Hindawi
Complexity
Volume 2017, Article ID 5137317, 14 pages
https://doi.org/10.1155/2017/5137317

WILEY | Hindawi

Research Article

A Distributed K-Means Segmentation Algorithm Applied to *Lobesia botrana* Recognition

José García,[1,2] Christopher Pope,[1] and Francisco Altimiras[1,3]

[1] *Telefonica Investigación y Desarrollo, Santiago, Chile*
[2] *Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile*
[3] *Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, Santiago, Chile*

Correspondence should be addressed to José García; joseantonio.garcia@telefonica.com

Received 1 May 2017; Accepted 4 July 2017; Published 9 August 2017

Academic Editor: Jia Wu

Early detection of *Lobesia botrana* is a primary issue for a proper control of this insect considered as the major pest in grapevine. In this article, we propose a novel method for *L. botrana* recognition using image data mining based on clustering segmentation with descriptors which consider gray scale values and gradient in each segment. This system allows a 95 percent of *L. botrana* recognition in non-fully controlled lighting, zoom, and orientation environments. Our image capture application is currently implemented in a mobile application and subsequent segmentation processing is done in the cloud.

1. Introduction

The grapevine moth or *Lobesia botrana* (Lepidoptera: Tortricidae) is an invasive insect species considered as one of the most damaging pests in vineyards *(Vitis vinifera L.)* [1]. Vineyards damage is caused by direct larval feeding inside grape berries, producing rot and dehydration, inducing lowers productive yields and increasing the susceptible to other diseases such as gray and black mold (*Botrytis cinerea* and *Aspergillus niger*) [2, 3]. The *L. botrana* is widely distributed in Europe and is considered as a major grapevine pest since the twentieth century. In South America, *L. botrana* was first detected in Chile, in the area of Linderos, metropolitan region in April, 2008 [4]. Today, *L. botrana* is spread to all grape growing regions of Chile.

L. botrana management has become particularly relevant in Chile due to the potential economical impact as the main grape exporter worldwide. The moth control is performed using traps containing pheromones which attracts the moth males [5]. These traps are located on vineyards within an area formed of radius of 500 meters from a detected outbreak [6]. Currently, there are about 33.000 traps distributed throughout Chile for the specific control of *L. botrana*. To calibrate and verify moth growth models, in order to take measures focused on the vineyards with outbreaks, it is needed to have information on the number of moths caught in each trap, in different time windows. However, the collection of this information is done manually, making the system costly in terms of time and not allowing to perform real-time actions on these vineyards. Therefore, it is needed to have a control system for automatically detection of *L. botrana*, to estimate the real number of the moths and to store the information associated with each trap for epidemiological studies. Given the number of traps and images captured, the system must have the ability to recognize specimens of *L. botrana* in large volumes of images. Additionally, because the jobs are executed in a specific period of the day, it is useful to manage the computing capacity according to demand.

One of the most satisfactory image recognition systems rely on binary classification using a sliding window approach. However, the sliding window approach strongly increases the computational cost, because the classifier function has to be evaluated over a large set of candidate subwindows [7]. Considering the need for massive image processing for the recognition of *L. botrana* specimens, this article proposes a novel method through a mobile application that captures images using a cell phone camera, together with an algorithm based on image segmentation, which performs

fast and automatic recognition of *L. botrana* specimens. The algorithm is based on using the k-means clustering technique together with gray scale and gradient descriptors to perform the classification [8]. Because the normal operation of the *L. botrana* recognition system may need to process over 30,000 images per day, a distributed version of the algorithm was implemented using Apache Spark as a distribution framework.

Experiments were performed where a version of the algorithm was evaluated using sliding window with the version that uses segmentation. The experiments evaluate the quality of the recognition and the execution times of both algorithms. For the algorithm that uses segmentation, we also evaluated the scalability of the distributed version. The results show that the segmentation algorithm has improvements to the execution time and the quality of the recognition with respect to the one that uses sliding window. The remainder of this paper is organized as follows. In Section 2 we present related work in agriculture and segmentation. The application that captures the images and stores them in the database is described in Section 3. A brief description of the distribution framework is done in Section 4. Section 5 describes the details of the segmentation algorithm together with its distributed version. Finally results and conclusions are described in Sections 6 and 7.

2. Related Work

2.1. Computer Vision in Agriculture. Computer vision has been used in agriculture and food processing over the last decades, with the aim to automate a variety of processes such us quality inspection, classification, and sorting of products [9]. The idea was to replace traditional manual operations that are labor intensive, slow, and prone to human error [10]. Despite the significant advances over the past decade, the creation of new algorithms, new paradigms, and new challenges to processing big complexity data allows applying computer vision research in real world problems, with a particular focus in agriculture [11–14].

One of these challenges is the application of image segmentation techniques, which separates the product region from background in image processing. It is one of the first steps in image analysis after the image capture to subdivide an image into meaningful regions. The segmentation result affects the subsequent image analysis. Image segmentation techniques in agriculture have been applied to the estimation of product size [15], shape [16], sorting [17], weed detection for site-specific treatment [18], and ground classification [19].

Another important area of research in the field of computer vision and agriculture has been the application of deep learning techniques [20–22]. This machine learning technique is based on the concept of neural networks and convolution to learn image features automatically by repeated training, error propagation, and incremental learning [20, 23–25]. Some deep learning architectures have been tested to identify moths and other insect pictures using images obtained mainly in laboratory conditions with great success [26–30].

2.2. Segmentation Techniques. In the next section we present the main image segmentation techniques currently being used.

(1) Edge Based Image Segmentation. Edge detection includes a variety of mathematical methods that is aimed at identifying points in a image at which the brightness changes sharply or has discontinuities. Using spectral methods and watershed morphological algorithms in [31], a segmentation framework based on edge detection was proposed. In [32] an edge detection based on illumination invariant feature detector phase congruency was proposed. The authors show that use of phase congruency for marking features has significant advantages over gradient-based methods. In [33] an edge based auto threshold select method to generate multiscale image segmentation was proposed. Band weight and Normalized Difference Vegetation Index (NDVI) are used to calculate edge weight.

