New Astronomy 48 (2016) 49–57

Contents lists available at ScienceDirect

New Astronomy

journal homepage: www.elsevier.com/locate/newast

Detection of coronal mass ejections using AdaBoost on grayscale statistic features☆

Ling Zhang[a,b], Jian-qin Yin[a,b,*], Jia-ben Lin[b,**], Xiao-fan Wang[b], Juan Guo[b]

[a] School of Information Science and Engineering, University of Jinan, Shandong Provincial Key Laboratory of Network Based Intelligent Computing, Jinan 250022, China
[b] China Academy of Science, Key Laboratory of Solar Activity, Peking 100012, China

HIGHLIGHTS

- The grayscale statistic features of the CME image are analyzed.
- We model the problem of CME detection as a classification problem.
- AdaBoost is used to detect CMEs.

ARTICLE INFO

Article history:
Received 19 August 2014
Revised 8 April 2016
Accepted 9 April 2016
Available online 19 April 2016

Keywords:
AdaBoost
Coronal mass ejection
Solar activity

ABSTRACT

We present an automatic algorithm to detect coronal mass ejections (CMEs) in Large Angle Spectrometric Coronagraph (LASCO) C2 running difference images. The algorithm includes 3 steps: (1) split the running difference images into blocks according to slice size and analyze the grayscale statistics of the blocks from a set of images with and without CMEs; (2) select the optimal parameters for slice size, gray threshold and fraction of the bright points and (3) use AdaBoost to combine the weak classifiers designed according to the optimal parameters. Experimental results show that our method is effective and has a high accuracy rate.

1. Introduction

Coronal mass ejections (CMEs) are massive eruptions of coronal material from the solar atmosphere containing plasma threaded by magnetic fields (DeForest et al., 2013; Howard et al., 1985; Hundhausen, 1993). These CMEs have close connections to many planetary disturbances and may influence the environment of the Earth. The detection of CMEs is not only important to solar physics but invaluable to space weather prediction (Robbrecht and Berghmans, 2006).

The traditional way of CME detection is based on human observation which is inefficient and easily influenced by individual subjective factors. Automatic identification technology has evolved rapidly in recent years. Researchers use image segmentation techniques to find CMEs in running difference and running ratio images and further classify CMEs into different categories (Qu et al., 2006). Some software applications were developed to autonomously detect CMEs in image sequences from LASCO, using Hough transform (Berghmans et al., 2002). The method was further improved by using clustering and morphological closing operations to mark out different CMEs (Robbrecht and Berghmans, 2004). Other useful tools were released to measure some important parameters and classify CMEs (Olmedo et al., 2008). Texture was employed to detect the CME region and track the event by applying segmentation to the corresponding time series of coronagraph image. An algorithm based on level set and region competition to detect and track the CMEs has been presented (Goussies et al., 2010, 2011). Some artificial intelligence technologies including wavelets, curvelets and ridgelets were introduced in the detection of CMEs to help suppress noises and outline the characteristics of position, velocity, and acceleration (Gallagher et al., 2011). Recently Byrne et al. reinforced the tracking technique by determining the dynamic CME separation with a multiscale edge-detection algorithm in their new system CORIMP (Byrne et al., 2012). Morgan et al. described a method of deconvolution to separate coronagraph images into quiescent and dynamic components, which could be applied to cases with a weak CME signal (Morgan et al., 2012).

☆ This work was supported partly by National Natural Science Foundation of China (Grant nos. 61203341, 61375084), and the Open Foundation of Solar Activities of Key Laboratory of Chinese Academy of Sciences (No. TQT1302).

* Corresponding author at: School of Information Science and Engineering, University of Jinan, Shandong Provincial Key Laboratory of Network Based Intelligent Computing, Jinan 250022, China, Tel.: +86 13658613338.
** Corresponding author.
E-mail addresses: ise_yinjq@ujn.edu.cn (J.-q. Yin), jiaben.lin@163.com (J.-b. Lin).

http://dx.doi.org/10.1016/j.newast.2016.04.002
1384-1076/

Several methods of CMEs detection have been presented. The techniques used in each detection algorithm are different from the others. However, nearly all of them deal with the following two aspects: feature selection and classifier design. Most techniques use a single classifier in their algorithm. Since the CME varies greatly in shape and spatial scale, the use of a single classifier is insufficient. At the same time, the concept of AdaBoost formulated by Yoav Freund and Schapire has been successfully employed in several classification related problems (Schapire et al., 1998). It can be used in combination with many types of learning algorithms to improve their performance. The output of each algorithm ('weak learner') is combined into a weighted sum that represents the final output of the boosted classifier. In this article, we attempt to detect CMEs using AdaBoost.

The automatic CME detection algorithm proposed here has three critical steps: (1) splitting the running difference image into blocks according to slice size and then analyzing the grayscale statistic of the brightest blocks of the CME images; (2) designing an algorithm to obtain the optimal parameters to detect the CME images, and (3) creating a strong classifier using AdaBoost to adjust the weight factor of each weak classifier and subsequently using the classifier to detect the CME in running difference images. These three steps are described in details in Section 2–4, respectively. Conclusions and future work are discussed in the final section.

2. Analysis of grayscale statistics in CMEs blocks

From the perspective of human vision, the regions of CMEs are usually brighter than other areas in running difference images. So the key of using a computer to simulate human vision is to find bright regions in an image. Before describing the algorithm imitating human vision to detect CMEs, we introduce several terms below:

Term 1. Bright point: Pixel with a high grayscale intensity.

Term 2. Slice size: Small blocks, which are either square or rectangular, having a width w and height h.

Term 3. Bright block: Bright block is one that has a higher fraction of bright points above a threshold p. It is most likely to be a CME region in an image.

In order to find regions with the highest intensity, we split the image into blocks. If there are one or more bright blocks in the image, after splitting the running difference image, there is a strong likelihood that a CME is present. As the scales, shapes and intensities of the CME regions in different images are different, it is difficult to detect all the CME regions by splitting the image using a single slice size. Moreover, selection of the grayscale threshold and the fraction of bright pixels also influence the detection results. In order to illustrate this, we provide an example below.

Firstly, the detection results are influenced by the slice size because of the variety of the shapes. The vertical CME regions are easily detected by splitting the images into vertical blocks, such as the CME region in Fig. 1(a). Likewise, the horizontal CMEs regions are easily detected by splitting the images into horizontal blocks, such as the CME region in Fig. 1(b).

