Knowledge-Based Systems 70 (2014) 345–360

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Solar flare detection system based on tolerance near sets in a GPU–CUDA framework ☆

G. Poli [a], E. Llapa [a], J.R. Cecatto [b], J.H. Saito [a,c], J.F. Peters [d,*], S. Ramanna [e], M.C. Nicoletti [a,c]

[a] Computer Science Dept., Federal Univ. of S. Carlos, S. Carlos, SP 13565-905, Brazil
[b] Astrophysics Division, Nat. Inst. of Space Res., S.J. dos Campos, SP 12227-010, Brazil
[c] Faculty of Campo Limpo Paulista, C.L. Paulista, SP 13231-230, Brazil
[d] Elec. & Computer Eng. Dept., University of Manitoba, Winnipeg, MB R3T 5V6, Canada
[e] Applied Computer Science, University of Winnipeg, Winnipeg R3B 2E9, Canada

ARTICLE INFO

Article history:
Received 10 November 2013
Received in revised form 3 July 2014
Accepted 21 July 2014
Available online 31 July 2014

Keywords:
GPU–CUDA
Tolerance near sets
Solar flare detection
Tolerance class
Pattern recognition

ABSTRACT

This article presents a unique application of tolerance near sets (TNS) for detecting solar flare events in solar images acquired using radio astronomy techniques. In radio astronomy (RA) applications, the interferometric array processing of data streams presents algorithmic and response time challenges as well as a high volume of data. The radio interferometer is an RA instrument composed of an array of antennas. Radio signals emitted by a celestial object are captured by the antennas and are subsequently processed in such a way that each pair of antennas produces correlated data. The overall correlated data is then accumulated and, after an integration period, the spectral image of the observed object is obtained. The process of deconvolution of the spectral image produces the desired spatial image of the celestial object. The proposed solar flare detection system is embedded in a computational platform framework suitable for dealing with huge volumes of data, based on a cluster of CPU–GPU pairs. The experimental results presented in the paper include comparison of the TNS-based algorithm (implemented as the SOL-FLARE system) with the K-means algorithm using significant samples of test images to validate the detection system. The performances of both systems are comparatively analyzed using Receiver Operating Characteristic (ROC) curves. The images used in the experiments were selected from a data repository produced by the Nobeyama Radioheliograph, in Japan, during the years 2004 up to 2013. The main contribution of the article is a novel approach to solar flare detection in a GPU–CUDA framework.

1. Introduction

This article describes a novel approach to detection of solar flare events in images acquired with radio astronomy techniques, using a tolerance form of near sets. The article also presents and discusses the implementation of the solar flare detection system SOL-FLARE [6], which employs (i) tolerance near sets as a formal model [25], (ii) a GPU–CUDA parallel processing environment [19], and is related to recent work on analyzing celestial spectral data, as described in [39]. SOL-FLARE has two phases. In the first phase, tolerance classes are induced from a set of typical flare images, and the representative (a kind of prototype) of each tolerance class is computed. In the second phase, a flare is detected when a region of a solar image can be assigned to, at least, one of the tolerance classes identified in the first phase. What follows is a brief introduction to tolerance near sets and background information specific to radio interferometer array processing and a work flow diagram that gives the details of the data acquisition and detection processes.

1.1. Tolerance near sets (TNS)

TNSs [11,24–28] provide an intuitive as well as mathematical basis in defining what it means for pairs of objects to be similar. Relations with the same formal properties as similarity relations of sensations considered by Poincaré [29] are nowadays, after Zeeman [37], called *tolerance relations*. A *tolerance relation* τ *on a set* O is a relation $\tau \subseteq O \times O$ that is reflexive and symmetric. Tolerance

☆ This research has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants 185986 and 194376.

* Corresponding author. Address: Computational Intelligence Laboratory, Department of Electrical & Computer Engineering, University of Manitoba, E1-526, 75A Chancellor's Circle, Winnipeg, MB R3T 5V6, Canada.
E-mail address: james.peters3@ad.umanitoba.ca (J.F. Peters).

http://dx.doi.org/10.1016/j.knosys.2014.07.012
0950-7051/

classes are sets where all the pairs of objects within each set are related via the tolerance relation and the set is maximal with respect to inclusion. Recent work by the authors include experiments with a form of tolerance nearness measure (tNM) in content-based image retrieval (CBIR) [1,10,12–14]. The focus of these works has been an effort to improve the performance of the tNM measure for a general CBIR task as well as comparison with well-known distance measures such as Earth Movers Distance and Integrated Region Matching. One key issue was to find an efficient and effective way to compute tolerance classes, which lead to experiments using a parallel computing platform.

In this article a TNS-based algorithm is used for the specialized task of detecting solar flares. The TNS algorithm can be considered as a non-partitional learning method where closeness is defined in terms of texture-based probe functions used to extract pixel information from solar images. The idea is to find sub-images whose set of probe function values resembles those of a representative of a flare tolerance class, within a given discrimination threshold. The term *flare tolerance class* is used in a machine learning context whereas tolerance classes are defined in the mathematical context of relations. A formal model for tolerance near sets used in this work is given in Section 3.

1.2. Radio interferometric arrays

The challenges associated with scientific computing, besides its inherent complexity, are mainly related to the specific and demanding needs of many of its applications, such as those associated with response time, complexity of the employed algorithms and the substantial volume of data involved. In radio astronomy (RA) applications, the interferometric array processing of data stream is a typical representative of such needs. The radio interferometer is an RA instrument composed of an array of antennas. Radio signals emitted by a celestial object are captured by the antennas and are subsequently processed, in such a way that each pair of antennas produces correlated data. The overall correlated data is then accumulated and, after an integration period, the spectral image of the observed object is obtained. The process of deconvolution of the spectral image produces the desired spatial image of the celestial object.

There are many advantages in making celestial observations based on radio wave frequencies. The first is that it is possible to detect radio waves using ground-based devices instead of using spacecrafts. The second is that some objects and phenomena are invisible or hard to detect in other wavelengths, and can only be detected or can be detected with higher sensitivity through their radio wavelengths. Third, the detected signals provide quantitative physical information about the conditions of the observed astronomical object. Due to the limited spatial resolution of single parabolic antenna radio telescopes, sophisticated techniques have been developed to emulate a single antenna telescope with a multiple antenna arrangement called a radio interferometer array. In such an array, the spatial resolution is determined not by the size of the individual directional antenna, but by the maximum separation between antennas, referred to as maximum baseline.

Interferometer arrays currently under construction, such as MeerKAT (South Africa) with 80 antennas, will generate a daily volume of 3 terabytes (TB). The Square Kilometre Array (SKA) with 2400 antennas is expected to generate a daily volume of 80 TB in 2021. The Nobeyama Radioheliograph (NoRH), part of the Nobeyama Radio Observatory in Japan, is dedicated to collecting data from the Sun and consists of 84 parabolic antennas with 80 cm diameter each, placed on 490 m long lines in the East–West direction and 220 m long in the North–South direction.

1.3. Overview of the SOL-FLARE detection system

In a radio interferometer array, after being captured by each individual antenna of the array, the radio signals are processed so that each pair of antennas produces correlated data, for each possible number of such pairs. Considering an array of N antennas, there are $(N(N-1))/2$ pairs (or baselines). The correlated data is then accumulated during what is called an integration period; at the end of this period, the overall result of all antenna pairs taking into account the positions of each antenna pair, forms the spectral image of the observed celestial object. After that, the spectral image undergoes a deconvolution process which, at the end, produces the desired spatial image of the object.

A high quality image requires a large number of different baselines between antennas. Although many radio interferometers

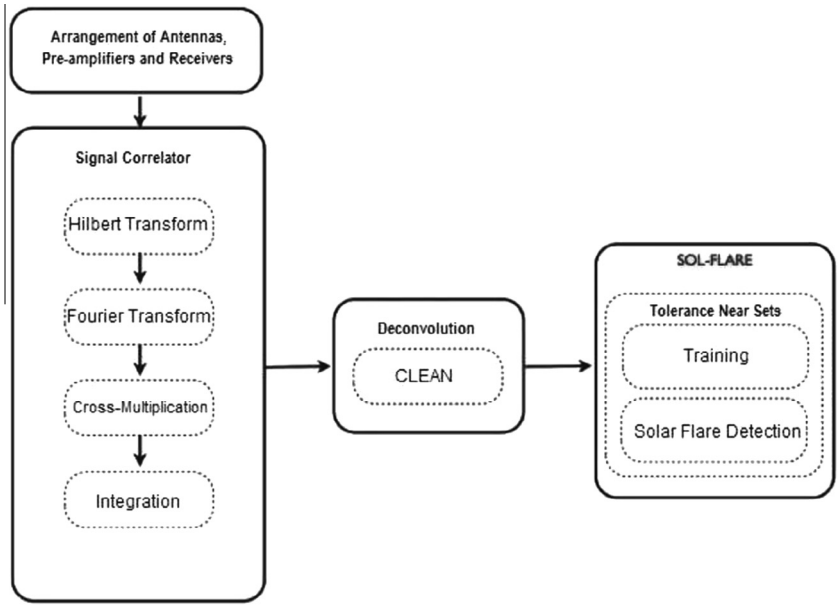


Fig. 1. A workflow diagram showing radio-interferometer data stream for obtaining solar images and the automatic detection of solar flares.

have been constructed to process signal correlations in hardware (subsystem known as signal correlator), the cost of such equipment is very high and is also proportional to the number of antennas of the array. Besides, a signal correlator is specialized hardware and therefore not suitable to be used in other processing tasks such as deconvolution and object recognition, which are also needed in a full radio interferometer system. The first radio interferometer signal correlator implemented in software was part of the DiFX Software Project [3], developed to run on a computer cluster and its most recent version uses an IBM Cell architecture.

The event known as a solar flare occurs around sunspots and can be detected by a sudden brightening over the Sun's surface interpreted as a large energy release (see [16,36] for details). Sunspots can be considered temporary events on the Sun that appear visibly as dark spots caused by huge magnetic fields and usually appear in pairs, one spot leading and the other following. The two simultaneously occurring spots have opposite magnetic polarities, joined by vertical loops, and due to the differential rotation of the Sun at different latitudes, the magnetic fields are twisted [17].

The workflow diagram in Fig. 1 describes the complete data flow from captured raw data up to flare detection. It starts with radio signals being captured by the antennas, followed by the process of correlation of the signal and its subsequent deconvolution; finally, it follows the process of flare detection, implemented as the SOL-FLARE system. The SOL-FLARE system includes the TNS-based algorithm embedded in a GPU–CUDA Interferometric Data Processing framework, suitable for dealing with huge volumes of data via a cluster of CPU–GPU pairs.

This article has the following organization. Synthesis imaging in radio astronomy, the Högbom's CLEAN method, and the solar image processing using a GPU–CUDA framework are briefly introduced in Section 2. A formal model for tolerance near sets used in this work is given in Section 3. The two main phases in the SOL-FLARE system for detecting solar flares in solar images based on TNS are introduced in Section 4. The SOL-FLARE system is exemplified in a case study, and its parallel processing details are presented in Section 5. Section 6 describes the results of flare detection considering a statistically significant number of images where both methods, TNS-based implemented as the SOL-FLARE system and K-means are compared. The Receiver Operating Characteristic (ROC) curve is used to analyze their performance. The goals and contributions, as well as a few research lines to be pursued, are briefly presented in Section 7.