(2) Threshold-Based Image Segmentation. In classical threshold image segmentation, an image is usually segmented and simply sorted to object and background by setting a threshold. But if there is complex information in the image, the threshold is not simple to obtain. An important line of research in threshold-based image segmentation has been to design algorithms that allow obtaining the optimum threshold. For example, in [34] a method to detect threshold based on entropy and applied to images in scale of grays was developed. Histograms were used in [35] to address the automatic detection of a threshold. In [36] a threshold-based level set approach including threshold-based segmentation and fast marching method was proposed. It was applied to medical image segmentation. In [37] they presented a new threshold segmentation method based on computational intelligence to improve the acquisition of images process in computer vision.

(3) Partial Differential Equation Based Image Segmentation. One way to perform segmentation is to detect the contours of the objects lying in the original image. The main idea of the PDE model is to translate the problem of segment objects into minimizing an energy function of a close curve. In literature there are numerous examples that apply PDE to segmentation. In [38] a variation model is presented using 4th-order PDE with 2nd-order PDE for the removal of finger vein image. In [39] a new segmentation model was used based on the geodesic contour model applied to color images. A new nonlinear partial differential equation (PDE) applied to gray images was proposed by [40].

3. Image Capture Application

A mobile and web application was developed in order to capture images and report relevant information to users. Figures 1 and 2 show screen shots for the mobile and web application. The mobile application is a native Android app. Images where taken on-field where traps hang, recording GPS position and date time of the capture. The image is then uploaded via cellular network (or Wi-Fi if available) to a

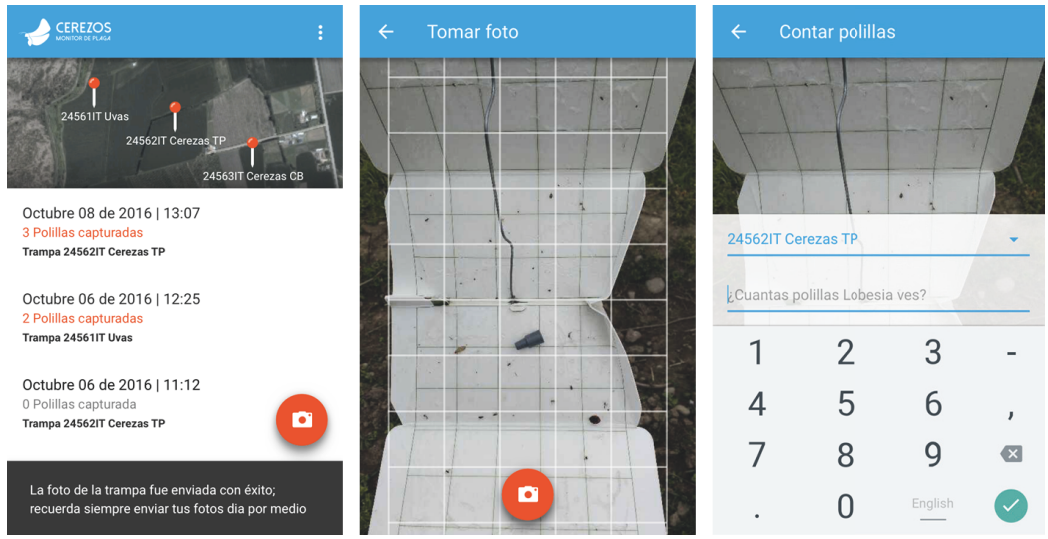


Figure 1: Mobile application.

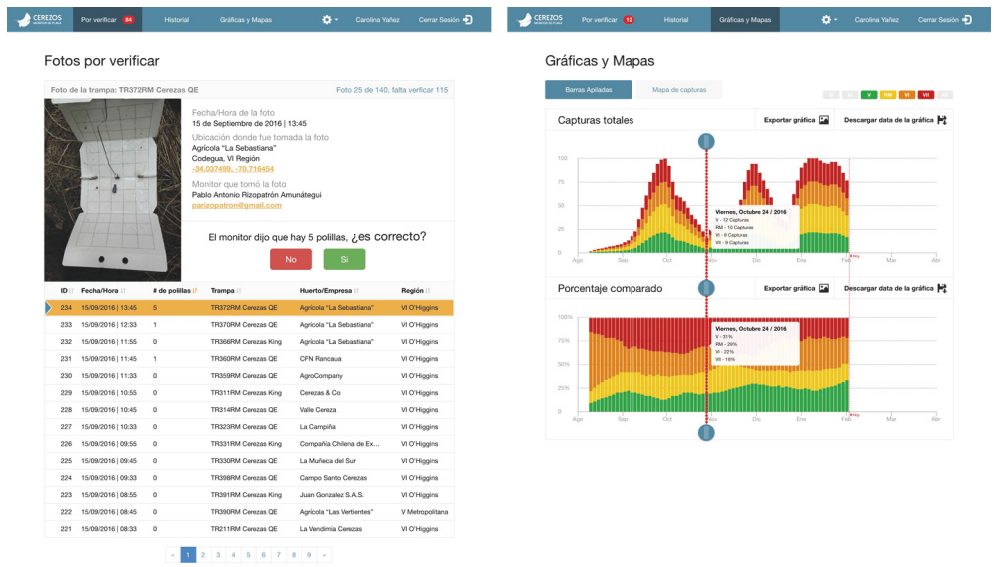


Figure 2: Web application.

cloud server to be processed. After processing, the image is displayed at the web application to an expert entomologist in order to asses the moth count and take action based on the pest situation for that specific place.

Figure 3 depicts the steps for capturing a trap image. First the trap must be opened and the sticky grid side must be placed facing forward to the capturing device. This device can be a cellular phone with a camera or a regular digital camera. Some traps have a triangular shape (delta traps) or a box shape. For some traps its sticky side has no grid, as shown in the example picture in Figure 3.

During September 2016 until February 2017, 50 traps where deployed between V, VI, VII, and metropolitan regions. A total of 26 users generated more than 400 on-field images. Figure 4 shows trap locations across V, VI, VII, and metropolitan regions. These traps were visited regularly once a week and a picture was obtained each time. Figure 5 shows example of images obtained on-field using the developed mobile application.

4. Spark Distributed Framework

This section describes the main concepts of the distributed Spark framework. This framework will be used in Section 5.4 for the distributed version implementation of the segmentation algorithm. Distributed implementation aims to address the problem of large volumes of images.