In order to illustrate the influence of the slice sizes, we provide an example. The two images in Fig. 1 are split into blocks shown in Fig. 2(a(1–2)) and (b(1–2)) according to slice sizes of 10 × 25 and 25 × 10 pixels. The blocks with CMEs are marked in blue rectangles manually in the above figures. In this example, the pixels with grayscale higher than 140 are considered as bright points. The blocks with the fraction of the bright points higher than 30% are considered as bright blocks. Scatter diagrams of Fig. 2(a(3–4)) and (b(3–4)) illustrate the distribution of the fraction of bright points in the blocks. The red triangles in Fig. 2(a(3–4)) and (b(3–4)) are corresponding to the CMEs blocks in Fig. 2(a(1–2)) and (b(1–2)). The blocks without CMEs are marked with blue diamonds in Fig. 2(a(3–4)) and (b(3–4)). Obviously, when the fraction threshold is set to 30%, the horizontal CME regions can be detected by splitting the image according to the slice size of 10 × 25 pixels and the vertical CME regions can be detected by splitting the image according to the slice size of 25 × 10 pixels.

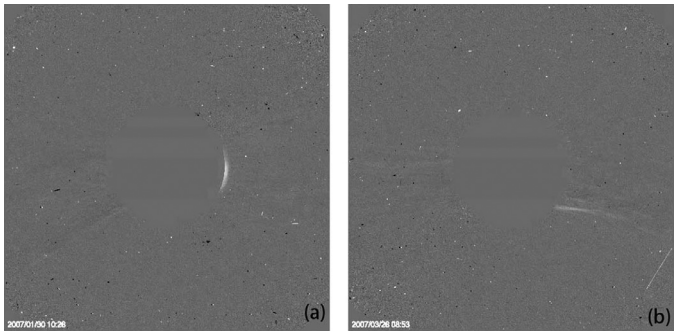

Fig. 1. (a) A running difference image with vertical CME region. (b) A running difference image with horizontal CME region.

Secondly, the value of the fraction threshold also influences the detection of the CME regions in the images. In Fig. 3, the image in Fig. 1(a) is split into blocks according to slice size of 20 × 25 pixels. The grayscale threshold is set to 140. If the fraction threshold is still 30% as Fig. 2, then there is no bright blocks, that is to say there is no CME in the image. But in fact, the blocks marked in blue are CME blocks. We can see from Fig. 3(b), that the blocks with a fraction of bright points higher than 20% are CME blocks. So if we split the image according to a slice size of 20 × 25 pixels, the fraction threshold should be set to 20%. In summary, the fraction threshold of bright points is not unique under different slice sizes.

Thirdly, the detection result is also influenced by the value of the grayscale threshold. Fig. 4 illustrates the distribution of the fractions of the bright points in the blocks of the image in Fig. 4(a) split according to a slice size of 35 × 35 pixels. To demonstrate the grayscale threshold in Fig. 4, the grayscale thresholds in Fig. 4(b) and (c) are set to 135 and 140, respectively. Under the grayscale threshold 135, there is an obvious fraction threshold to differentiate the CMEs blocks and the blocks without CMEs. But there is no such a threshold in Fig. 4(c). So it is also important to select the value of the grayscale threshold.

From the above analysis, we should find some optimal groups of parameters including suitable slice sizes, grayscale thresholds and the fraction thresholds to detect CMEs. In addition, we need to find a scheme to combine multiple detection results based on different parameters. We first design an algorithm to search for the optimal parameters for detecting a CME. AdaBoost is then employed to combine different detection results based on the multiple group parameters.

3. Searching algorithm for optimum parameters

In order to determine optimal slice sizes, grayscale thresholds and the fraction thresholds to detect CMEs, we design the following algorithm. The input of the algorithm consists of two parts: a set of running difference images, i.e., learning samples $\{L_1, L_2, \cdots, L_S\}$, where S is the size of the sample set, and a range of quantities for the following: grayscale threshold [125–255], fraction threshold [10–80%], height [5–50] and width [5–50]. The learning samples are obtained from the SOHO/LASCO website http://cdaw.gsfc.nasa.gov/. The result of manual detection is denoted by an array A (A_i is 1 if the ith sample is an image with a CME, otherwise A_i is 0.). We design a classification model for each

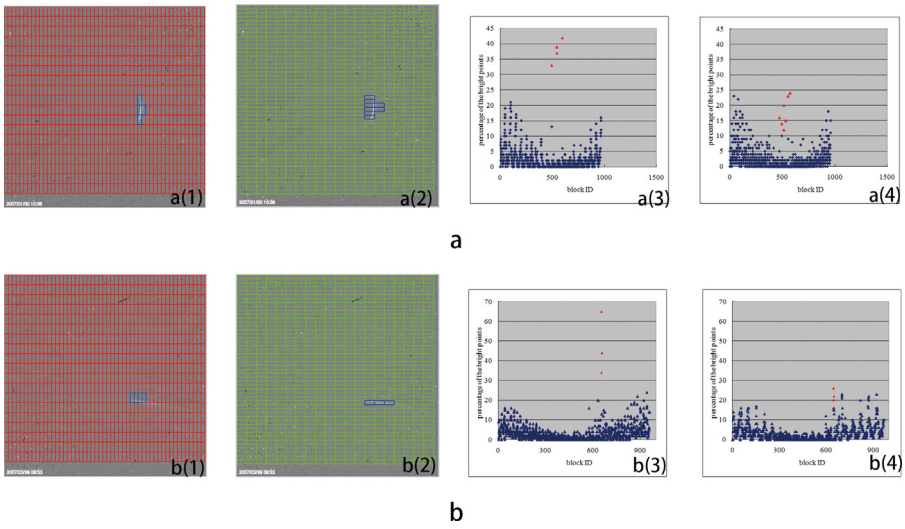

Fig. 2. (a(1–2), b(1–2)) Two running difference images split according to the slice sizes of 10×25 and 25×10 pixels. (a(3–4), b(3–4)) Scatter diagrams of the fraction of the bright points in blocks in (a(1–2)) and (b(1–2)), respectively. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