2. Synthesis imaging in radio astronomy

This section introduces radio signal acquisition with radio telescopes, deconvolution and image processing steps and the GPU–CUDA framework.

2.1. Radio signal acquisition and correlation

Radio telescopes are used to study naturally occurring radio frequency emissions from stars, galaxies, quasars, and other astronomical objects between wavelengths of about 10 m (30 megahertz (MHz)) and 1 mm (300 gigahertz (GHz)). One important application of radio telescopes is to monitor solar activities.

The angular resolution i.e., the characteristic that allows a radio telescope to detect details in the sky, depends on the wavelength of observations divided by the diameter of the instrument. Yet, even the largest antennas, when used at their shortest operating wavelength, have an angular resolution only a little better than one arc minute, which is comparable to that of the unaided human eye at optical wavelengths. Because radio telescopes operate at much longer wavelengths than do optical telescopes, radio telescopes must be much larger than optical telescopes to achieve the same angular resolution. Radio telescopes vary widely, but they all have two basic components: (1) a large radio antenna and (2) a sensitive radiometer or radio receiver. The sensitivity of a radio telescope, i.e., the ability to measure weak sources of radio emission, depends on the area and the efficiency of the antenna as well as on the sensitivity of the radio receiver used to amplify and detect the signals. For broadband continuum emission, the sensitivity also depends on the bandwidth of the receiver. Due to the fact that cosmic radio sources are extremely weak, radio telescopes are usually very large and only the most sensitive radio receivers are used.

Since its early days, radio astronomy has been coping with technical limitations and costs; it is not possible to reduce the wavelength within the frequency band as much as one would like to. A solution could be to increase resolution power by increasing the diameter of the antenna. However, due to both mechanical and cost limitations, there exists a limit for increasing the diameter of a parabolic antenna. It is possible, however, to emulate a virtual giant antenna, by using the principle of aperture synthesis, through the use of many relatively small antennas, in a convenient arrangement. All the operational antennas in the arrangement are simultaneously positioned pointing at the same radio source, for capturing and measuring its radio emission (Fig. 2 exemplifies this situation using two antennas). The result of these measurements is the Fourier Transform of the brightness distribution of the source observed by the instrument with a finite set of components of spatial frequencies [35].

The most basic case of an interferometer array is a composition of only two antennas pointing at a distant radio source, as in Fig. 2. The direction of the radio wave is indicated by the vector s and the distance between the antennas, known as the baseline, represented by the vector b. Let τ_g denote a radio wave geometric delay and let c be the speed of light in a vacuum. In general, the radio wave from the source reaches Earth in the form of a plane wave and reaches the antenna at different times, producing a geometric delay (depending on the wave incidence angle), given by $\tau_g = s \cdot b/c$. In Fig. 2, observe that the signal received by each radio interferometer antenna passes through a pre-amplifier (to amplify the received signal), as well as by a receiver system (which has,

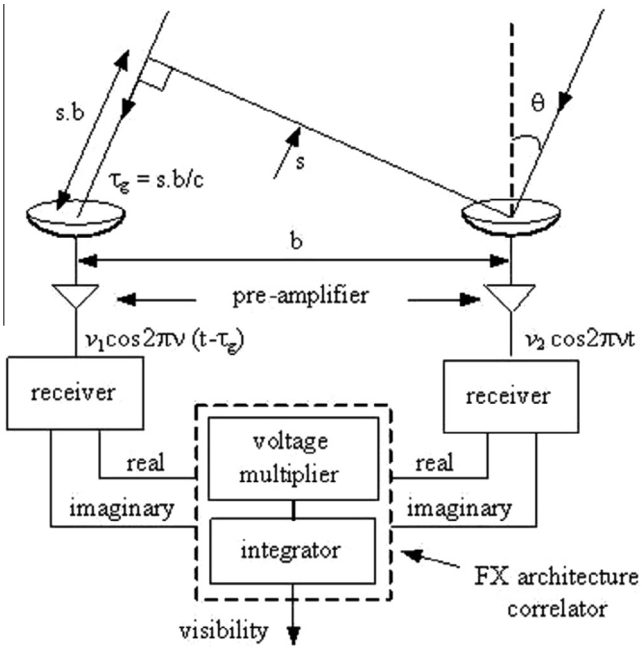


Fig. 2. Radio interferometer.

among other components, an embedded filter to select the frequency of interest within the wave bandwidth).

To obtain visibility information about the radio source, the voltage signals are sampled, quantized and then an appropriate geometric delay is introduced in one of the arriving signals, to promote synchronization. Finally the signals are multiplied, integrated by a suitable time and normalized to provide the output signal. This whole process, which involves transforming the received raw data into visibility information, occurs within a device known as correlator. The correlator receives as input the voltage signals $V_1(t)$ and $V_2(t)$ produced by the antennas, via the receiver, given by Eq. (1), respectively, where v is the frequency of the signal, t is time and v_1 and v_2 are amplitudes of the voltage signal received at each one of the antennas,

$$V_1(t) = v_1 \cos 2\pi v(t-\tau_g), \quad V_2(t) = v_2 \cos 2\pi v t. \tag{1}$$

The product of $V_1(t)$ by the complex conjugate of $V_2(t)$ is proportional to the received signal power, and is the output of the correlator, given by Eq. (2),

$$r(\tau_g) = v_1 v_2 \cos 2\pi v \tau_g. \tag{2}$$

It is interesting to note that without the introduction of the geometric delay before a correlation process, an inconvenient side effect, known as interference fringe, is produced, due to rotation of the Earth. The output of the correlator can also be approached in terms of the brightness of the radio source, $I_v(s)$ in direction s having frequency v. Let $A(s)$ represent the effective collecting area of each antenna. Having two antennas pointing in the same direction s, the power of the received signal in the bandwidth $\Delta\tau$, within a solid angle $d\Omega$ is given by $I_v(s)A(s)\Delta v d\Omega$. Thus, the output of the correlator can also be obtained using Eq. (3),

$$dr = I_v(s)A(s)\Delta v d\Omega \cos 2\pi v \tau_g. \tag{3}$$

To obtain information about the amplitude as well as the phase of the measured signal, a more elaborate correlator, known as complex correlator, which processes an incoming signal as two signals, one of them known as imaginary component, is required. The real component of the signal does not change but the imaginary component undergoes a Hilbert Transform, which causes a shift in its phase of exactly 90 degrees; the process is detailed next. In a complex correlator the two input voltage signals produced by the antennas, i.e., $V_1(t)$ and $V_2(t)$, are given by Eq. (4), where v is the frequency of the signals, t is time, A_1 and A_2 are the amplitudes, and α_1 and α_2 are the phases of the two signals, respectively,

$$V_1(t) = A_1 e^{j(2\pi v t+\alpha_1)}, \qquad V_2(t) = A_2 e^{j(2\pi v t+\alpha_2)}. \tag{4}$$

To measure the correlation of the two signals, the correlator multiplies the two inputs (i.e., $V_1(t)$ is multiplied by the complex conjugate of $V_2(t)$) and takes the time average of the product, as shown in Eq. (5),

$$\begin{aligned} \langle V_1(t)V_2^*(t)\rangle &= A_1 e^{j(2\pi v t+\alpha_1)} \times A_2 e^{-j(2\pi v t+\alpha_2)} \\ &= A_1A_2[\cos(\alpha_1-\alpha_2)+j\sin(\alpha_1-\alpha_2)] \\ &= A_1A_2[\cos(\alpha_1-\alpha_2)+j\cos(\alpha_1+\pi/2-\alpha_2)]. \end{aligned} \tag{5}$$

From (5), the complex correlation results in a complex number, composed of a real part, given by $A_1A_2\cos(\alpha_1-\alpha_2)$ and an imaginary part, given by $A_1A_2 j\cos(\alpha_1+\pi/2-\alpha_2)$. To measure the imaginary part, another correlator is required, to which is fed the same signals but with an extra $\frac{\pi}{2}$ phase shift in one of them. A complex correlator performs two operations: a Fourier Transform (F) and a cross-multiplication of the signals (X). The order of the two operations is what determines the architecture of the correlator, giving rise to two types of correlators: FX and XF [34]. The FX correlator performs the F operation first and then the X operation. The imaging system of an FX correlator radio interferometer used to obtain a solar image is introduced in Section 2.2.

2.2. Deconvolution and image processing

In a radio interferometer the result of the correlation of signals is known as sampled visibilities. In what follows, several existing relations among various elements involved in a radio interferometer imaging process are considered.

The upper row of Fig. 3 (extracted from [5]) shows three figures which correspond, from left to right, to the sky plane (l,m) representations of: 3(a) the sky object map $(I(l,m))$, 3(b) sampling beam $(B(l,m))$, and 3(c) dirty map $(I(l,m)*B(l,m))$, where $*$ is a convolution operation. The lower row, from left to right, is the corresponding u, v plane images (spectral images) of: 3(d) visibility $(V(u,v))$, 3(e) sampling function $(S(u,v))$ and 3(f) sampled visibility $((V(u,v)\times S(u,v))$, which represents the actual measurements made by a radio array. Fig. 3(c) shows the corresponding

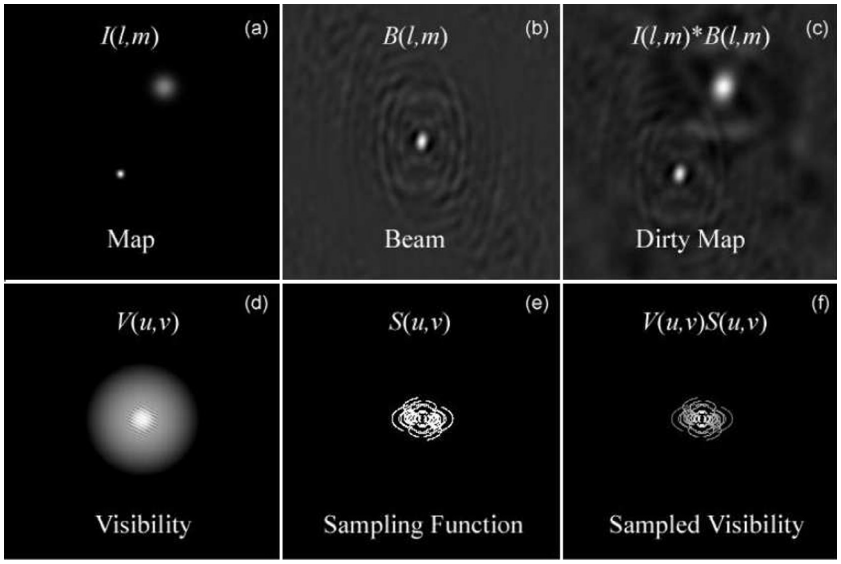


Fig. 3. Dirty map.

spatial image associated with the celestial object. Note that the dirty map uses only the sampled visibilities, so it does not contain full information about the sky brightness distribution. Standard image reconstruction techniques should be used to recover the missing spacings, in an attempt to recover the true map (Fig. 3(a)).