In recent years the amount of available data has increased significantly [41]. This is mainly due to the simple and inexpensive storage process. However, this amount of data is

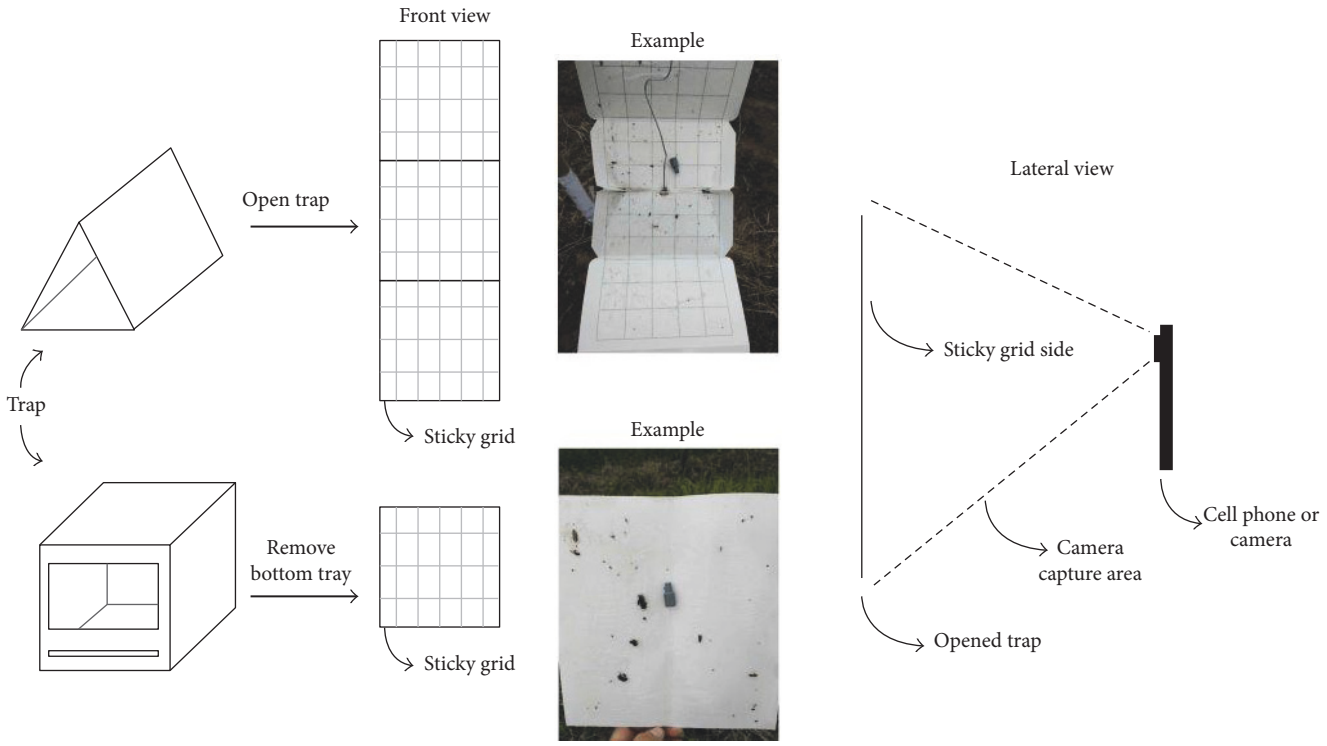


Figure 3: Image capture diagram.

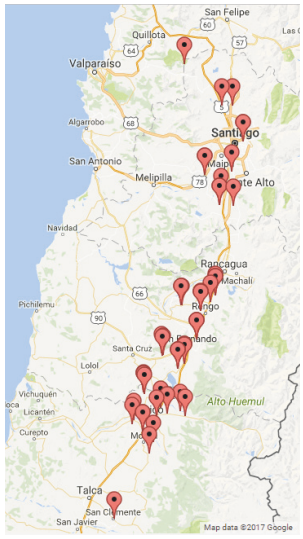


Figure 4: Deployment map between V and VII regions for 50 traps, Chile. *Source: Google Maps.*

useless without an adequate process of extraction of knowledge where we use machine learning methods. This volume, diversity, and complexity [42] of the data brings challenges to researchers, because traditional machine learning methods can not deal with this situation. Cloud-based technologies provide an ideal environment to handle the data challenge. A pioneer in addressing this problematic of massive and complex data was the MapReduce Framework [43]. This framework is based on the principle of locality of the data [44] which is implemented in a distributed filesystem. However, MapReduce is insufficient for applications that need to share data through multiple steps, or for iterative algorithms [45]. Many platforms for large-scale processing have recently emerged to overcome the issues presented by Hadoop MapReduce. Among them, Spark [46] appears as one of the most flexible and powerful engines to perform faster distributed computing in big data by using in-memory primitives.

Spark is a state of the art framework for high performance parallel computing designed to deal with iterative procedures that recursively perform operations over the same data. This has been used in machine learning algorithms [47], imaging processing [48], bioinformatics [49], computational intelligence [50], astronomy [51], medical information [52], and so on.

Spark was born as an in-memory cluster computing framework for processing and analyzing large amounts of data. It provides a simple programming interface, which enables an application developer to easily use the CPU, memory, and storage resources across a cluster of servers for processing large datasets in memory [46].

Spark has been positioned quickly as a general purpose platform. It provides a unified integrated platform for different types of data processing jobs. It can be used for batch processing, iterative process, interactive analysis, stream processing, machine learning, and graph computing.

Resilient Distributed Datasets (RDDs) are the core data units in Spark. These units are distributed and immutable;

Figure 5: Captures images examples.

that is, the transformation of RDDs is RDDs and fault-tolerant memory abstraction. There are two types of operations: transformations, which take RDDs and produce RDDs, and actions, which take RDDs and produce values. Various cluster management options can be used for run Spark, from simple Spark's standalone solutions, Apache Mesos, and Hadoop YARN [53].

Considering engineering applications we choose to use the Hadoop YARN management. Hadoop YARN, most recent implementation, uses cloud computing [54] which makes hundreds of machines provide services such as computing and storage on demand. Generally in-house implementations require large investments in hardware, software, and maintenance [54].