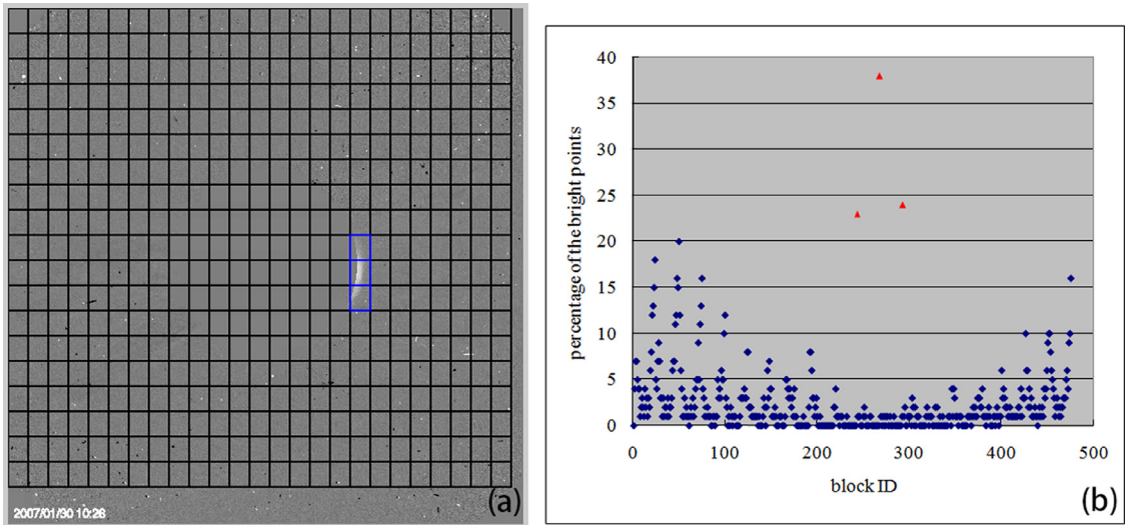

Fig. 3. (a) Running difference image split according to slice size of 20×25 pixels. (b) Scatter diagram of the fraction of the bright points in blocks in (a). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

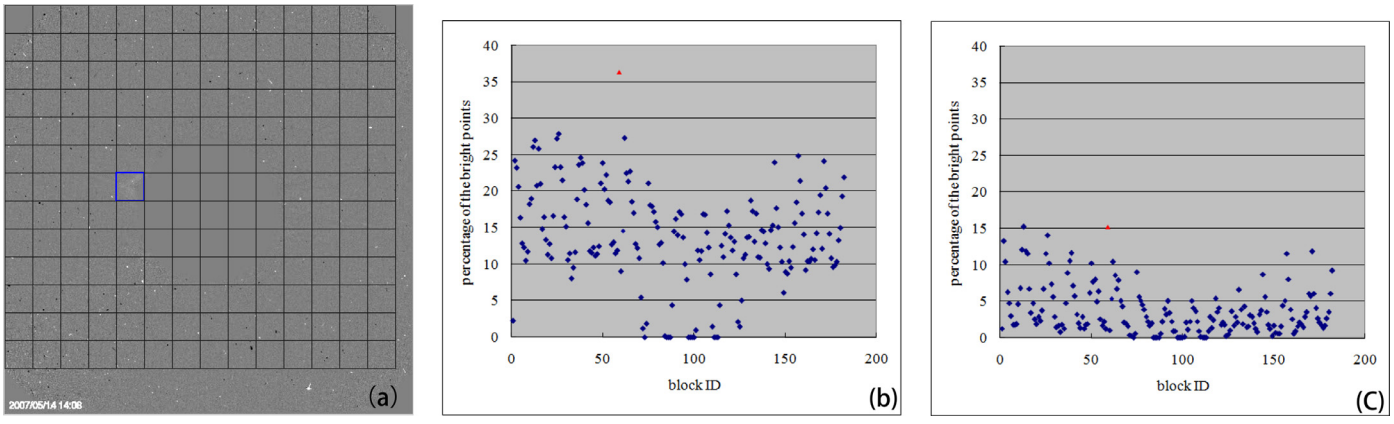

Fig. 4. (a) A running difference image split according to slice size of 35×35 pixels. (b) Scatter diagrams of the fractions of the bright points with grayscale higher than 135 in blocks in (a). (c) Scatter diagrams of the fractions of the bright points with grayscale higher than 140 in blocks in (a).

four-tuple (g, p, h, w) to classify the learning samples, where $g \in [125, 255]$, $p \in [10\%, 80\%]$, $h \in [5, 50]$ *and* $w \in [5, 50]$. Using the classification model to classify all the images and then comparing the classification result to A, we can obtain the accuracy rate. After traversal of all the tuples, the corresponding sets of tuples $\{(g_1, p_1, h_1, w_1), (g_2, p_2, h_2, w_2) \ldots (g_N, p_N, h_N, w_N)\}$ with first N highest accuracy rates will be chosen, as the set of the classifiers.

The detail of the procedure is as follows:

Firstly, by using parameters of each tuple (g, p, h, w), classify the learning samples to be images with CMEs or images without CMEs. For the ith sample image:

(1) Use a sliding box that goes pixel by pixel from left to right and bottom to top to traverse the image. During the traversal, calculate the number of the points with grayscale higher than g in the sliding box according to Eq. (1).

$$M_q = \sum_{j=1}^{w} \sum_{k=1}^{h} f(j,k) \quad q = 1, 2, \ldots, t$$
$$f(j,k) = \begin{cases} 1, & u(j,k) > g \\ 0, & else \end{cases} \tag{1}$$

where M_q is the number of the points with grayscale higher than g in the sliding box. t is the number of the sliding boxes during the traversal of the image. $u(j, k)$ is the gray value of pixel (j, k) in the qth sliding box. $f(j, k)$ is 1 if pixel (j, k) is a bright point, otherwise $f(j, k)$ is 0.

(2) Classify the ith sample image according to Eq. (2). The detection result is denoted by D_i.

$$D_i = \begin{cases} 1, & M_q/(h * w) > p \ \ \exists q \in [1, t] \\ 0, & else \end{cases} \tag{2}$$

In Eq. (2), D_i is 1 if the ith sample is detected with CME, otherwise D_i is 0. That is, the detection result of the sample depends on whether there exist blocks detected with CME.