The dirty map (Fig. 3(c)), representing the observed celestial object shown in Fig. 3(a), should be processed aiming at a better quality image. The dirty map is the result of applying the FFT (Fast Fourier Transform) on the sampled visibility; it has unavoidable noise components, since the actual visibility function measured by an instrument is $V'(u,v) = V(u,v) + N(u,v)$, where $N(u,v)$ is the noise component. Also, to be inverted with FFT, the gridding of the $u,\ v$ sampled visibility data in two-dimensional array positions is required, so that the resampled visibility array has some cells empty and other cells that do not correspond to a $u,\ v$ point measured with the array. The goal of image restoration techniques is to find an algorithm that allows reasonable values for the correction of all the above mentioned occurrences. Several methods are available and the two most popular are the CLEAN algorithm, proposed in [15] and the MEM (Maximum Entropy Method) [2,32,33]. A high-level description of the CLEAN algorithm, used in the SOLFLARE system, is given in Algorithm 1.

The CLEAN algorithm is based on the a priori assumption that the sky intensity distribution is a collection of point sources, i.e., points that radiate the same intensity of radiation in all directions. The algorithm employs a simple iterative approach to find the positions and intensities of the point sources and applies a correction operation. The final deconvolved image is the collection of the point sources components convolved with a beam, which is the point spread observation function (Fig. 3(b)).

Algorithm 1. Högbom CLEAN method [15].

Input : *initial dirty image, initial* DM (dirty map), *initial* DB (dirty beam), I_o, TH (threshold)
Output: *final* DM (cleaned image)
repeat
  Compute DM, DB *using Fourier inversion methods*;
  Find strength, position of peak I_o *in the dirty image*;
  Subtract over whole map a DB pattern centered at I_o;
  $DM \leftarrow$ remaining map from previous step;
  Compute new I_o;
until $I_o < TH$;
Convolve the resulting point source model with an ideal beam (usually an elliptical Gaussian);
Add the residuals of the dirty image to obtain the final DM;

The workflow diagram shown in Fig. 1 starts with the radio signals from the Sun being captured by the antennas. The FX correlation of the signals is detailed in four sequential steps (Hilbert Transform, Fourier Transform, cross-multiplication, and integration). The first three steps are repeated several times and the results are accumulated during a period of integration of approximately 100 ms, for a real time imaging radio interferometer for Sun monitoring, also known as radioheliograph. After the period of integration, the resulting data can be processed to obtain the spatial dirty image, so a deconvolution process takes place. At the end, the solar image is obtained and subsequently used for the flare detection process, implemented as a tolerance near set-based

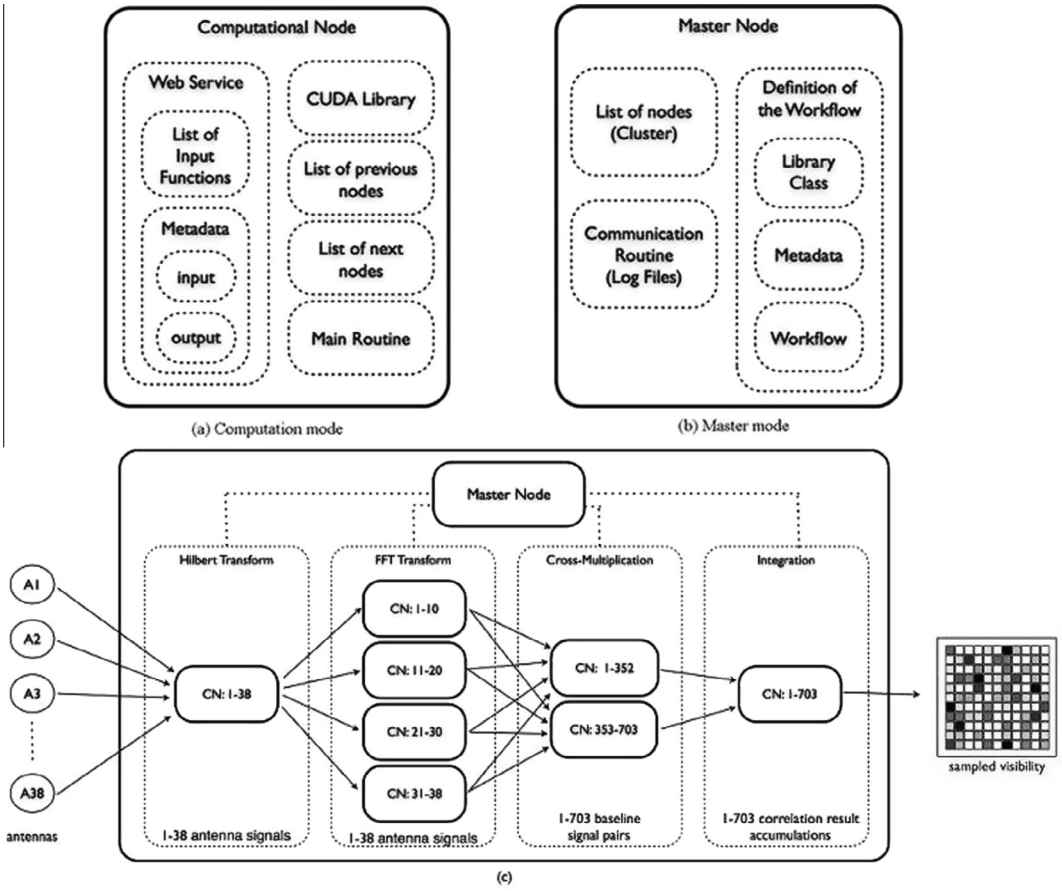


Fig. 4. Proposed GPU–CUDA framework diagrams.

computational system SOL-FLARE, in a CUDA parallel processing environment (see Section 2.3).

2.3. The GPU–CUDA Interferometric Data Processing

The proposed GPU–CUDA framework, suitable for processing huge volumes of data, is composed of a cluster of CPU–GPU pairs in a CUDA environment. A Graphic Processing Unit (GPU) can be considered a parallel multi-thread processor, coupled to a host processor CPU, where the thread processors can be scheduled by a control unit denoted as scoreboard [9]. Currently CPU–GPU pairs, besides graphic processing are also widely used in general parallel processing applications. The Compute Unified Device Architecture (CUDA) is a parallel processing environment created for GPU general purpose applications [19].

The proposed GPU–CUDA framework was developed in collaboration with the Astrophysics Division of the National Institute of Space Research in Brazil as part of the Brazilian Decimetric Array (BDA) radio interferometer project (see Fig. 4) [31]. It is based on an object-oriented model, where the implemented classes are distributed throughout the cluster; each computational node is composed of a CPU–GPU pair, and the communication interface used between them is based on the Web Services technologies (WS) [4] (Fig. 4). Each object in a computational node represents a simple service, i.e., a function responsible for the execution of a specific task, for example a Fourier Transform, a Hilbert Transform, a cross-multiplication, among several others. The use of an object-based model for implementing the signal correlator processing in a distributed environment was described in [30]. The proposed GPU–CUDA framework allows two kinds of classes: (1) infrastructure class and (2) operational class. Infrastructure classes are responsible for the usual operations on all computational nodes, such as communication between nodes, exchange of information, supplying an API for an operational class to use the GPU as a parallel device, and many others. Operational classes are related with the problem domain, i.e., an operational class performs a specific function related to a particular application.

Since the proposed GPU–CUDA framework uses the WS technology, each operational class is treated as an available service in the network. The number of computational nodes made available is related to the data volume and the required response time. Another important characteristic is the scalability of the services it provides. In the continuous growing data volume, it is possible to add new nodes of desired/required services, to meet the application requirements.

The workflow is defined by a special node called the master node (Fig. 4). The master node contains a set of configuration files defining: the number of nodes, the services available at each node, the description of the input dataset and the sequence of service calls. The master node, the set of computational nodes, and the set of configuration files that define how the job should be executed, can be processed in pipeline.

Fig. 4 presents a general view of the workflow of the signal correlator, identifying its four functionalities, namely: (1) Hilbert Transform; (2) Fourier Transform (FFT); (3) cross-multiplication and (4) integration of the results of each processing cycle. A set of computational nodes is associated with each functionality. Each set of such nodes processes the input data and forwards the results to the next set of nodes. The arrangement presented is composed of eight computational nodes (CN), one for Hilbert Transform, four for Fourier Transform, two for cross-multiplication, and one for integration. This arrangement is sufficient to process the data generated by an interferometer composed of thirty-eight antennas, inducing 703 baselines, with sampling signals at 5 MHz, at an integration time of 105.8 ms [30]. The notation CN:N1–N2 refers to a computational node associated with data in the interval [N1,N2].

3. Near sets

The SOL-FLARE system employs a set-theoretic approach based on near set theory embedded in a GPU–CUDA framework. Near set theory was influenced by the rough set theory [8,20]. The basic structure which underlies the near set theory is a perceptual system which consists of perceptual objects (i.e., objects that have their origin in the physical world) [22,23]. Near set theory is characterized by a perceptual system, whose objects are associated by relations such as indiscernibility, weak indiscernibility, and tolerance, taking into account descriptions using a tuple of probe functions. In this section, a brief introduction to the main definitions underlying near sets is given.

3.1. Preliminaries

Definition 1 (*Perceptual System [22,23]*). A perceptual system is a pair $\langle O, F\rangle$, where O is a nonempty set of perceptual objects and F is a countable set of real-valued probe functions $\phi_i : O \to \mathbb{R}$.

An object description in a perceptual system is a tuple of probe function values $\Phi(x)$ associated with an object $x \in X$, where $X \subseteq O$, as defined by Eq. (6), where $\phi_i : O \to \mathbb{R}$ is a probe function of a single feature,

$$\Phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x)). \tag{6}$$

The probe functions give rise to a number of perceptual relations between objects of a perceptual system. This approach is useful when decisions on nearness are made in the context of a perceptual system i.e., a system consisting of objects and our perceptions of what constitutes features that best describe these objects.

Definition 2 (*Perceptual Indiscernibility Relation [28]*). Let $\langle O, F\rangle$ be a perceptual system and let $B \subseteq F$. A perceptual indiscernibility relation is given by Eq. (7),

$$\sim_B = \{(x, y) \in O \times O : \text{for all } \phi_i \in B,\ \phi_i(x) = \phi_i(y)\}. \tag{7}$$

Definition 2 is a refinement of the original idea of an indiscernibility relation between objects to what is known as perceptual indiscernibility that is more in keeping with the perception of objects in the physical world.

Definition 3 (*Weak Indiscernibility Relation [28]*). Let $\langle O, F\rangle$ be a perceptual system and let $B \subseteq F$. A weak indiscernibility relation is given by Eq. (8),

$$\cong_B = \{(x, y) \in O \times O : \text{for some } \phi_i \in B,\ \phi_i(x) = \phi_i(y)\}. \tag{8}$$

Definition 3 bears a resemblance to the human visual recognition system which compares similarity based on some features, but not all of them.

3.2. Tolerance near sets

The term *tolerance space* was coined by Zeeman in 1961 in modelling visual perception with tolerances [38]. A tolerance space $(X, \simeq)$ consists of a set X endowed with a binary relation $\simeq$ (i.e., a subset $\simeq \subset X \times X$) that is reflexive (for all $x \in X$, $x \simeq x$) and symmetric (for all $x, y \in X$, $x \simeq y$ and $y \simeq x$) but transitivity of $\simeq$ is not required. The notion of tolerance is directly related to the idea of closeness between objects, such as image segments that

resemble each other with a tolerable level of difference [27]. Tolerance relations can be considered generalizations of equivalence relations.