5. Algorithm

This section aims to describe the segmentation algorithm. First, a general description will be presented and then detailed by modules. The last section contains the distributed version of the algorithm.

Figure 6 shows the flowchart that allows constructing the classifier for *L. botrana* moths. As a dataset we consider 360 images obtained through the application described at Section 3. The 360 images correspond to three groups of 120 images each with resolutions 1280×720, 1920×1080, and 2048×1536. 100 images are left to perform validation. For each one of the images a preprocessing, then a segmentation, and finally the generation of the descriptors that allow performing the training for the SVM classifier are applied. For the classifier training, a set of 5018 segments without moths and 2136 segments with moths were selected.

Additionally with the goal of evaluating the segmentation algorithm with respect to the recognition of *L. Botrana* we compared the performance to different elements usually glued on the sticky floor of the trap like other insects, leaf pieces, and equipment used in the trap. With this purpose, we analyzed 5018 segments without moths and a subset of 1325 segments containing elements other than the bottom of the trap were extracted.

5.1. Preprocessing Stage. As the first activity of the preprocessing stage, a Median Filter is used with the intention of removing particles of dust and brightness that appear at the photographs. An example is shown in Figure 7. The filter used was a 4×4 matrix where each pixel has the same weight. Subsequently an equalization is applied using the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm. This algorithm uses histograms computed over different tile regions of the image. Local details can therefore be enhanced even in regions that are darker or lighter than most of the image. Then we apply 3 scaling ratios to the image: 0.75, 0.5, and 0.25. These scaling steps aim to consider that photographs can be taken at different distances from the cell phone. To perform the scaling, nearest-neighbor interpolation was used. Finally a gray scale conversion was executed. This conversion uses the ratio $0.3R + 0.6G + 0.1B$.

5.2. K-Means Segmentation Algorithm. The main objective of the project is to recognize *L. botrana* moths. A fundamental stage corresponds to the segmentation of moths. To perform the segmentation, each color image is scaled at 0.75, 0.50, and 0.25 and K-means clustering technique is applied to each image. In the definition of metric space to be able to apply K-means, the pixel distance shown in (1) and color distance shown in (2) are used. To achieve the proper segmentation a trade-off between pixel distance and color distance is realized. This trade-off is defined in (3). This type of segmentation has previously been used by [55] in benchmark datasets and in biomedical applications:

$$d_{xy} = \sqrt{\left(i_x - i_y\right)^2 + \left(j_x - j_y\right)^2}, \tag{1}$$

$$d_{\text{Lab}} = \sqrt{\left(L_x - L_y\right)^2 + \left(a_x - a_y\right)^2 + \left(b_x - b_y\right)^2}, \tag{2}$$

$$D_s = \sqrt{\left(d_{\text{Lab}}\right)^2 + \left(\frac{m}{S}\right)^2 \left(d_{xy}\right)^2}, \quad \text{where } S = \sqrt{\frac{N}{K}}. \tag{3}$$

D_s corresponds to the metric used to perform the segmentation and S is a normalizing factor, where N is the

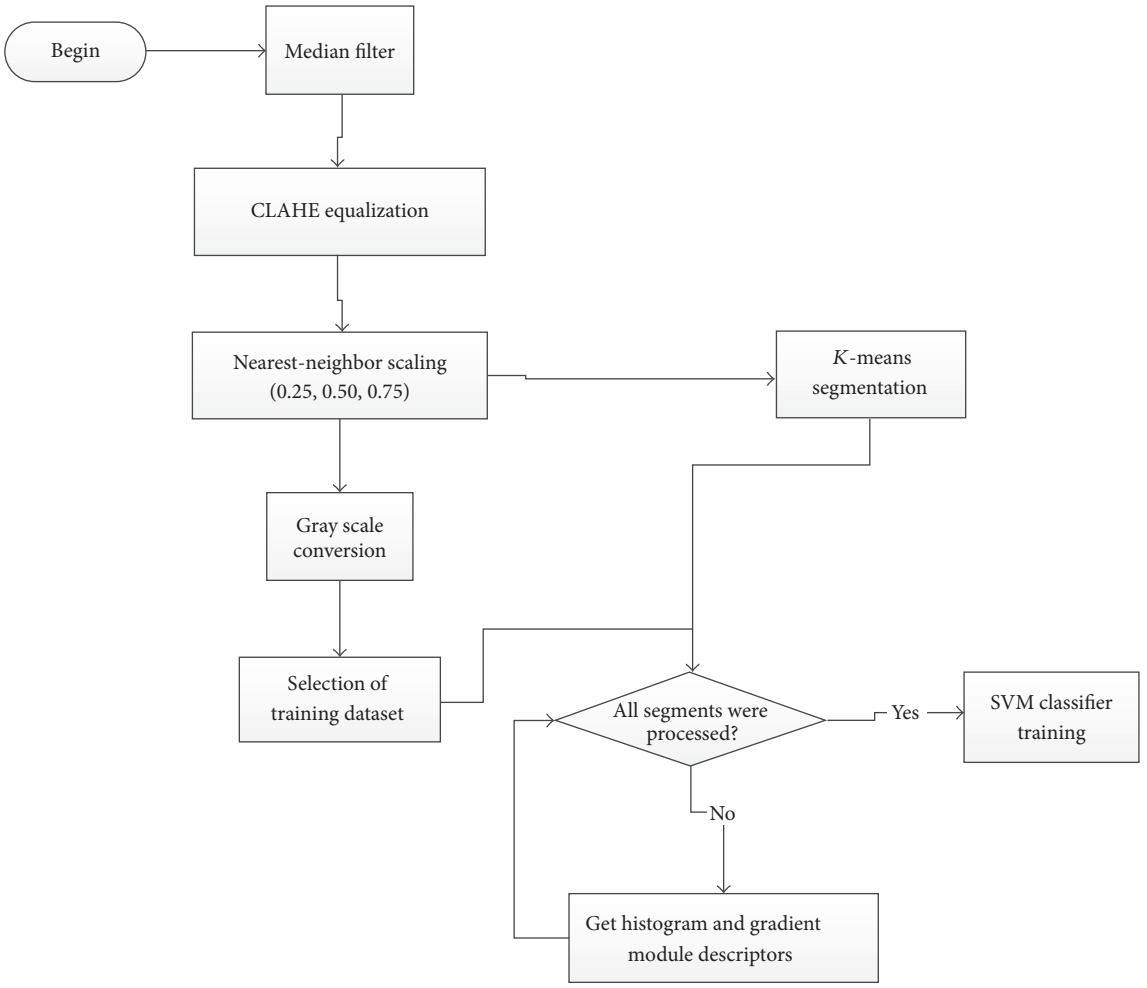
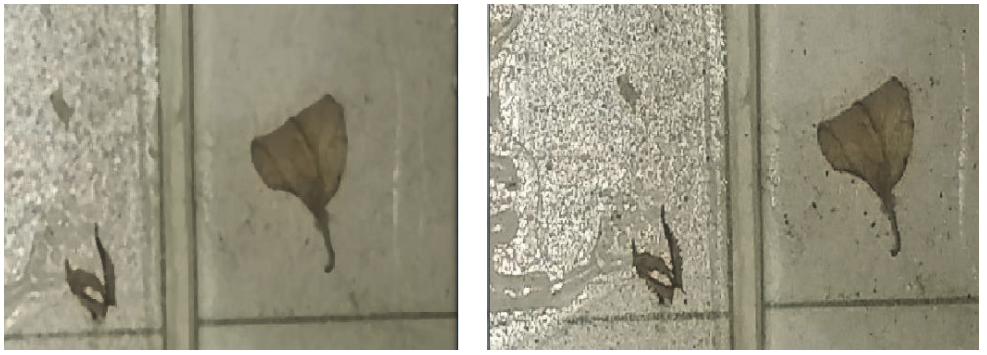


