows one centered solar image. Figure 8(b) shows one of the eight solar images shifted with respect to the center of the camera.

Figures 8(c) and (d) are the calculated gain function and the corrected object image of our algorithm. Figures 8(e) and (f) are the calculated gain function and the corrected object image after 15 iterations of KLL’s algorithm. They show that the gain function calculated by KLL’s method is non-uniform. The contrast of Figure 8(e) is low and the feature is not clear. But the feature is obvious with our method. There is a fan-shaped pattern in Figure 8(c). Moreover, the center of Figure 8(f) is too light and the feature is not clear.

6. CONCLUSION

A new method is presented to calculate the gain function from a set of relatively shifted images of a non-uniform object. Unlike KLL’s method and Chae’s method, we use the MCC to obtain an iterative solution. Compared with KLL’s algorithm and Chae’s algorithm, ours has the highest accuracy and fastest convergence. As a result, we found that our algorithm effectively suppresses noise, especially in the case of typical non-Gaussian noise. In addition, its computing time is the least.

The authors would like to thank the Global High Resolution Hα Network, operated by the Big Bear Solar Observatory, New Jersey Institute of Technology. This work was supported by the National Natural Science Fund Committee and the Chinese Academy of Sciences astronomical union funds under Grant U1331113, 2014FY120300.

Facility: Big Bear Solar Observatory.

REFERENCES

Boyd, S., & Vandenberghe, L. 2004, Convex Optimization (Cambridge: Cambridge Univ. Press)
Chae, J. 2004, SoPh, 221, 1
Denker, C., Johannesson, A., Marquette, W., et al. 1999, SoPh, 184, 87
Kuhn, J. R., Lin, H., & Loranz, D. 1991, PASP, 103, 1097
Liu, W., Pokharel, P. P., & Principe, J. C. 2006, in Int. Joint Conf. on Neural Networks, 4919
Liu, W., Pokharel, P. P., & Principe, J. C. 2007, ITSP, 55, 5286
Principe, J. C. 2010, Information Theoretic Learning: Renyi’s Entropy and Kernel Perspectives (Berlin: Springer)
Rockafellar, R. T. 1997, Convex Analysis (Princeton, NJ: Princeton Univ. Press)
Stetson, P. B., & Harris, W. E. 1988, AJ, 96, 909
Toussaint, R. M., Harvey, J. W., & Toussaint, D. 2003, AJ, 126, 1112
Vapnik, V. 1995, The Nature of Statistical Learning Theory (New York: Springer)
Yan, H., Yuan, X., Yan, S., & Yang, J. 2011, Pattern Recognit., 44, 2834
Yang, G., Xu, Y., Wang, H., & Denker, C. 2003, ApJ, 597, 1190