Definition 4 (*Perceptual Tolerance Relation [24,25]*). Let $\langle O, F\rangle$ be a perceptual system and let $B \subseteq F$. A perceptual tolerance relation is given by Eq. (9), where $\|\cdot\|_2$ denotes the L^2 norm of a vector,

$$\cong_{B,\epsilon} = \{(x,y) \in O \times O : \|\phi(x) - \phi(y)\|_2 \leqslant \varepsilon\}. \tag{9}$$

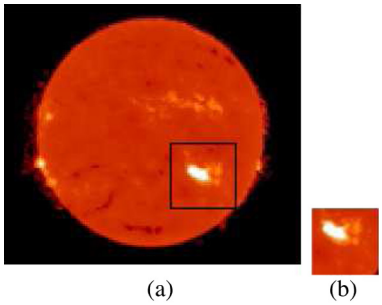
Mathematically, the set of images I described by set of probe functions B, considering a tolerance level ε, is divided into tolerance classes, where a tolerance class C can be formally defined as a set of images $x,\ y \in$ I such that, $x \cong_{B,\epsilon} y$, and C is maximal with respect to inclusion.

In the flare detection system, Definition 4 implies that the idea is to find images that resemble a representative of a flare tolerance class, within a given discrimination threshold, based on their feature values.

4. Detecting solar flares based on TNS

In what follows an automatic solar flare detection process based on image processing is proposed with TNS as its formal model. The basic structure for a TNS includes a non-empty set of objects and a countable set of probe functions. Descriptions of each object (via probe functions) consist of several measurement values obtained using image processing techniques. As previously discussed, a radio interferometer can obtain a sequence of solar images that present active regions of solar flares. Fig. 5(a), for instance, shows a solar radio interferometer image, seen at 17 GHz by the Nobeyama Radioheliograph [18], where the flare region is shown inside a square box. Fig. 5(b) shows the corresponding flare sub-region. Recall that a radio interferometer can obtain a sequence of solar images that present active regions of solar flares.

In this paper a flare event refers to the temporary event that can be characterized as a sudden brightening over the Sun's surface. In this article, solar flare tolerance classes differ from each other based on the *texture* of their images due to the different positions from which they are extracted from the full Sun disk, as well as the different energy intensities of the event. The SOL-FLARE detection system first performs a TNS-based tolerance classification of typical flare image samples to compute the representatives of each tolerance class to be used in the detection phase. This procedure helps reducing the SOL-FLARE system computational cost in the GPU–CUDA framework. By using representatives, the number of calculations can be reduced. Instead of performing pairwise distance measurements taking into account all available images, only the representatives are considered.



(a) (b)

Fig. 5. Sun: 17 GHz.

4.1. Choosing the probe functions

The probe functions used for describing images are based on statistical measurements generated by Gray Level Co-occurrence Matrices (GLCMs). The gray scale value is obtained from the RGB image by calculating the value $0.3R + 0.59G + 0.11B$ associated to each pixel, where R, G and B are the red, green, and blue intensities, respectively. GLCMs characterize the texture of an image by calculating how often pairs of pixels with respect to specified angle and distance, as well as specific pixel values occur in an image. Such an approach has been used in experiments described in [11]. Let I represent an image and let $L_x = \{0,\ldots,N_x\},\ L_y = \{1,\ldots,N_y\}$ be the horizontal and vertical domains of I, quantized to N_g pixel levels.

In the $N_g \times N_g$ dimensional GLCM matrix, rows refer to the reference pixel values varying from 0 to $N_g - 1$ (up-down), and the columns refer to the neighbor pixel value varying from 0 to $N_g - 1$ of image I. An element $p(i,j)$ refers to a texture element, defined by (i,j) of reference pixel gray level i and neighbor pixel gray level j. When calculating the GLCM matrix element $p(i,j)$, the entire image I is scanned for the corresponding texture (i,j). The resulting value of $p(i,j)$ is the number of occurrences of the searched texture element.

The neighbor pixel may be in a specific angle of the considering texture, such as 0°, 45°, and 90°. The distance is 1 when the adjacent pixel is considered. A symmetrical GLCM is obtained when a pair of angles such 0° and 180° is considered. The counts are then normalized. Consider the simple case when $angle = 0^\circ$ and distance $d = 1$, then GLCM is defined by matrix P whose elements are given by Eq. (10),

$$p(i,j) = |\{((k,l),(k,l+1)) \in (L_x \times L_y)^2 : I(k,l) = i, I(k,l+1) = j\}|. \tag{10}$$

Table 1 shows the five probe functions energy, entropy, contrast, homogeneity, and correlation used in the experiments, where p_{ij} represents the element in position (i,j) of the normalized symmetrical GLCM and N is the number of gray levels in the image.

This study considers probe functions for the GLCM properties, namely, energy, entropy, contrast, homogeneity, and correlation (for details, see [7]).

4.2. SOL-FLARE phases

The Solar Flare detection system (SOL-FLARE) has two phases (Phase I and Phase II). In the first phase, given a tolerance level ε, tolerance classes are induced from a set of typical flare image patterns, and the representative of each tolerance class is computed. In the second phase, a flare is detected when the L^2 distance from the probe vector a new solar sub-image and at least one of the

Table 1
Probe functions.

Probe function	Obtained as	Equation
Energy	$\sum_{i,j=0}^{N-1} p_{ij}^2$	(11)
Entropy	$\sum_{i,j=0}^{N-1} p_{ij}(-ln(p_{ij}))$	(12)
Contrast	$\sum_{i,j=0}^{N-1} p_{ij}(i-j)^2$	(13)
Homogeneity	$\sum_{i,j=0}^{N-1} \frac{p_{ij}}{1+(i-j)^2}$	(14)
Correlation	$\sum_{i,j=0}^{N-1} p_{ij} \times \frac{(i-\mu_i)(j-\mu_j)}{\sqrt{\sigma_i^2\sigma_j^2}}$	(15)
	$\mu_i = \sum_{i,j=0}^{N-1} i p_{ij}$	
	$\mu_j = \sum_{i,j=0}^{N-1} j p_{ij}$,	
	$\sigma_i^2 = \sum_{i,j=0}^{N-1} p_{ij}(i-\mu)^2$	
	$\sigma_j^2 = \sum_{i,j=0}^{N-1} p_{ij}(j-\mu)^2$	

tolerance class representatives obtained in the first phase, is smaller than a chosen discrimination threshold η. The next algorithms use the notation: *procedure_name*($input_1, \ldots, input_n; output_1, \ldots, output_m$).

Algorithm 2. Phase I: tolerance class assignments.

Input : $\varepsilon > 0$, // Tolerance level
Φ, // Probe functions
K, // Number of probe functions
$SR = \{SR_1, \ldots, SR_M\}$, // 50× 50 RGB flare sub-images
M // Number of sub-images
Output: $(NC, \{R_1, \ldots, R_{NC}\})$
for $i \leftarrow 1$ **to** M **do**
  convert($SR_i; GSSR_i$);
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow i$ **to** K **do**
    computeProbe($GSSR_i, \phi_j$; ProbeValue$_{ij}$);
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow i+1$ **to** M **do**
    computeNormL$_2$($GSSR_i, GSSR_j, ProbeValue_{ij}$; NormL2$_{ij}$);
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow i+1$ **to** M **do**
    defineToleranceRelation(NormL2$_{ij}, \varepsilon$; SetOfPairs);
// Begin of defineClass(SetOfPairs; NC, H);
$H \leftarrow \emptyset$; // Set of tolerance classes
for $i \leftarrow 1$ **to** M **do**
  computeNeighbour(SetOfPairs, $i, SR; N_i$); // Compute the neighbourhood N_i of image SR_i
  $C_i \leftarrow N_i$; // Start the class C_i as the set containing all N_i
  for ***all*** $x, y \in N_i$ **do**
    if $x, y \notin SetOfPairs$ **then**
      $C_i \leftarrow C_i - \{y\}$; // Exclude y from class C_i
  $H \leftarrow H \cup \{C_i\}$;
  // C_i is one tolerance class induced by the tolerance relation
$NC \leftarrow |H|$; // Number of classes
// End of defineClass
defineClassRepresentative($NC, H; \{R_1, \ldots, R_{NC}\}$);

In Phase I, given a tolerance level ε and a set of typical solar sub-images containing flare events, Algorithm 2 returns the tolerance class(es) associated with each sub-image, taking into account the tolerance relation induced on the set of image pixels. The *convert* function converts M color sub-images SR_i to grayscale images $GSSR_i$. Then the *computeProbe* function uses the probe functions in Φ to extract the feature values from each of the sub-images $GSSR_i$. The L_2 norm distance between each pair of images is computed using *computeNormL*$_2$. Then, by using the defineToleranceRelation function, a set of sub-image pairs within the tolerance level ε is found.

To obtain a tolerance class corresponding to a flare sub-image SR_i, its neighborhood N_i is composed of all the sub-images within the tolerance level ε of image SR_i, including SR_i. If $N_i = \{SR_i\}$ then SR_i is a tolerance class with only one element. If a pair of neighbors in N_i does not satisfy the tolerance relation, the corresponding element is excluded from the tolerance class C_i.

The *defineClassRepresentative* function computes the representatives (prototype) of each of the *NC* tolerance classes. The tolerance class prototype is a vector whose values are computed as the mean of the probe function values of those belonging to that class. In Phase II of SOL-FLARE, after converting the color solar images to grayscale, the flare detection is performed using Algorithm 3.

Algorithm 3. Phase II: detection of solar flares.

Input : $\eta > 0$, // Discrimination threshold
Φ, // Probe functions
FSI, // Full solar image
$\{R_1, \ldots, R_{NC}\}$, // From Phase I
N // No. of sub-images from FSI defined by user
Output: $[flare_1, \ldots, flare_N]$
obtainSubimages($FSI, N; \{SI_1, \ldots, SI_N\}$);
for $i \leftarrow 1$ **to** N **do**
  convert($SI_i; GSSI_i$);
for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** K **do**
    computeProbe($GSSI_i, \phi_j$; ProbeValue$_{ij}$);
for $i \leftarrow 1$ **to** N **do**
  $flare_i \leftarrow false$;
  for $j \leftarrow 1$ **to** NC **do**
    computeNormL$_2$($R_j, ProbeValue_i; N_{ij}$);
  for $j \leftarrow 1$ **to** NC **do**
    if $N_{ij} < \eta$ **then**
      $flare_i \leftarrow true$;
  // the detection of $flare_i$ is returned

The algorithm starts with the *obtainSubimages* function, which divides the full solar image FSI into *N* (user defined) square sub-images. Next, the *convert* function converts the RGB sub-images into gray-scale representation. The *computeProbe* function, similar to the one used in Algorithm 2, is used to calculate the value of the probe functions for each of the N sub-images. Next, the flare detection process is conducted for each of the sub-images, using the *computeNormL*$_2$ function, which calculates the distance between the sub-images N_{ij} and the NC class representatives obtained in Algorithm 2. In a sub-image, if the computed distance is less than the specified discrimination threshold η, given by the user, with respect to at least one of the representatives, then a flare is detected.

5. SOL-FLARE system: an initial case study

This section illustrates the use of SOL-FLARE on a small sample of solar images extracted from the Nobeyama repository. More substantial experiments are reported in Section 6.