The flowchart is shown in Fig. 5.

Secondly, calculate the accuracy rate of the detection method based on the tuple (g, p, h, w).

$$a(g,\ p,\ h,\ w) = \left(\sum_{i=1}^{S} F_i\right)/S$$
$$F_i = \begin{cases} 1 & if\ D_i = A_i \\ 0 & else \end{cases} \tag{3}$$

where F_i denotes the detection result and S denotes the size of the sample set. If the detection result is right, F_i is 1; otherwise F_i is 0. $a(g, p, h, w)$ represents the accuracy rate of the detection method under the tuple (g, p, h, w).

Finally, sort the array a and output the N tuples $\{(g_1, p_1, h_1, w_1), (g_2, p_2, h_2, w_2) \cdots (g_N, p_N, h_N, w_N)\}$ with the top N accuracy rates.

The components of the tuples with the top 20 accuracy rates are given in Table 1.

4. The detection algorithm using AdaBoost

In the field of artificial intelligence, CME detection can be regarded as a classification problem, of a binary nature, namely identifying two classes in which CMEs exist and do not exist, respectively. These two classes are represented by 1 and 0, respectively. With the N sets of parameters obtained previously, we can design N weak classifiers according to Eqs. (1) and (2). The N weak classifiers are combined into a weighted sum, namely the final classifier that can be used to detect the CME pixels in a given image. The determination of the weights is carried out with a set of sample images which serve as the learners. The following describes the AdaBoost algorithm for determining the weights of the classifiers.

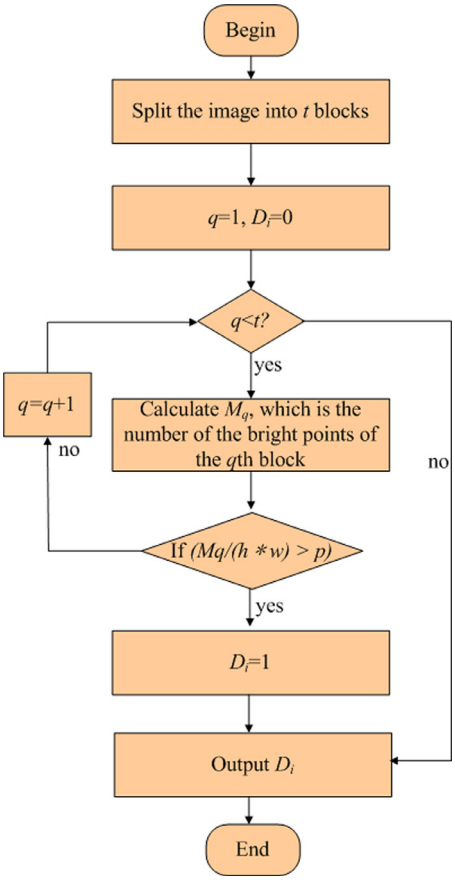

Fig. 5. Flowchart of the algorithm to find the optimal tuples.

Input: N weak classifiers $\{C_1, C_2, \cdots, C_N\}$ (C_i is the weak classifier related to the tuple (g_i, p_i, h_i, w_i)), training samples $\{L_1, L_2, \cdots, L_S\}$, where S is the size of the sample set, weight array W (W_j is the weight of the jth sample initialized to $1/S$), array A (A_j is 1 if L_j is the image with CME, otherwise A_j is 0.)

Output: Final strong classifier.

Algorithm representation:

Firstly, N weak classifiers are used to classify the training sample subset. The classification procedure using the ith weak classifier is designed as follows:

(1) Select the training subset $\{L_1, L_2, \cdots, L_f\}$ from the training samples by using the roulette wheel strategy (Goldberg, 1989), where f is the size of the subset. This training subset will be used to determine the weight of the ith classifier.

(2) Detect CME in each training sample image. The jth sample's detection result is denoted as $D(i, j)$, which can be calculated according to Eq. (4).

$$D(i,j) = \begin{cases} 1, & \exists\, C_i(B_q) = 1 \quad q = 1 \cdots t \\ 0, & else \end{cases} \tag{4}$$

where $D(i, j)$ is 1 if the image is classified with CME by the using ith weak classifier, otherwise $D(i, j)$ is 0. B_q denotes the qth block of L_j. $C_i(B_q)$ denotes the detection result of the qth block in the image. $C_i(B_q)$ is 1 if B_q is classified as CME block, otherwise $C_i(B_q)$ is 0. The value of $C_i(B_q)$ can be calculated

Table 1
Optimal tuples with the top 20 accuracy rate.

Fraction threshold	Grayscale threshold	Width	Height	Accuracy rate
0.3	141	25	15	87.7%
0.3	139	25	20	87.2%
0.3	139	30	15	87.1%
0.4	139	20	15	87.1%
0.4	141	10	15	87.1%
0.6	139	10	10	87.1%
0.4	141	15	15	86.9%
0.5	139	15	10	86.9%
0.4	139	15	20	86.3%
0.4	139	25	10	86.3%
0.4	141	20	10	86.3%
0.3	139	40	15	86.1%
0.3	141	10	30	86.1%
0.3	141	15	25	86.1%
0.3	141	20	20	86.1%
0.3	141	35	10	86.1%
0.4	139	10	20	86.1%
0.3	141	30	10	85.9%
0.3	139	35	15	85.8%
0.3	139	40	10	85.8%

from Eq. (5).

$$C_i(B_q) = \begin{cases} 1, & if \quad M_q/(h_i * w_i) > p_i \\ 0, & else \end{cases} \tag{5}$$

where M_q can be calculated according to Eq. (1).

(3) Calculate the accuracy rate of the ith weak classifier using Eq. (6).

$$R_i = \left(\sum_{j=1}^{f} F_j\right)/f$$
$$F_j = \begin{cases} 1 & if \quad D(i,j) = A_j \\ 0 & else \end{cases} \tag{6}$$

where R_i is the accuracy rate of the ith weak classifier.

(4) Adjust the weight of each sample in the training subset according to Eq. (7). Normalize the weight of each sample in the training set by Eq. (8).

$$W_j = \begin{cases} W_j * e^{-\beta_i} & if \quad D(i,j) = A_j \\ W_j * e^{\beta_i} & else \end{cases}$$
$$\beta_i = \frac{1}{2} \ln \frac{R_i}{1 - R_i} \tag{7}$$

$$W_j = W_j / \left(\sum_{k=1}^{S} W_k\right) \tag{8}$$

where W_j represents the weight of the jth sample image.