Figure 6: *Lobesia botrana* recognizer algorithm flow diagram.

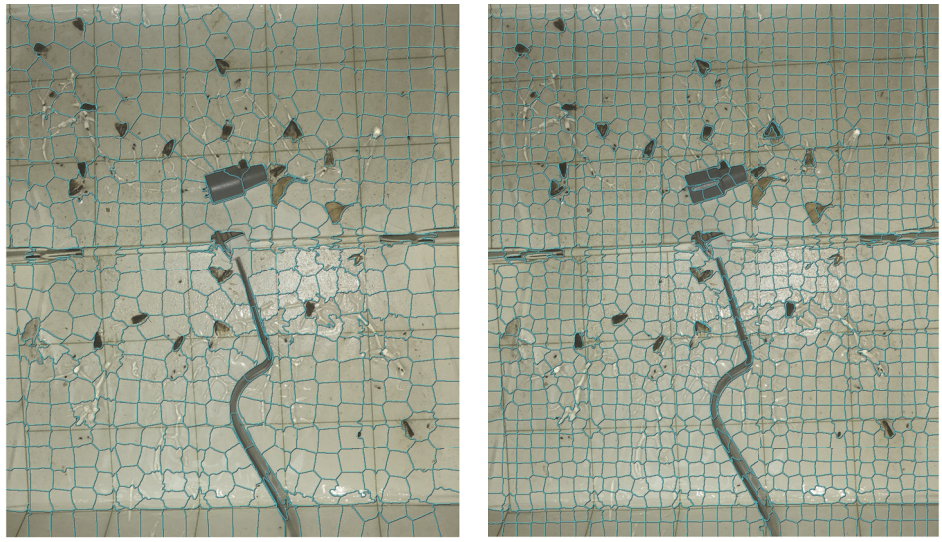


(a) (b)

Figure 7: (b) The original image, where powder can interfere with the histogram. (a) The processed image with equalization and median filter applied; the dust particles disappear.

number of pixels and K is the number of segments. The parameter m allows handling the weighting between the spatial distance and the colors distance. In Figure 8 it is possible to observe two segmentations. Figure 8(a) uses K = 500 and Figure 8(b) uses K = 1000.

5.3. Descriptors and SVM Classifier. After segmentation was performed, the next step is to differentiate the segments containing moths from those that do not include moths. To achieve this classification, gray scale segments are used. For each segment two descriptors are defined. The first



(a) (b)

Figure 8: (a) corresponds to a segmentation considering 500 segments. (b) corresponds to segmentation considering 1000 segments.

descriptor corresponds to the histogram of the gray values of the segment. The histogram is configured with 10 bins and each bin is normalized with the amount of pixels that the segment contains. This generates a 10-dimensional vector where each dimension has values in the interval [0, 1]. A second descriptor is obtained as a result of calculating the histogram on the gradient modulus. This histogram is also normalized by the total number of pixels in the segment. In this case the number of bins used is 5. The following steps are used to calculate the gradient modulus.

(1) Gradient Calculation. A Sobel [56] operator is used. The complete image, I, is used to calculate each component of the gradient in all pixels of the image:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 3 \end{bmatrix} * I, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I. \tag{4}$$

(2) Calculation of the Gradient Modulus. For all pixels in image I the gradient modulus is calculated according to

$$G = \sqrt{G_x^2 + G_y^2}. \tag{5}$$

(3) Histogram Calculation. The histogram of G is calculated for each segment obtained from the K-means segmentation process.

The last step corresponds to the training the classifier. At training, 5018 segments are used without moths and 2136 segments with moths are considered. A K-fold cross validation was performed with $K = 5$ to perform the parameter adjustment.

5.4. Distributed Algorithm. Due to the estimated volume of photographs that need to be processed and considering that the process of segmentation along with the generation of descriptors corresponds to the one who consumes the most time, in this section we present a distributed version of the algorithm for segmentation and generation of descriptors. This version was designed using the Spark distributed computing framework.

The algorithm can be divided into two main stages: a segmentation stage and a descriptor generation stage. In the segmentation stage the algorithm operates on the complete image. The result of this stage corresponds to a list of segments. The descriptor generation stage has as input the list of segments and, for each one of these segments, descriptors are calculated. The distributed algorithm will address the distribution of calculations in the segmentation process as well as the calculation of descriptors in the segment list.