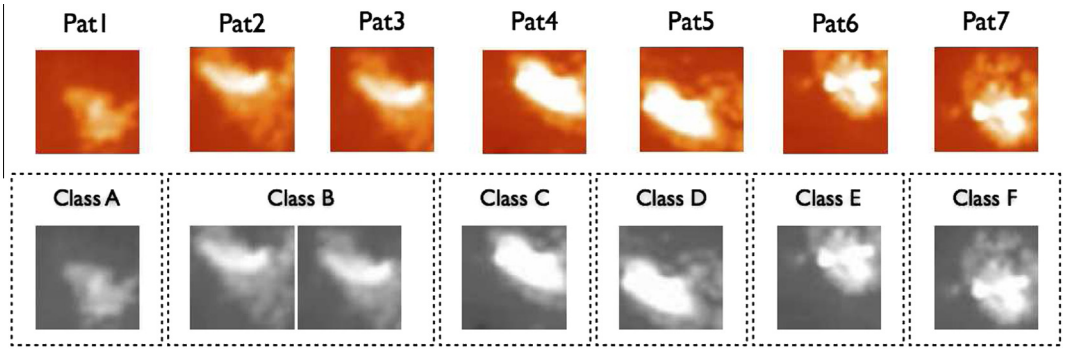


Fig. 6. Phase I: tolerance classes.

Table 2
Flare patterns and the corresponding dates.

Pat1	Pat2	Pat3	Pat4	Pat5	Pat6	Pat7
07/01/12	07/04/12	07/04/12	07/05/12	07/05/12	07/06/12	07/06/12

5.1. Phase I: tolerance class determination

Phase I automatically considers a set of solar sub-images and, based on their probe function values, collects those images identified as *near each other*, into tolerance classes. For the experiments with the SOL-FLARE system described in this section, four full solar images displaying flare events were chosen. The images are those from days 01, 04, 05 and 06 of July 2012. Also, five probe functions ($K = 5$) were selected, namely: contrast, correlation, energy, entropy, and homogeneity. The solar sub-images containing flare events in the four full solar images, were identified by human experts, and the seven square flare sub-images, Pat1, ..., Pat7, extracted (shown in the top row of Fig. 6). In Fig. 6 (bottom row), labels *ClassA*, ..., *ClassF* identify the six induced tolerance classes. Table 2 relates each flare sub-image to the corresponding full solar images it was extracted from.

Table 3
Probe function values associated with the seven flare sub-images.

	Contrast	Correlation	Energy	Entropy	Homogeneity
Pat1	0.5318	0.4451	0.7932	0.5717	0.7946
Pat2	0.6640	0.6029	0.8875	0.7529	0.9971
Pat3	0.6019	0.6209	0.8653	0.7100	0.9542
Pat4	0.7371	0.7516	0.9106	0.8104	0.8446
Pat5	0.7468	0.8644	0.9446	0.8796	0.9371
Pat6	0.8105	0.6324	0.8823	0.7489	0.6563
Pat7	0.6622	0.8300	0.9067	0.7966	1.0000

Table 4
L_2 norm values between each pair of sub-images.

	Pat1	Pat2	Pat3	Pat4	Pat5	Pat6	Pat7
Pat1	0.0000						
Pat2	0.3566	0.0000					
Pat3	0.2926	0.0904	0.0000				
Pat4	0.4575	0.2296	0.2439	0.0000			
Pat5	0.6142	0.3248	0.3546	0.1795	0.0000		
Pat6	0.4138	0.3717	0.3663	0.2442	0.4091	0.0000	
Pat7	0.5204	0.2259	0.2422	0.1900	0.1497	0.4267	0.0000

Table 3 relates each flare sub-image to the corresponding full solar images it was extracted from.

After converting the seven sub-images from RGB to grayscale representation, their probe function values were computed as shown in Table 3. The L_2 norm of each pair of sub-images were then computed and the results are shown in Table 4. The seven images converted to grayscale were then grouped using a tolerance relation, taking into account the values of the five probe functions as well as $\varepsilon = 0.1000$, by Algorithm 2. Sub-images Pat2 and Pat3, for instance, ended up in the same tolerance class (*ClassB*), as can be seen in the bottom row of Fig. 6. Notice that the difference in values between the L_2 norm for Pat2 and Pat3 is $0.0904 \leqslant \varepsilon$, i.e., they belong to the same tolerance class. Each one of the remaining sub-images defines a distinct singleton tolerance class, namely, A, C, D, E and F, as shown in the bottom row of Fig. 6.

5.2. Phase II: flare detection

The Phase II, as described by Algorithm 3 in Section 4.2, uses the information obtained in Phase I, i.e., the set of representatives of

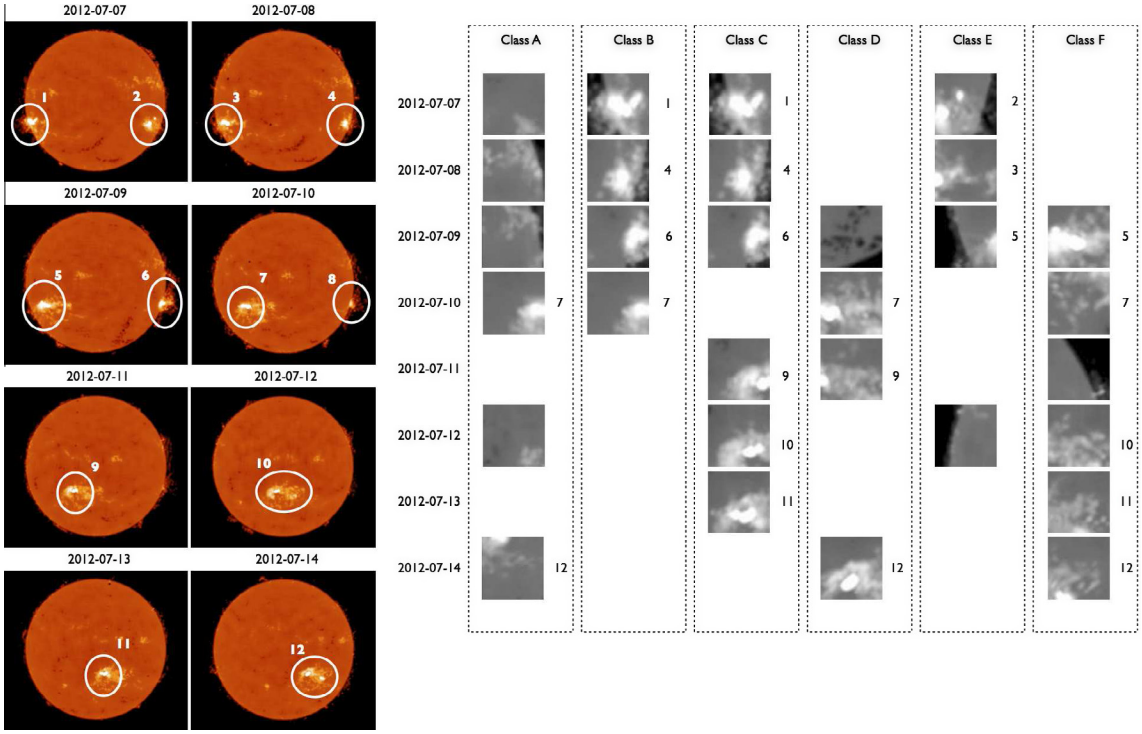


Fig. 7. Solar flares from Phase II.

images representing flare events and their associated tolerance class, inferred from the tolerance relation established between them.

The six tolerance classes of flare events detected in Phase I, each represented by one representative, were then used as input to the detection phase. The flare detection process was conducted on a set of 8 full solar images extracted from the Nobeyama Radioheliograph repository (images from the day 7th up to 14th of July 2012). None of the eight images were used in Phase I and each of them was defined by 500×500 pixels; square sub-images extracted from each of them were defined by 50×50 pixel windows.

Table 5
SOL-FLARE detection process described in Algorithm 3.

Class	1	2	3	4	5	6	7	8	9	10	11	12
A							x					x
B	x			x		x	x					
C	x			x		x			x	x	x	
D							x		x			x
E		x	x		x							
F					x		x			x	x	x
Detection	T	T	T	T	T	T	T	F	T	T	T	T

The set of $N = 8$ full solar images produced a total of 800 solar sub-images (100 per full image). After converting all of them from RGB to grayscale, the five probe function values associated to each of them were calculated. Finally, the L_2 norm between the probe function values of each sub-image and the probe function values of each one of the six representatives obtained in Phase I are computed. The code implementing Algorithm 3 was run 8 times, one per each full image. Fig. 7 shows the result of the solar flare detection process. On the left side of Fig. 7, the 8 full solar images used as input are sequentially shown, in chronological order. In each of them, the solar flares (visually detected by experts) are identified with a white circular mark, and sequentially labeled with numbers, from 1 to 12. On the right side of Fig. 7, each of the $NC = 6$ column labels represents a flare tolerance class (A, B, C, D, E and F) identified in Phase I and each of the 8 rows represents one of the 8 full solar images, labeled with the corresponding date it was obtained. Notice that more than one flare event can be in the same row.

The sub-images shown on the right side are those that have their probe function values 'close' to those of the representative of the tolerance class described by the column they belong to. There are 30 sub-images from which 4 sub-images are repeated twice, resulting in 26 different sub-images. On the right side of most of the sub-images, a number identifying the corresponding

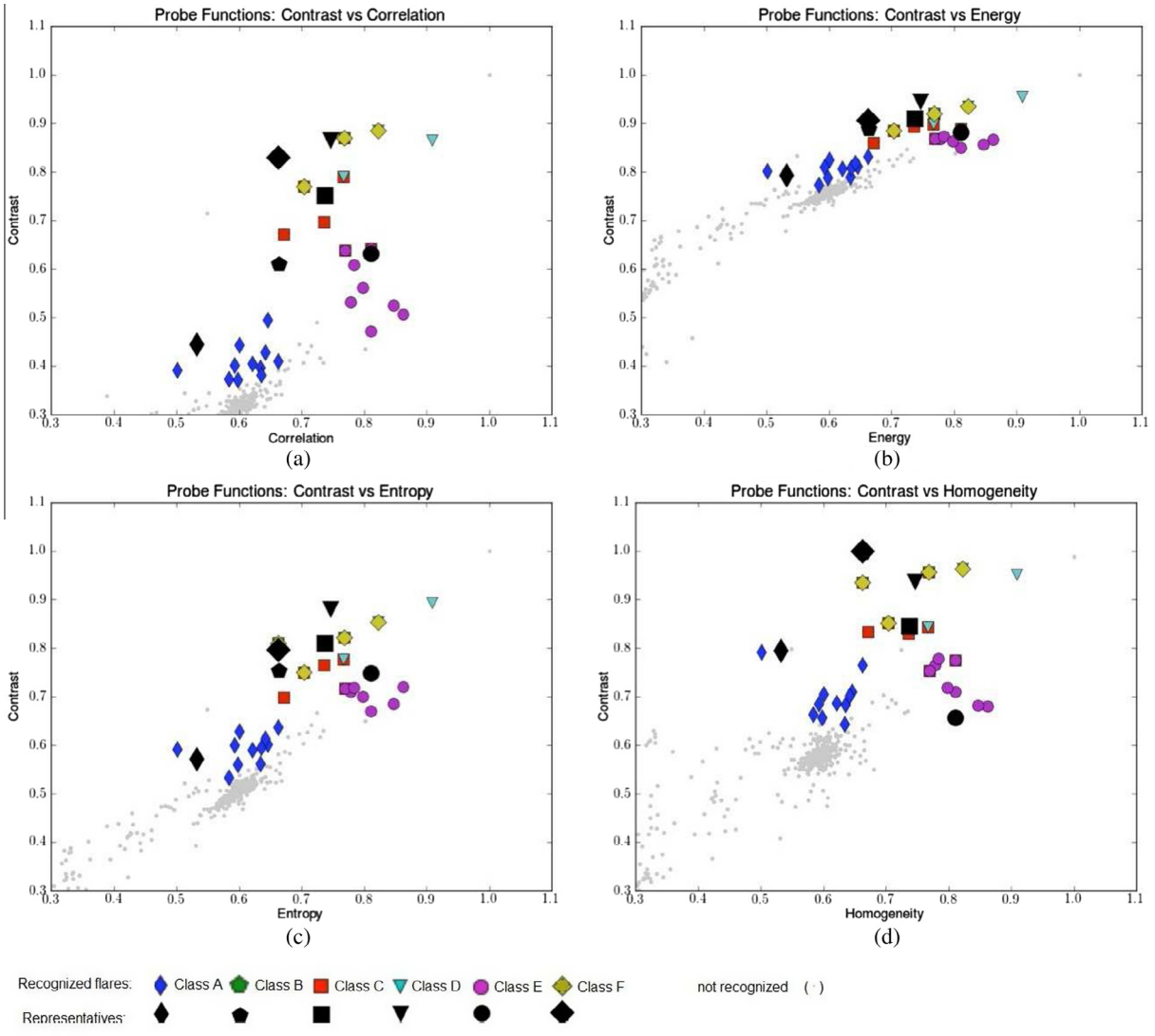


Fig. 8. GLCM contrast vs. correlation, energy, entropy, and homogeneity.