Secondly, normalize the weight of each weak classifier α_i as

$$\alpha_i = R_i / \left(\sum_{k=1}^{N} R_k\right) \tag{9}$$

The final strong classifier can be designed according to Eq. (10).

$$F(x) = \begin{cases} 1 & \left(\sum_{i=1}^{N} C_i(B_q) * \alpha_i\right) > 0.5 \quad \exists q = 1 \ldots t \\ 0 & else \end{cases}$$
$$\sum_{i=1}^{n} \alpha_i = 1 \tag{10}$$

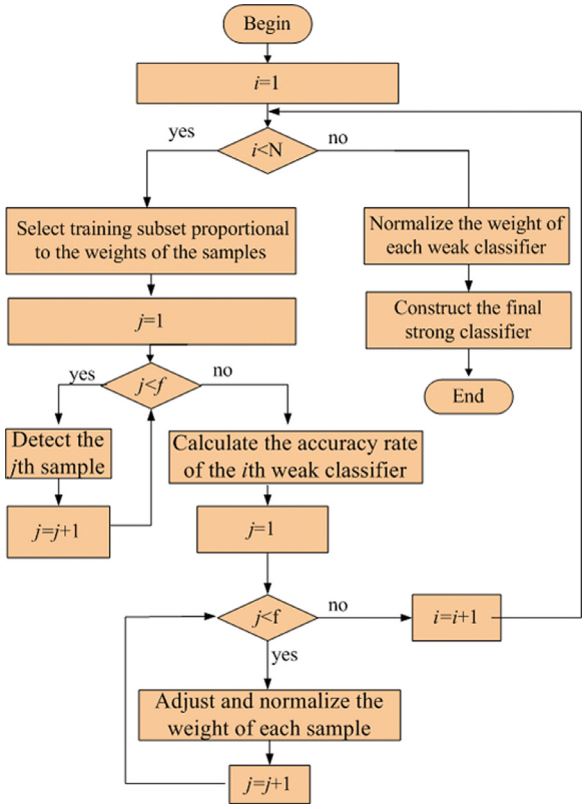

Fig. 6. Flowchart of the construction procedure of the strong classifier.

$F(x)$ means the final detection result of the image x. $C_i(B_q)$ is the detection result of the qth block in x using the ith weak classifier.

The flowchart of the algorithm is shown in Fig. 6 and the procedure to detect the image is shown in Fig. 7. In order to show the procedure clearly, only bright blocks split according to slice sizes of 30 × 15, 18 × 6 and 14 × 6 pixels are shown in green, red and white colors respectively.

Using the final classifier, we now proceed to detect the CME in a running difference image. Using the parameters for each classifier, a sliding box centered on a pixel of interest is evaluated to be a CME pixel or otherwise by invoking Eq. (10). The sliding box

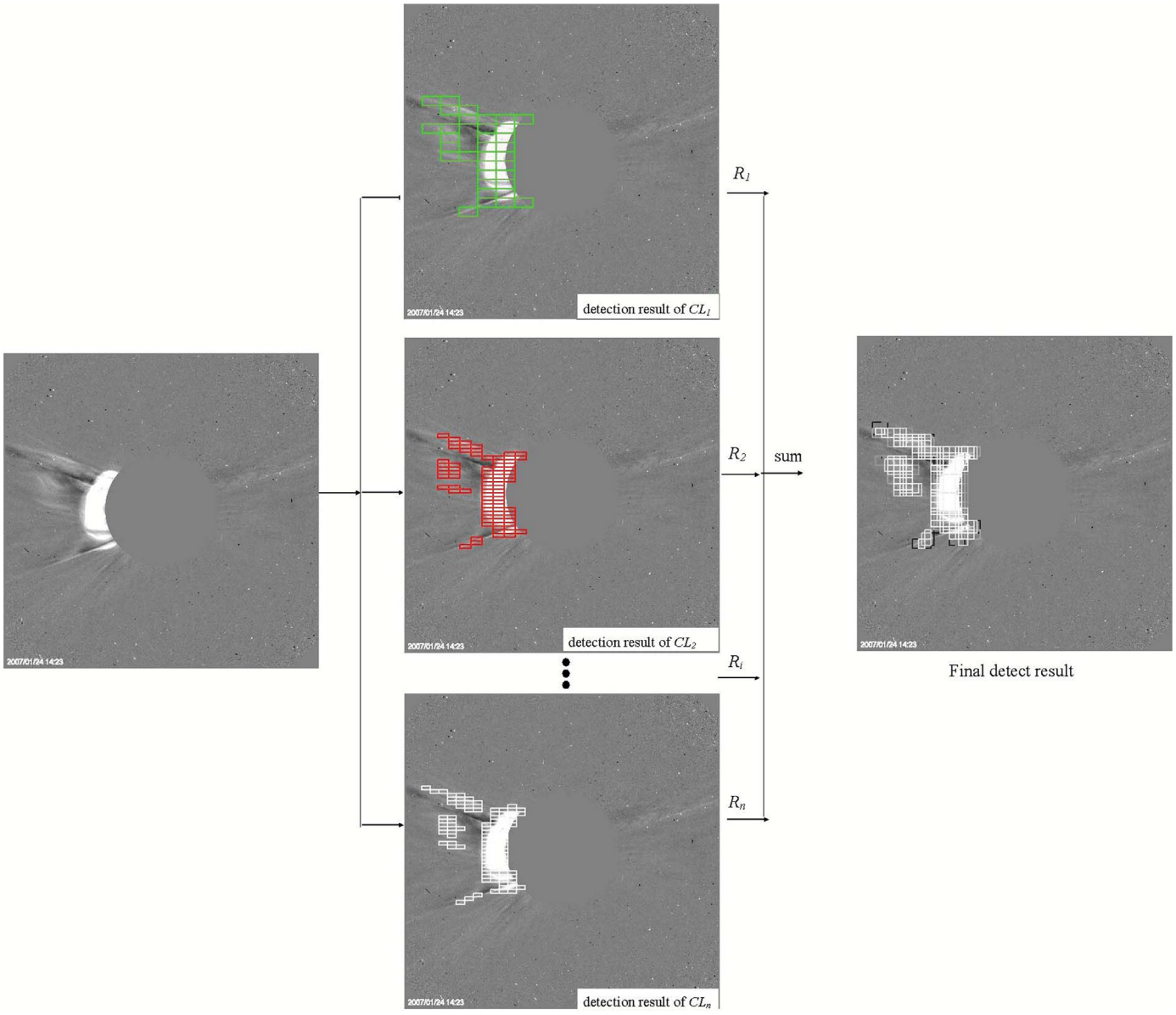

Fig. 7. Procedure of detection.

traverses left to right and top to bottom of the image and in the process, the presence or absence of a CME in that image can be determined.