Segmentation Stage. Let us assume that the list of images has been read from Hadoop distributed file system (HDFS), used to store images. To perform the reading, the binaryFiles function is used, generating the corresponding RDD files. In addition, the list c of initial centroids has been obtained, considering an equidistant distribution for the different centroids. The distribution process in the segmentation stage is performed at the pixel and centroid level, where a pixel corresponds to the 5-dimensional distance-color vector defined in (3). The first stage of the segmentation phase is to perform a mapping shown in Algorithm 1. Each pixel i is mapped to the pair ($closestC$, i) where $closestC$ corresponds to the centroid closest to pixel i. To perform this operation we use the Spark map transformation. The second stage is to perform a reducebykey, where the key corresponds to the centroid $closestC$(see Algorithm 2). The vector sum of the pixels is done in the 5-dimensional space, together with the count of the pixels associated with the centroid. To perform this

```
(1) Input Pixel i, ListCentroids
(2) Output Pixel i, ClosestC
(3) for each c in ListCentroids do
(4)   if D_s(p,c) < MinDist then
(5)     MinDist ← D_s(p,c)
(6)     ClosestC ← c
(7)   end if
(8) end for
(9) return (closestC, i)
```

Algorithm 1: Map segmentation function.

```
(1) Input keyValue closestC, Pixel i
(2) Output total, count
(3) for each c in closestC do
(4)   total ← total + i
(5)   count ← count + 1
(6) end for
(7) return (total, count)
```

Algorithm 2: ReducebyKey segmentation function.

```
(1) Input segment s
(2) Output segment s, descriptor d
    d ← getDescriptor(s)
(3) return (s, d)
```

Algorithm 3: Map descriptor function.

operation, consider the reducebykey action. Finally the value of the new centroid is updated.

Descriptor Generation Stage. This stage aims to perform the calculation of the descriptors that allows classification. At this stage the image is distributed at the segment level. To perform the distribution a map function is used that maps the segment to the pair (segment, descriptor). Each map involves the calculation on a segment (see Algorithm 3).

6. Results

6.1. Parameters Setting. Considering that each algorithm involved requires parametrization, in order to find the best parameters, it was considered to perform a k-fold cross validation for each of the combinations shown at the Range column in Table 1. In the cross validation, $K = 5$ was considered. As a measure of the best configuration, F_1-score = 2((precision ∗ recall)/(precision + recall)) was used. In Table 1, final column shows the selected value for the configuration that yields the best accuracy.

In order to have a better clarity of the effect of the segmentation on the classification, the parameter *NumSegment* was studied. In order to perform this study, an experiment

Table 1: Segmentation algorithm setting parameters.

Parameters	Description	Range	Final
GridSize	CLAHE grid size	[8×8, 16×16, 32×32]	16 × 16
clipLimit	Clipping limit histogram	[2, 3, 4]	3
Median Filter Grid	Medium filter grid size	[4 × 4, 5 × 5, 6 × 6]	4 × 4
NumSegment	Number of image segments for segmentation algorithm	[500, 1000, 1500]	1000
m	Distance-color trade-off	[10, 15, 20]	15
C	Penalty parameter C SVM	[0.1, 1, 10]	1

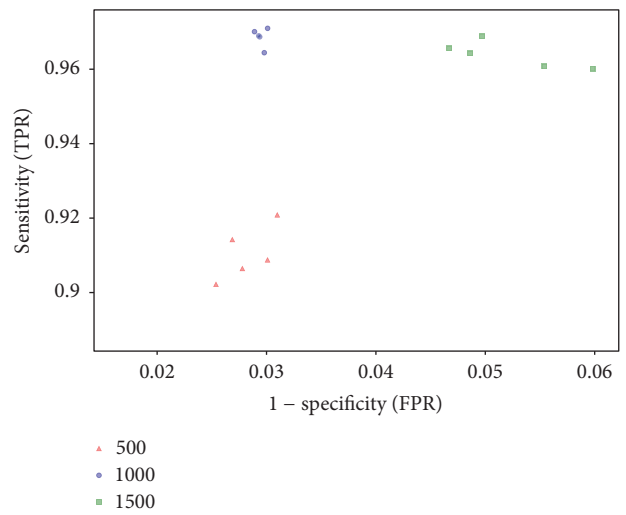


Figure 9: Sensitivity-specificity number segment evaluation.

was defined in which the remaining parameters are fixed considering the value of the final column and K-fold cross validation is performed with $K = 5$ considering the values 500, 1000, and 1500 in the parameter *NumSegment*. For each cross validation result, the True Positive Rate (TPR) and the False Positive Rate (FPR) were calculated. The result of the experiment is shown in Figure 9. It is observed that *NumSegment* = 1000, represented in blue color, corresponds to the best result, for the case *NumSegment* = 500 has a worse TPR performance and *NumSegment* = 1500 has a lower performance for FPR.

6.2. Descriptors Setting. As described at Section 5.3 a descriptor was used that combines a normalized histogram of gray scale with a normalized histogram using the gradient modulus. Equation (6) was used in order to determine the number of bins that best differentiates moth segments from those that do not contain moths

$$\text{Histogram Measure} = \sum_{i=1}^{n} \text{abs}\left(H_i(L) - H_i(\text{WL})\right). \quad (6)$$

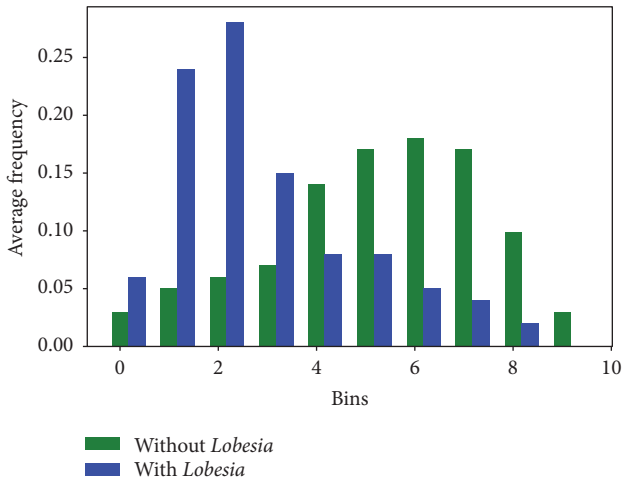


FIGURE 10: Gray scale descriptor *Lobesia* versus that without *Lobesia*.