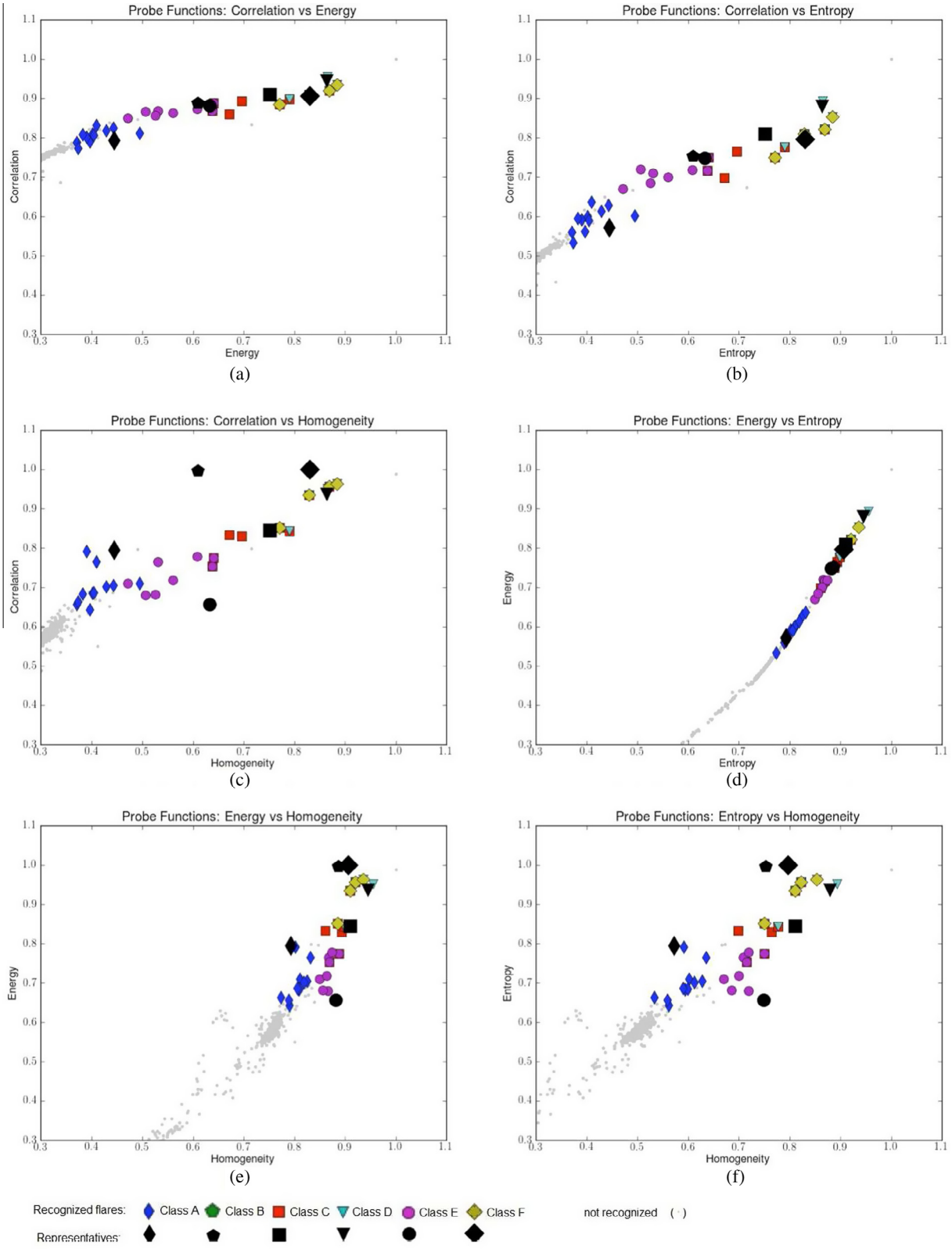


Fig. 9. Comparison of GLCM properties.

flare on the full solar image, is shown. Sub-images without a corresponding number represent a situation of false positive.

In the experiment, eleven out of twelve flare events were detected, which represents a success rate of approximately 91%. The bottom row of Table 5 shows the 'detection' status of each one of the 12 flare samples, where T = true and F = false. Some flare events were detected in different tolerance classes, as presented in Table 5. This is due to the occurrence of different sub-images of the same flare, such as flare 5 in tolerance classes E and F, flare 7 in tolerance classes D and F, flare 9 in tolerance

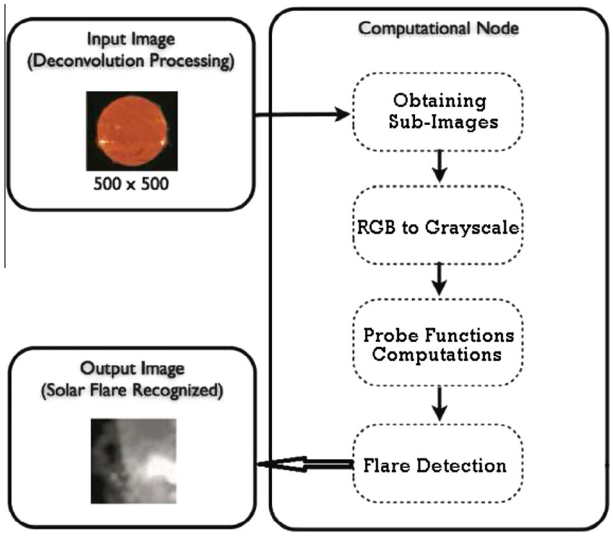


Fig. 10. GPU–CUDA processing.

Table 6
Real-time flare detection system processing results. Best results are bold faced.

Processor(s)	Allocated memory (KB)	Processing time (ms)	Speedup
CPU		2033.9	1
1 × CPU–GPU	720	20.4	99.7
2 × CPU–GPU	**360**	**10.2**	**199.4**

classes C and D, flare 10 in tolerance classes C and F, flare 11 in tolerance classes C and F and flare 12 in tolerance classes A, D and F. Besides, some sub-images were also detected in different tolerance classes, because of the proximity of their L_2 norm, such as flare 1, detected in tolerance classes B and C, flare 4 detected in tolerance classes B and C, flare 6 detected in tolerance classes B and C and flare 7 detected in tolerance classes A and B. This is consequence of using a tolerance relation where the set of elements is not partitioned into disjoint sets, but in overlapping sets (i.e., an element can belong to more than one set).

5.3. Analysis of probe functions in flare detection

In this section, the scatter plots for the different probe functions used during the flare detection experiment are given. In Fig. 8 each of the 6 flare tolerance classes is identified with a different symbol. The tiny gray dots represent sub-images that have not been detected. It can be noticed that in all plots, the non-detected sub-images have smaller probe function values than those in the detected sub-images. This is due to the fact that the probe functions representing flare sub-images have high values. It can be noticed that in Fig. 8(a), (c) and (d), i.e., in plots referring to contrast *versus* correlation, entropy and homogeneity, respectively, there is a greater degree of data dispersion than in Fig. 8(b). This type of dispersion promotes a better separation between tolerance classes and, consequently, helps to identify a group of more suitable probe functions for discriminating them. Fig. 9 also shows the pairwise plots comparing probe functions, identifying how they are related to each other in the experiment. The results suggest that three probe functions are correlated. The similarity of plots in Fig. 9(c), (e) and (f) reflect the fact evidenced in Fig. 9(a) and (b), namely, correlation, entropy and energy are correlated. In Fig. 9(d), the energy × entropy plotting follows the exponential form of the GLCM properties. Energy sums squared elements and entropy sums the product of an element by its logarithm.

5.4. Parallel processing results for the SOL-FLARE system

The process carried out during Phase II of the SOL-FLARE system has as input the full solar image, obtained from the deconvolution process, using the CLEAN algorithm (Section 2.2). Considering the proposed GPU–CUDA framework (see Fig. 10), parallel processing starts by transferring the data that represents the solar image from

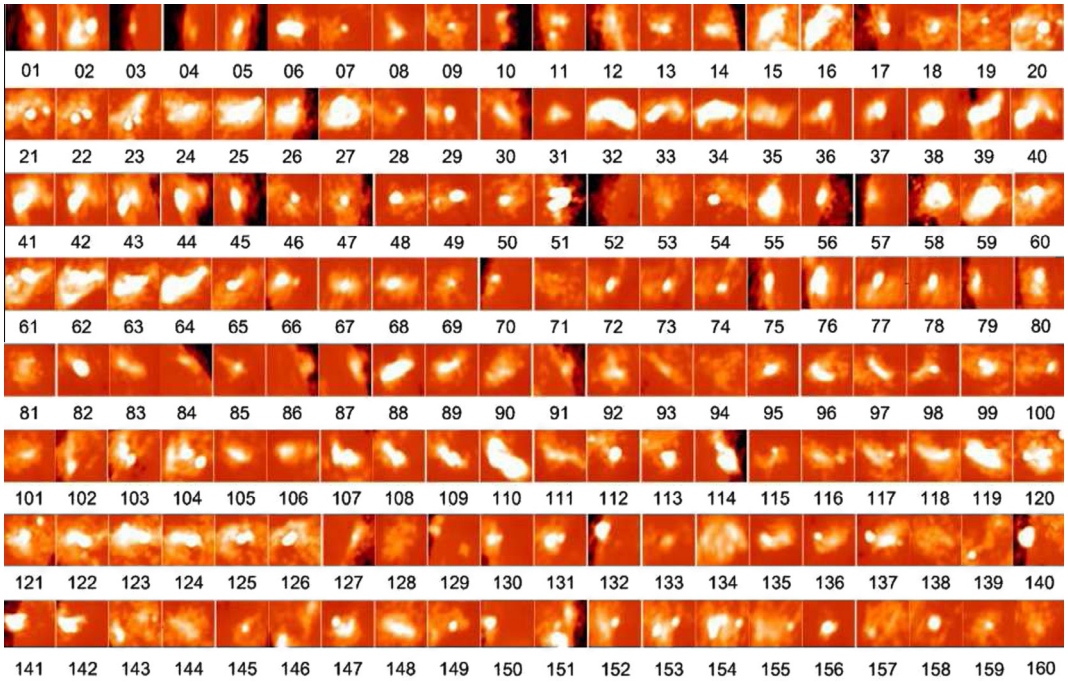


Fig. 11. Flare samples of 50 × 50 dimension, from the years 2004 up to 2013, extracted from Nobeyama Radio Heliography [18] enumerated from 1 up to 160.

the host (PC) memory into the GPU memory and then divided into sub-images of size 50×50 pixels. Information about each sub-image retained in the GPU memory is stored in data structures composed of fields: image data, probe function values (contrast, correlation, energy, entropy, homogeneity) and a flag value (conveying the information if the corresponding image data represents (or not) a flare event).