There is another problem that we need to pay attention. The accuracy rate of the final classifier is influenced by the label of the learning sample. In order to determine the influence of the incorrectly labeled samples, we carry out an experiment. The learning sample set is denoted by L. The label of the ith sample is denoted by A_i. The test samples are denoted by T. We discuss these experiments from the following three aspects.

(1) Select some learning samples and swap their labels. That is to say, we replace A_i with 1 if A_i is 0, and vice versa. The wrongly labeled samples are selected by the roulette wheel strategy according to levels of 10%, 20%, 40%, 60%, and 80%, respectively.
(2) Search for the optimal tuples to design weak classifiers and the final classifier. In each experiment, 20 tuples with top 20 accuracy rates to classify the labeled samples including the wrong labeled samples are selected by the searching algorithm to combine the final classifier. Table 2 shows the result of each experiment. Due to space constraints, only 7 tuples with top 7 accuracy rates are given in Table 2. The 7 rows show the results of one experiment in a single column. Each weak classifier in each experiment is given in the format m–n in the first column, where m is the number of the experiment, and n is the number of the weak classifier. The next 4 columns are optimal parameters obtained by the searching algorithm. Each row of parameters is used to design one weak classifier. The sixth column gives accuracy rates of the weak classifier on L. The accuracy rate of the final classifier of each experiment on test samples T is given in the last column.
(3) Analyze the influence of the wrongly labeled samples. In Table 2, we can see that when the wrongly labeled samples are less than 50%, the accuracy rates of the first 3 final classifiers are all above 90%. But once the wrongly labeled samples exceed 50%, the accuracy rates of the last two final classifiers are lower than 30%.

Table 2
Parameters and accuracy rates of weak classifiers and the final classifier.

No.	Fraction threshold	Grayscale threshold	Width	Height	Accuracy rate of weak classifier	Accuracy rate of final classifier
1–1	0.3	141	25	15	80.03%	
1–2	0.3	141	30	15	79.23%	
1–3	0.5	141	15	10	79.23%	
1–4	0.3	137	40	20	78.75%	95%
1–5	0.4	141	25	10	78.75%	
1–6	0.3	141	20	20	71.25%	
1–7	0.3	141	40	10	71.09%	
2–1	0.3	141	15	25	72.20%	
2–2	0.3	141	15	30	71.73%	
2–3	0.5	141	10	15	71.73%	
2–4	0.4	141	10	25	71.57%	95%
2–5	0.3	141	10	40	71.25%	
2–6	0.6	137	10	20	71.25%	
2–7	0.3	141	20	20	71.09%	
3–1	0.3	141	15	25	59.42%	
3–2	0.4	137	20	10	59.42%	
3–3	0.4	133	35	20	59.27%	
3–4	0.4	141	20	15	59.27%	94%
3–5	0.4	133	30	35	59.11%	
3–6	0.5	137	25	5	59.11%	
3–7	0.3	141	15	30	58.95%	
4–1	0.2	141	5	15	53.51%	
4–2	0.3	133	5	35	53.51%	
4–3	0.4	141	5	5	53.51%	
4–4	0.5	129	40	5	53.51%	20%
4–5	0.6	129	15	5	53.51%	
4–6	0.6	129	10	10	53.51%	
4–7	0.1	125	5	5	53.35%	
5–1	0.2	141	15	5	56.71%	
5–2	0.4	141	5	5	56.71%	
5–3	0.1	125	5	5	56.71%	
5–4	0.1	125	5	10	56.55%	15%
5–5	0.1	125	5	15	56.55%	
5–6	0.1	125	5	20	56.55%	
5–7	0.1	125	5	25	56.55%	

In the first three groups, the 20 tuples obtained by the searching algorithm are the optimal tuples with the highest accuracy rates to differentiate all the labeled learning samples. Since the majority of the learning samples are correctly labeled, it implies that most samples with CMEs have features of high intensity while those labeled without CMEs do not have bright blocks. Hence, the optimal parameters obtained by the searching algorithm are the tuples that can differentiate the majority of the samples correctly. In addition, Table 2 shows that the tuples obtained in the first 3 experiments are very similar. In other words, the minority of incorrectly labeled samples do little to the selection of the optimal parameters. However, these wrongly labeled samples can decrease the accuracy rate of the weak classifiers designed according to the parameters. The reason is that the wrongly labeled samples will be misclassified. This explains why the accuracy rate of the weak classifiers decreases with the increase of the number of the wrongly labeled learning samples. Finally, we see that the accuracy rates of the first 3 groups are similar. This is easy to understand because the weak classifiers generated by the searching algorithm are similar. Moreover, the correctly labeled learning samples dominate the wrongly labeled ones, and as such AdaBoost increases the performance of the base weak classifiers under the constraints of the above learning samples. Both the good base classifier and majority of rightly labeled learning samples contribute to the high accuracy rate.

In the last two groups, more than half the samples are wrongly labeled. The searching algorithm selects the optimal tuples to classify these labeled samples, among which a majority are wrong. So the optimal solution corresponds to those parameters which can be used to correctly classify the majority of wrongly labeled samples with a high probability. From Table 2, we can see that from the fourth and fifth experiment, and we choose the fourth one as an example. In this experiment, the first three tuples are not reliable. Considering the high intensity characteristic of the CMEs, the grayscale threshold seems good. But if we consider it with the slice size and the fraction threshold together, it is easy to see that these parameters are unreliable. The slice size is small or the fraction threshold is low, so this case may correspond to noise or the CME. The other problem is that the grayscale thresholds of the left tuples are not higher than 129, which is the mean value of all pixels in the entire image. Consequently the test samples without CMEs are classified as samples with CMEs by using the combined detection results of the multiple weak classifiers designed according to the tuples that include components with low grayscale thresholds. This explains why the accuracy rate of this test set is very low.