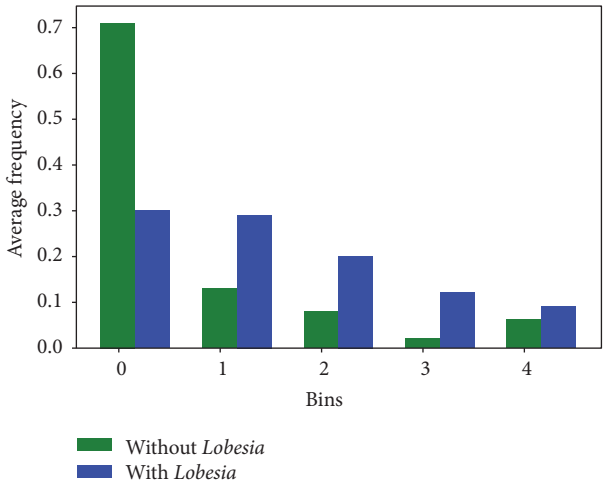


FIGURE 11: Gradient modulus descriptor *Lobesia* versus that without *Lobesia*.

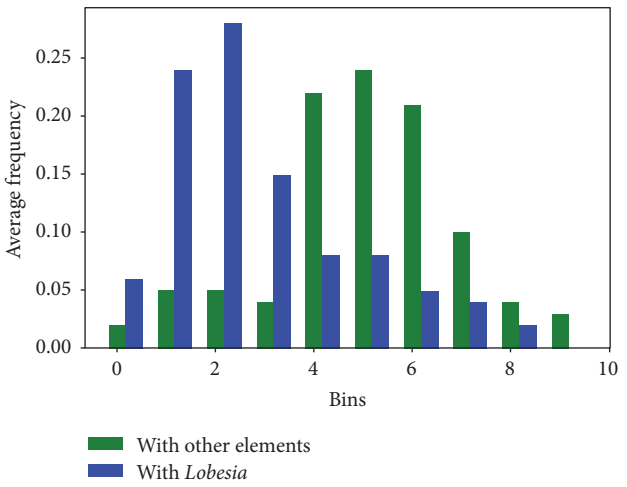


FIGURE 12: Gray scale descriptor *Lobesia* versus other elements.

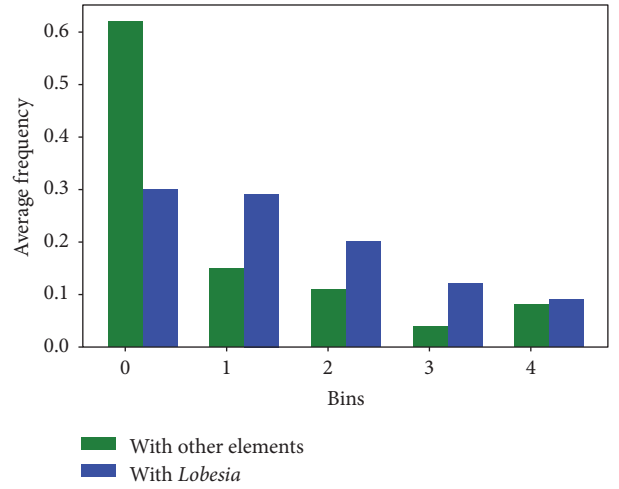


FIGURE 13: Gradient modulus descriptor *Lobesia* versus other elements.

L corresponds to the normalized average histogram of the segments with *L. botrana* and WL to the normalized average histogram of the segments without *L. botrana*. H_i corresponds to the function that returns the bin i of a histogram. For each bin, the value obtained from the *L. botrana* segments is subtracted to the value of segments without *L. botrana*. Then the modulus function and the sum on n bins are applied. The more alike the histograms are, the closer the *Histogram Measure* value is to 0; in our case we seek to maximize the *Histogram Measure* value. Ranges between 5 and 12 bins for the gray scale case and 4 to 10 bins for the gradient module were verified. The best results obtained after applying (6) are shown in Figures 10 and 11.

Finally, using the other elements dataset described in Section 5, we generate normalized average histograms for Gray and modulus of gradient. In Figures 12 and 13 the results are shown. Although the behavior of dataset histograms without *L. botrana* differs from histograms obtained by the dataset with other elements, when comparing the latter with *L. botrana* histograms, an adequate separation is observed.

6.3. K-Means Segmentation Algorithm Evaluation. The evaluation of the segmentation algorithm was performed using the 100 images that were not included at the parameters configuration stage. To perform the comparisons an algorithm was designed using sliding window. When we apply sliding window, it is assumed that objects have regular appearance that do not deform much. A database is built with *nxm* fixed size windows containing the centered object and others that do not contain it. This is used to train the classifier. A *nxm* window is slid across the entire image and the classifier is queried if each of these windows contains or not the object. The sliding algorithm uses the same steps and configurations as the segmentation algorithm described at Section 5.2, except in the K-means segmentation stage where it uses sliding window. The window size was 64×64 and the step size was 8 pixels.

Table 2: Confusion matrix results.

Scale	Segmentation				Sliding window			
	TP	TN	FN	FP	TP	TN	FN	FP
1	4336	95174	191	299	4138	94805	389	668
0.75	4337	95120	190	353	4205	94996	322	477
0.50	4269	95062	257	411	4097	95187	430	286
0.25	4285	95082	242	391	4142	95091	385	382
Average	4307	92110	220	363	4145	95020	382	454

Comparisons were focused on the quality of the recognition as well as the execution speed of the different algorithms. In the case of the quality evaluation, the recognition for the rescaled images at 1, 0.75, 0.5, and 0.25 was analyzed. For the execution of the instances we use a PC running Windows 10, on an Intel® Core i7-4770 processor with 16 GB in RAM and programmed in Python 2.7.

Table 2 shows the resulting confusion matrix for both algorithms. It should be noted that the segmentation algorithm was superior to sliding window in all cases. Because our matrix has unbalanced class distribution, the following indicators were considered for comparison:

(1) Sensitivity: TP/(TP + FN)

(2) Precision: TP/(TP + FP)

(3) F_1 Score: 2((precision $*$ sencitivity)/(precision + sencitivity))

(4) Mathews correlation coefficient: $(TP * TN - FP * FN)/\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}$.