The full solar image, represented by 500×500 pixels, is then segmented into 100 sub-images, each defined by 50×50 pixels.

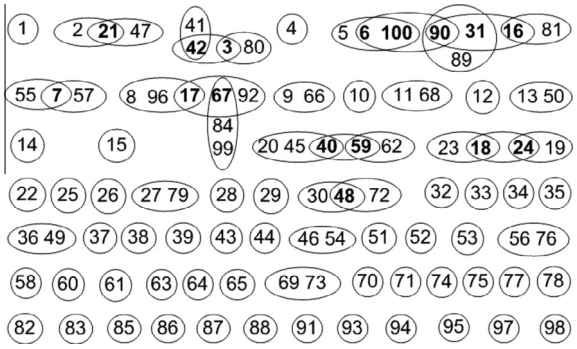


Fig. 12. SOL-FLARE results with 100 images and $\varepsilon = 0.015$, resulting in 78 tolerance classes.

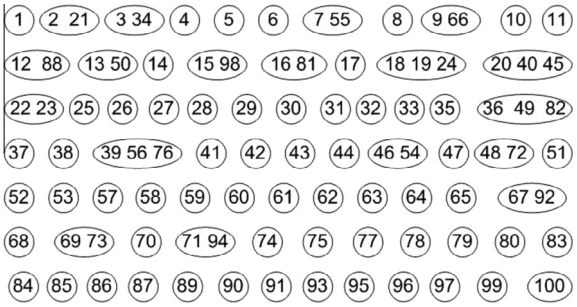


Fig. 13. K-means results with 100 images and $k = 78$ clusters.

Table 7
Results from experiment Phase I (see Figs. 12 and 13).

SOL-FLARE		K-means	
No. tolerance classes having	No. images	No. clusters having	No. images
8	3	4	3
23	2	14	2
47	1	60	1

Next, each sub-image is converted from RGB into grayscale, and the probe function values associated with each of them are computed. Next the L_2 norm between probe function values associated to sub-images and probe function values associated to representatives is calculated and each sub-image is considered detected when it is at least in one of the classes identified in Phase I or dismissed as a flare event. During the computational node processing, the same GPU memory is used with the same data structure. In the experiments reported here, two computational nodes, were used. The CPU–GPU pair was composed of 2.6 GHz Core i7 processor, and NVIDIA GeForce GT 650 M GPU, with 384 cores. The memory was allocated for the 500×500 pixel solar image processing and the processing time was measured. The speedup relative to single CPU processing is given in Table 6.

6. SOL-FLARE system vs. K-means algorithm

To provide an experimental validation of the proposed system, experiments were conducted on a larger image set of 160 solar sub-images and the results compared with results of the well-known K-means algorithm. The Receiver Operating Characteristic (ROC) [21] graphical plot was then used for analyzing the results. The K-means clustering algorithm is a strict partitional method where an element belongs to only one cluster, whereas the TNS-based SOL-FLARE can be considered as a non-partitional method, where elements can belong to more than one tolerance class.

As input to both algorithms, subsets of the initial 160 (50×50 pixels) flare sub-images, extracted from the data repository produced by Nobeyama Radioheliograph [18], as shown in Fig. 11, were used.

6.1. Experimental results: SOL-FLARE vs. K-means

To compare the performance of the two algorithms in a fair manner, the number of the induced tolerance classes, by SOL-FLARE (Phase I), was used as the value of k for the K-means algorithm. An experiment using a subset of 100 sub-images from the 160 initial sub-images, was given as input to SOL-FLARE. The tolerance level ε was set to 0.015 and the value of k, for running K-means, was established as the number of tolerance classes obtained by SOL-FLARE, i.e., 78. Fig. 12 shows the distribution of sub-images per tolerance class, by SOL-FLARE, and Fig. 13 shows the distribution per cluster, by K-means. Table 7 summarizes the results presented in Figs. 12 and 13.

6.2. Receiver Operating Characteristic (ROC) of the flare detections

In this section a number of sub-images either having a flare event or not having it, were used to determine the Receiver

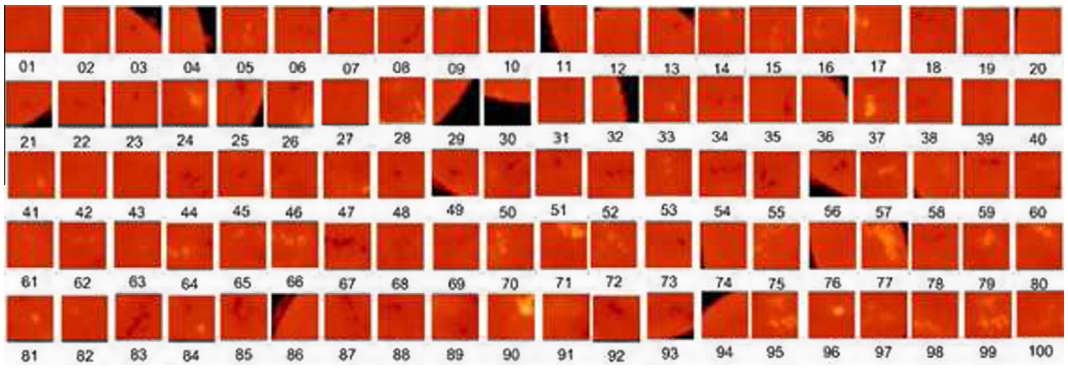


Fig. 14. Non-flare images used for the ROC curve experiments.

Operating Characteristic (ROC) curve, aiming to analyze the performance of the two algorithms for different values of the discrimination threshold, η. ROC curve is a graphical plot used for comparing the performance of a binary classifier system as its discrimination threshold varies. It is obtained by plotting the fraction of the true positives out of the total actual positives, called true positive rate (TPR), *versus* the fraction of false positives out of the total actual negatives, called false positive rate (FPR) at various threshold η settings.

In the following analysis, 60 flare sub-images, not used in Phase I, extracted from the 160 flare sub-images shown in Fig. 11, together with a set of 100 non-flare sub-images shown in Fig. 14, extracted from the Nobeyama Repository [18], were used. The 100 non-flare sub-images in Fig. 14 where extracted from several solar disk inactive region images.

For the ROC experiments, results from SOL-FLARE (Phase I) using as input 100 flare sub-images, varying the tolerance levels ε from the set {0.01, 0.015, 0.10, 0.20, 0.30, 0.40}, were used. Table 8 shows the number of tolerance classes (89, 78, 24, 21, 6, and 2) obtained for the six ε values using Algorithm 2. Next, the SOL-FLARE (Phase II) was run with a slightly modified Algorithm 3. Instead of a full solar image, the modified Algorithm 3 receives as input, subsets of the images shown in Figs. 11 and 14. This is justified since, to construct a ROC curve, a known set of positive and negative inputs is necessary to compute the true positive and false positive rates.

Table 8
Tolerance levels vs. tolerance classes.

ε	No. tolerance classes
0.01	89
0.015	78
0.10	24
0.20	21
0.30	6
0.40	2

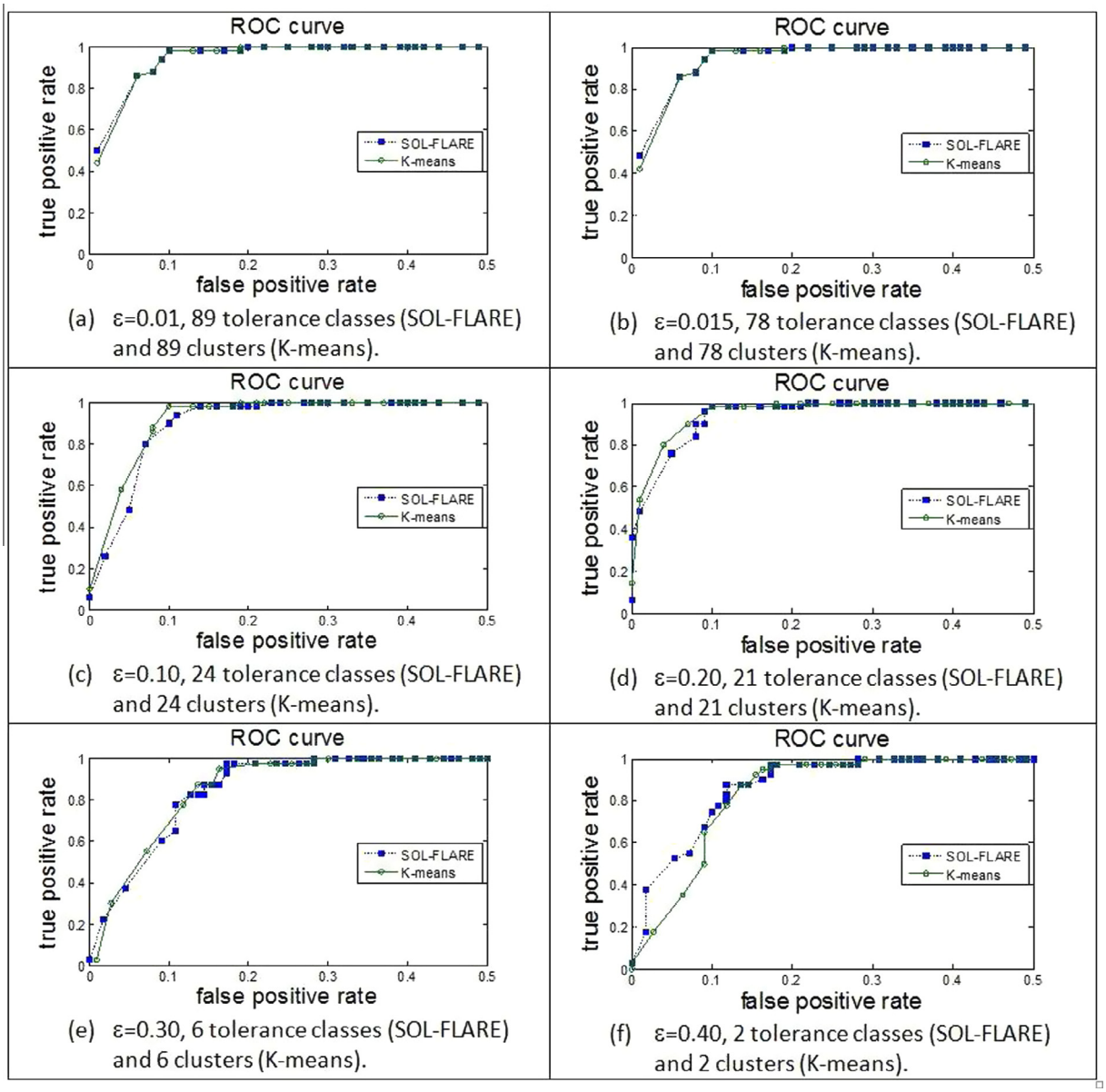


Fig. 15. ROC curves for both, SOL-FLARE and K-means. The ROC curve threshold η varies from 0 to 0.5, step 0.01.