To summarize, as long as more than half the samples are labeled correctly, the final classifier works well due to the advantages of the searching algorithm and the boosting method.

5. Analysis of experimental results

The running difference image sample sets were obtained from the LASCO website http://cdaw.gsfc.nasa.gov/CME_list/ in order to test the performance of our algorithm. Accuracy rates of the weak and final classifier were obtained at the end. Some results are shown at http://cmedetection.blog.com/. The results were used to make comparisons with other existing catalogs.

5.1. Accuracy rate of the detection method

We chose 1551 samples from January 2007 as training samples to design the weak classifiers and adjust their weights. We used

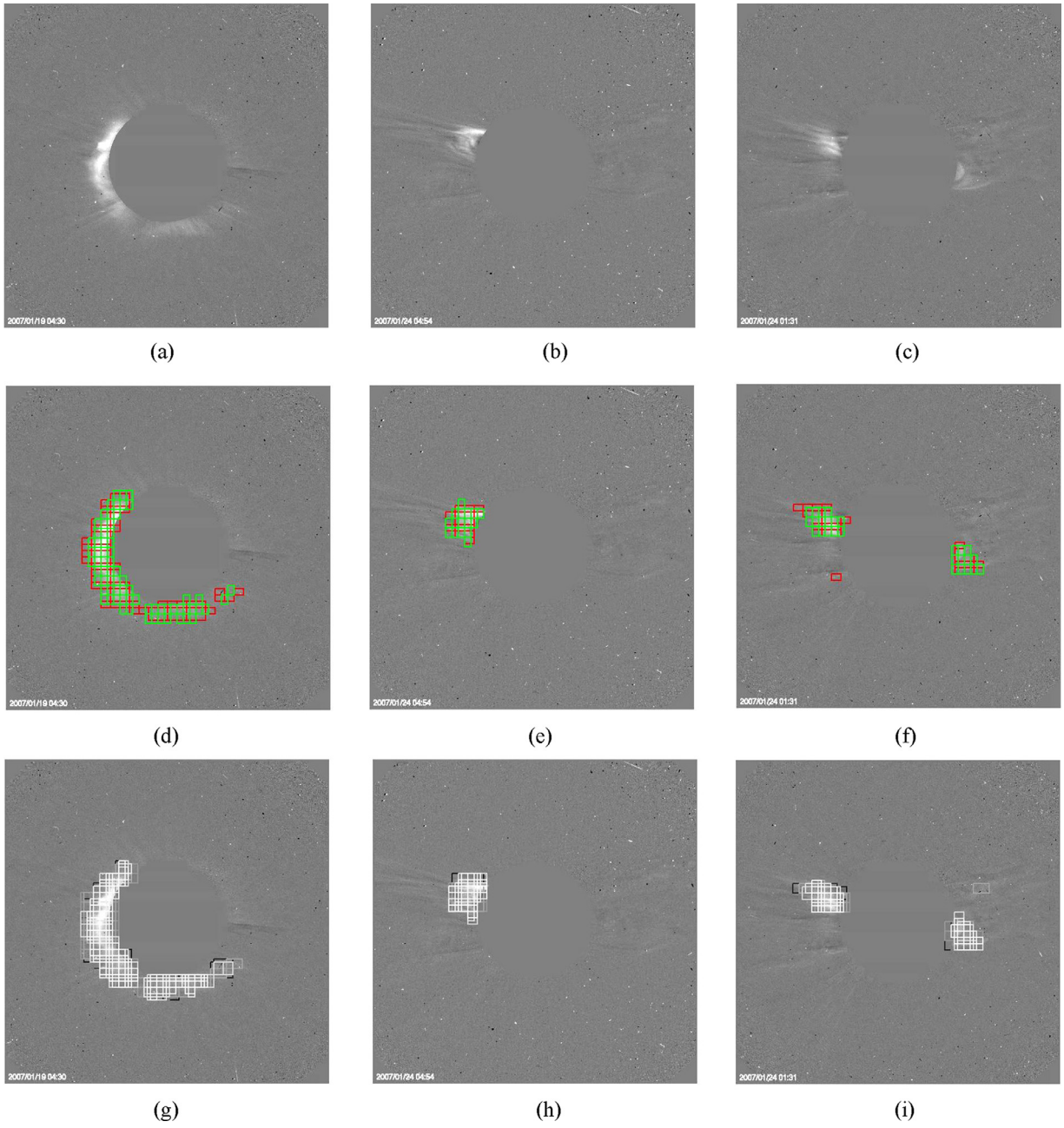

Fig. 8. (a–c) Three LASCO C2 running difference images. (d–f) Bright blocks detected by the weak classifiers with slice sizes of 10 × 15 and 15 × 10 pixels. (g–i) Final detection results of the algorithm using AdaBoost. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

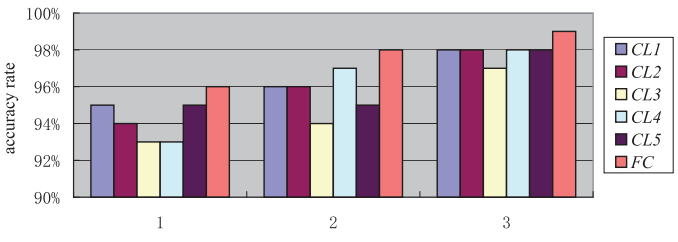

Fig. 9. Comparison of the accuracy rates between the weak classifiers and the final classifier.

Table 3
Accuracy rates of the final classifiers combined by different numbers of weak classifiers.

Number of the weak classifiers	Accuracy rate of the final classifier
5	92
10	93
20	96
30	96
40	96

data from May 1 to May 31, 2007 to test the final classifier. The classifiers were chosen according to their accuracy rates of detection. Experimental results (Table 3) show that the accuracy rate of the final classifier is related to the number of weak classifiers. The accuracy rate of the strong classifier increases with the number of the weak classifiers, but does not improve beyond 20.