For each tolerance level and its corresponding tolerance classes, the discriminating threshold η varied from 0 up to 1, with step 0.01, for the ROC experiment. For each value of threshold η, the flare detection algorithm (modified Algorithm 3) was executed 160 times, 60 for flare sub-image inputs and 100 for non-flare sub-image inputs. For an input image I, the threshold η corresponds to the maximum distance between probe vector values that describe I and those of each of the tolerance class representative C, which would allow I to be detected as a flare image. The η value that produces a high TPR and a low FPR would be the best choice.

Analysing the ROC curve results (Fig. 15), aiming at high TPR and low FPR, the following situations deserve attention. In Fig. 15(a) for $\eta = 0.01$ (second blue square point), we obtain a value of TPR > 0.85 and FPR < 0.1 with 89 tolerance classes. In Fig. 15(b) for $\eta = 0.01$ (second blue square point), we obtain a value of TPR > 0.85, with FPR < 0.1 with 78 tolerance classes. In Fig. 15(c), for $\eta = 0.03$ (fourth square point), we obtain a value of TPR = 0.8, with FPR < 0.1 with 24 tolerance classes. In Fig. 15(d) for $\eta = 0.04$ (fifth square point), we obtain a value of TPR > 0.8, with FPR < 0.1 with 21 tolerance classes. In Fig. 15(e), for $\eta = 0.06$ (seventh square point), we obtain a value of TPR > 0.8, with FPR > 0.1 with 6 tolerance classes. In Fig. 15(f), for $\eta = 0.08$ (ninth square point), we obtain a value of TPR > 0.8, with FPR > 0.1 with 2 tolerance classes.

Summarizing, the SOL-FLARE system with 89, 78, 24 and 21 tolerance classes, achieved a TPR value of > 0.8 with FPR < 0.1. This result can be viewed as a good one as far as flare detection is concerned. With the K-means algorithm, the TPR and FPR values obtained are very close to the corresponding values obtained by SOL-FLARE. The existing differences between their results are due to their intrinsic differences and also, to the different η values that promote the best result for each of them.

The use of tolerance classes by the SOL-FLARE system reduces the processing time in a GPU–CUDA environment, due to the use of representatives. Flare detection results confirm that by using SOL-FLARE, with a varying number of tolerance classes such as 89, 78, 24, and 21, TPR > 0.8 and FPR < 0.1, are obtained. When the number of tolerance classes becomes smaller, such as 6 or 2, a TPR value > 0.8 can be achieved but with FPR > 0.1, which can be considered as a non-satisfactory result. As a final remark, it can observed that a choice of 21 tolerance classes can be a reasonable choice, since fewer classes means a reduction in the number of representatives to be dealt with.

7. Conclusion

This article proposes a GPU–CUDA as a scalable and general framework suitable to be used in applications involving the processing of signals captured by radio interferometers. The article also presents and discusses the implementation of a solar flare detection system, named SOL-FLARE, suitable for the proposed GPU–CUDA framework, which helps the automatic detection of solar flare events in solar images. The system is based on the notion of tolerance (specifically tolerance near sets) in a perceptual system and uses a set of probe functions namely, contrast, correlation, energy, entropy, and homogeneity, for describing images. A probe function based approach is very suitable for the GPU–CUDA framework. The data structure used to represent images in the GPU memory reduces the size of the memory to be allocated, since all the necessary information for conducting the detection process is stored in the allocated memory in the data structure. The use of a tolerance relation makes it possible to implement simple probe functions that result in a straightforward flare detection implementation using an object oriented paradigm. In the experiments, with images obtained by the Nobeyama Radioheliograph, the SOL-FLARE results were analyzed and compared, with those obtained with the classical K-means algorithm, with the aid of ROC curves. Although the SOL-FLARE produces more diversity in class sizes when compared with K-means, this does not affect the GPU–CUDA system performance, since the elements in a class are represented by their representatives. The flare detection experiment confirmed that the flare detection using SOL-FLARE results in TPR close to 0.8, with FPR close to 0.1, with varying number of tolerance classes, so that the GPU–CUDA system should use the one with the smallest number of tolerance classes that yields reasonable TPR and FPR. The contribution of the article is a novel and unique application of a tolerance near set-based approach in solar flare detection. The results are promising and future work will include the use of tNM measure (see Section 1.1) in the SOL-FLARE system.

Acknowledgements

We gratefully acknowledge the suggestions and corrections made by the anonymous reviewers of this article.

References

[1] T. Alusaifeer, S. Ramanna, C.J. Henry, J.F. Peters, GPU implementation of MCE approach to finding near neighbourhoods, LNAI 8171 (2013) 251–262.
[2] J. Burg, Modern Spectral Analysis, IEEE Press, NY, 1978.
[3] A. Deller, S. Ingay, M. Bailes, C. West, Difx: a software correlator for very long baseline interferometry using multiprocessor computing environments, Publ. Astron. Soc. Pac. 119 (2007) 318–336.
[4] R. Engelen, Gsoap 2.8.15 User Guide, Tech. Rep., GENIVIA INC, 2013 <http://www.cs.fsu.edu/engelen/soapdoc2.pdf>.
[5] G. Gary, Physics 728. Radio Astronomy: Lecture No. 6:, Tech. Rep., New Jersey Institute of Technology, 2010 <http://web.njit.edu/gary/728/Lecture6.html>.
[6] G. Poli, A Hybrid Computational Platform of Distributed Parallel Coprocessing Using Web Services, Applied to Radio Interferometry (in Portuguese), Ph.D. thesis, Department of Computer Science, 2013.
[7] R. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cyb. 3 (6) (1973) 610–621.
[8] A. Hassanien, A. Abraham, J. Peters, G. Schaefer, C. Henry, Rough sets and near sets in medical imaging: a review, IEEE Trans. Inf. Technol. Biomed. 13 (6) (2009) 955–968, http://dx.doi.org/10.1109/TITB.2009.2017017.
[9] J. Hennessey, D. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufman–Elsevier, 2012.
[10] C. Henry, Near Sets: Theory and Applications, Ph.D. thesis, Department of Electrical & Computer Engineering, supervisor: J.F. Peters, 2010 <http://wren.ee.umanitoba.ca>.
[11] C. Henry, J.F. Peters, Perception-based image classification, Int. J. Intell. Comput. Cyb. 3 (3) (2010) 410–430.
[12] C.J. Henry, S. Ramanna, Parallel computation in finding near neighbourhoods, in: Rough Sets and Knowledge Technology, Lecture Notes in Computer Science, vol. 6954, Springer, Berlin Heidelberg, 2011.
[13] C.J. Henry, S. Ramanna, Maximal clique enumeration in finding near neighbourhoods, Trans. Rough Sets LNCS 7736 (2013) 103–124.
[14] C.J. Henry, S. Ramanna, D. Levi, Qantifying nearness in visual spaces, Cybernet. Syst. 44 (1) (2013) 38–56.
[15] J. Högbom, Aperture synthesis with a non-regular distribution of interferometer baselines, Astron. Astrophys. Suppl. 15 (1974) 417–426.
[16] H. Hudson, B. Haisch, K. Strong, Comment on the solar flare myth by J.T. Gosling, J. Geophys. Res. 100 (A3) (1995) 3473–3477.
[17] R. Kane, Strong solar flares, weak geo-effectiveness, Indian J. Radio Space Phys. 41 (2012) 575–578.
[18] H. Nakajima et al., The Nobeyama Radio Heliograph, vol. 82, 1994.
[19] NVIDIA, Cuda Programming Guide v5.0, 2012 <http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf>.
[20] Z. Pawlak, Classification of Objects by Means of Attributes, Polish Academy of Sciences, 429.
[21] M.S. Pepe, Estimation and comparison of receiver operating characteristic curves, Stata J. 9 (1) (2009) 1–16 (16) <http://www.stata-journal.com/article.html?article=st0154>.
[22] J. Peters, Near sets. General theory about nearness of objects, Appl. Math. Sci. 1 (53) (2007) 2029–2609.
[23] J. Peters, Near sets. Special theory about nearness of objects, Fundam. Inform. 75 (1–4) (2007) 407–433.
[24] J. Peters, Tolerance near sets and image correspondence, Int. J. Bio-Insp. Comput. 1 (4) (2009) 239–245.
[25] J. Peters, Corrigenda and addenda: tolerance near sets and image correspondence, Int. J. Bio-Insp. Comput. 2 (5) (2010) 310–318.
[26] J. Peters, S. Naimpally, Applications of near sets, Not. Am. Math. Soc. 59 (4) (2012) 536–542.
[27] J. Peters, P. Wasilewski, Tolerance spaces: origins, theoretical aspects and applications, Inform. Sci.: An Int. J. 195 (5) (2012) 211–225, http://dx.doi.org/10.1016/j.ins.2012.01.023.
[28] J.F. Peters, P. Wasilewski, Foundations of near sets, Inform. Sci. 179 (18) (2009) 3091–3109.

[29] J. Poincaré, Dernières Pensées, Trans. by J.W. Bolduc as Mathematics and Science: Last Essays, Flammarion & Kessinger Pub., Paris & NY, 1913 2009.
[30] G. Poli, J. Saito, J. Cecatto, Mapreduce and object-oriented paradigm applied to the radio interferometer signal correlation in gpu environment, in: IADIS International Conference Applied Computing, Madrid, Spain, 2012.
[31] H. Sawant et al., Brazilian decimetric array (phase-i), Solar Phys. 242 (2007) 213–220.
[32] J. Skilling, R. Bryan, Maximum entropy image reconstruction. General algorithm, Mon. Not. Roy. Astron. Soc. 211 (1) (1984) 11–124.
[33] J. Skilling, S. Gull, The entropy of an image, SIAM Am. Math. Soc. Proc. Appl. Math. 14 (1984) 167–189.
[34] G. Taylor, C. Carilli, R. Perly, Synthesis imaging in radio astronomy ii, in: Astron. Soc. Pacific – Conf. Series, vol. 180, 1999, 729pp. <http://www.phys.unm.edu/~gbtaylor/astr423/s98book.pdf>.
[35] A. Thompson, J. Moran, Interferometry and Synthesis in Radio Astronomy, Krieger Publishing Co., Malabar, Florida, 1994.
[36] R. Tousey, The solar corona, in: Cospar Space Research XIII, Akademic-Verlag, Berlin, 1973.
[37] E. Zeeman, The topology of the brain and visual perception, University of Georgia Institute Conference Proceedings, Published in M.K. Fort, Jr. (Ed.), Topology of 3-Manifolds and Related Topics, Prentice-Hall, Inc., 1962, pp. 240–256.
[38] E. Zeeman, O. Buneman, Tolerance spaces and the brain, Towards Theor. Biol. 1 (1968) 140–151. Published in C.H. Waddington (Ed.), Towards a Theoretical Biology. The Origin of Life, Aldine Pub. Co.
[39] J. Zhang, X. Zhao, S. Zhang, S. Yin, X. Qin, Interrelation analysis of celestial spectra data using constrained frequent pattern trees, Knowl.-Based Syst. 41 (2013) 77–88.