Some detection results using the final classifier combining 20 weak classifiers are shown in Fig. 8. The CME regions in Fig. 8(a–c), detected by two weak classifiers, are marked with different colors in Fig. 8(d–f), respectively. The red rectangles are the detection results using a weak classifier with a slice size of 15 × 10 pixels, and regions marked green are those detected by the weak classifier with a slice size of 10 × 15 pixels. The final detection results are given in Fig. 8(g–i), where the boxes with different gray colors are equivalent, and the boxes with different gray colors only represent the results from different boxes. The colors represent the number of the weak classifiers that detect a CME in the image.

Fig. 9 shows the accuracy rates of the weak classifiers $C_1, C_2, \ldots, C_5$ under slice sizes of 25 × 20, 30 × 15, 20 × 20, 15 × 20, 10 × 20 pixels, respectively and the final classifier after running the program three times. From the figure, we can see that the accuracy rate of the final strong classifier is not lower than an individual weak classifier.

5.2. Noise impact analysis

Several weak classifiers are used to construct the final classifier using a weighted summation. As a result, our method is insensitive to noise, such as streamers, bright lines, and points. Streamers can be eliminated when we compute the running difference image as they are steady bright structures in the coronagraph images. Moreover, images with noise such as bright lines and points can be detected correctly owing to the collective wisdom of the weak classifiers. But our detection algorithm is limited to images that do not contain artifacts which sometimes appear during data acquisition or as a result of data reduction.

5.3. Comparisons for CME detections

The running time of the program to process 6800 CMEs images from 1 month of observations is about 1 h on a HP personal computer with a CPU speed of 3.2 GHz and a memory of 4Gb. The running time of the experiment shows that our algorithm can meet the needs of real-time detection. The detection results are compared with two existing catalogs, namely LASCO and CACTUS (http://sidc.oma.be/cactus/catalog.php). We used data from May 1 to May 31, 2007. The LASCO catalog lists 137 CMEs while CACTUS finds 30. Nearly 68 CMEs are found by using AdaBoost. There is only one case included in CACTUS which occurred on May 1, 2007, 02:06 UT which was not detected by our method and also excluded in the LASCO catalog. Some CMEs omitted in CACTUS are detected by AdaBoost, e.g. A CME at 12:30 UT on May 1, 2007. For more detailed information, we refer the reader to http://cmedetection.blog.com/.

6. Conclusions and prospects

In this paper, an automatic algorithm to detect CMEs is presented. Experimental results show that the method using AdaBoost on grayscale statistic features is effective in detecting CMEs from running difference images. Our existing method can be further improved by adopting appropriate slice ways such as sectors. We will extend our work in the near future to extract CMEs parameters such as onset time, angular width, velocity, and acceleration, which are important for space weather studies.

Acknowledgment

The authors are highly appreciative of the referee's enormous time and effort spent in reading and evaluating our paper, and thank the referee for the valuable advice, comments, and suggestions which helped to improve this paper significantly.

References

Berghmans, D., Foing, B.H., Fleck, B., 2002. Automated detection of CMEs in LASCO data. In: Berghmans, D. (Ed.), Proceedings of the SOHO-11 Symposium on From Solar Min to Max: Half a Solar Cycle with SOHO. ESA Publications Division C/O ESTEC AG Noordwijk, pp. 437–440.

Byrne, J.P., Morgan, H., Habbal, S.R., Gallagher, P.T., 2012. Automatic detection and tracking of coronal mass ejections. II. Multiscale filtering of coronagraph images. Astrophys. J. 752 (2), 145.

DeForest, C.E., Howard, T.A., McComas, D.J., 2013. Tracking coronal features from the low corona to earth: a quantitative analysis of the 2008 December 12 coronal mass ejection. Astrophys. J. 769 (1), 43. doi:10.1088/0004-637x/769/1/43.

Gallagher, P.T., Young, C.A., Byrne, J.P., McAteer R. T., J., 2011. Coronal mass ejection detection using wavelets, curvelets and ridgelets: applications for space weather monitoring. Adv. Space Res. 47, 2118–2126. http://dx.doi.org/10.1016/j.asr.2010.03.028.

Goldberg, D.E., 1989. Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Goussies, N.A., Mejail, M.E., Jacobo, J., Stenborg, G., 2011. Detection and tracking of coronal mass ejections based on supervised segmentation and level set. Pattern Recognit. Lett. 31 (6), 496–501. doi:10.1016/j.patrec.2009.07.011.

Goussies, N.A., Stenborg, G., Vourlidas, A., Howard, R., 2010. Tracking of coronal white-light events by texture. Sol. Phys. 262 (2), 481–494. doi:10.1007/s11207-009-9495-6.

Howard, R.A., Sheeley, N.R., Koomen, M.J., Michels, D.J., 1985. Coronal mass ejections: 1979–1981. J. Geophys. Res. 90, 8173–8191. doi:10.1029/JA090iA09p08173.

Hundhausen, A.J., 1993. Sizes and locations of coronal mass ejections: SMM observations from 1980 and 1984-1989. J. Geophys. Res. 98, 13177–13200. doi:10.1029/93JA00157.

Morgan, H., Byrne, J.P., Habbal, S.R., 2012. Automatically detecting and tracking coronal mass ejections. I. Separation of dynamic and quiescent components in coronagraph images. Astrophys. J. 752, 144.

Olmedo, O., Zhang, J., Wechsler, H., Poland, A., Borne, K., 2008. Automatic detection and tracking of coronal mass ejections in coronagraph time series. Sol. Phys. 248 (2), 485–499. doi:10.1007/s11207-007-9104-5.

Qu, M., Frank, Y.S., Jing, J., Wang, H., 2006. Automatic detection and classification of coronal mass ejections. Sol. Phys. 237 (2), 419–431. doi:10.1007/s11207-006-0114-5.

Robbrecht, E., Berghmans, D., 2004. Automated recognition of coronal mass ejections (CMEs) in near-real-time data. Astron. Astrophys. 425 (3), 1097–1106. doi:10.1051/0004-6361:20041302.

Robbrecht, E., Berghmans, D., 2006. A broad perspective on automated CME tracking: Towards higher level space weather forecasting. Geophysical Monograph Series, vol. 165. American Geophysical Union, pp. 33–41. doi:10.1029/165gm06.

Schapire, E.R., Freund, Y., Bartlett, P., 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. Ann. Stat. 26 (5), 1651–1686. doi:10.1214/aos/1024691352.