The segmentation algorithm yielded an average specificity of 95.1%; in the case of sliding window this was 91.4%. In the case of the precision indicator, the segmentation algorithm yielded an average value of 92.2%, and the sliding window was 90.01%. The F_1 score yielded by the segmentation algorithm was 93.65% and in the case of sliding window was 90.76%. Finally for the Matthews coefficient the value was 93.36% for segmentation and 90.41% for sliding windows.

Additionally, a confusion matrix was constructed to evaluate the performance of the algorithm with respect to the other elements data (Table 3). This dataset was generated from the 100 images with 1135 segments containing other elements. The segmentation algorithm for the case of the other elements data, obtained an average specificity of 95.15%, an accuracy of 99.5%, an F_1 coefficient of 97.2, and a Matthews coefficient of 88.1%.

To perform the comparison of execution times, we considered the three groups of images with resolutions of 1280 × 720, 1920 × 1080, and 2048 × 1536. For each of the images the processing time is obtained and represented by a box plot shown in Figure 14. In this figure it is observed that the segmentation algorithm has a better execution time than sliding window. For the case of the segmentation algorithm the average time was 31.8 (s) for the group 1280 × 720, 51.5 (s) for the result 1920 × 1080, and 118 (s) for 2048 × 1536. When using sliding window the time was 316.5, 649.2, and 1515.5 respectively. Finally Figure 15 shows the classification result for two test images.

Table 3: Confusion matrix results with other elements.

Scale			Segmentation	
			True	False
Ground	1	*True*	4336	191
Truth		*False*	21	1114
Ground	0.75	*True*	4337	190
Truth		*False*	22	1113
Ground	0.5	*True*	4269	257
Truth		*False*	17	1118
Ground	0.25	*True*	4285	242
Truth		*False*	20	1115
Ground	Average	*True*	4306.8	220
Truth		*False*	20	1115

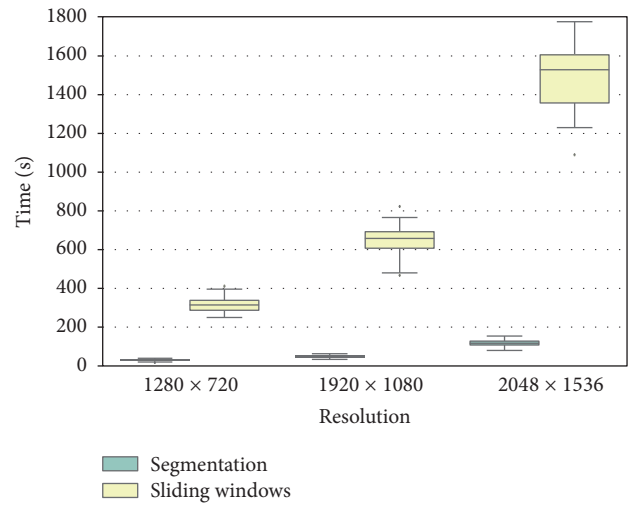


Figure 14: Time comparison using segmentation and sliding window techniques. Interquartile ranges: 25%, the median and 75% of the values.

6.4. K-Means Segmentation Algorithm Scalability. In the case of the distributed algorithm the experiments were focused on measuring its scalability. The algorithm was developed using Python 2.7 and Spark libraries. It was run on Azure platform using version 1.5.2 of Spark and Hadoop 2.4.1.

In order to evaluate the algorithm we considered the resolution, number of images, and different numbers of executors. As dataset the 360 images used to train and validate the algorithm were considered. These images were replicated

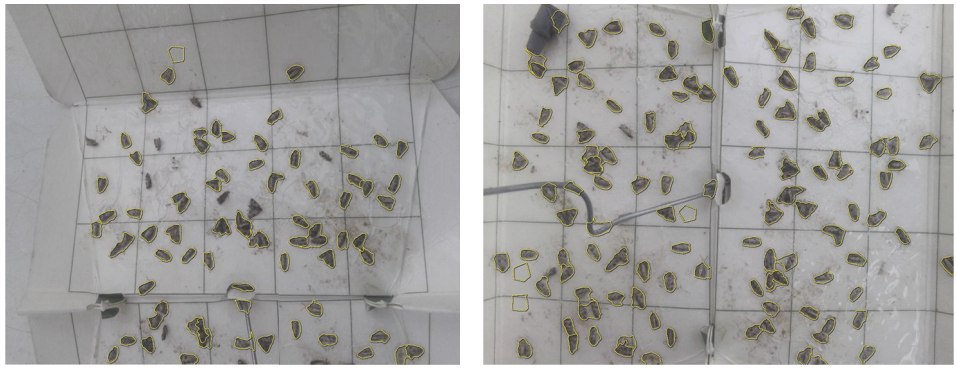


Figure 15: Example of processed images.

Table 4: Spark configuration.

num-executors	Executor-cores	Executor-memory (Gb)
1	3	4
2	3	4
4	3	4
8	3	4
16	3	4

1, 10, and 100 times in order to generate groups of 360, 3600, and 36000 images. Each group was divided according to their image resolution. For the scalability test 1, 2, 4, 8, and 16 executors were considered. The relative speed-up by group and resolution was measured. For the Spark configuration, three parameters were considered: num-executors which controls the number of executor requested, executor-cores property which controls the number of concurrent tasks an executor can run, and executor-memory which corresponds to the memory per executor. For the proper use of an executor it is recommended to use between 3 and 5 cores. The considered Spark settings are shown in Table 4.

Figure 16 shows the relative speed-up curves for Spark distributed implementation of segmentation algorithm. The relative speed-up is calculated with respect to the execution time obtained when performing the calculation using 1 executor. The blue line corresponds to the perfect result. Curves show at the three experiments that they continue delivering speed-up to 16 executors. When over 16 executors the performance is clearly sublineal. For the dataset with 360 images the deviation from the perfect result on average was 20.6%, for the case of 3600 images 25.6%, and for the case of 36000 images 32.1%. Regarding resolutions, the one that had the best performance in all cases was the one with the lowest resolution, but the effect of the resolution was small.

7. Conclusions

In this work we have presented a segmentation algorithm to perform the early recognition of *Lobesia botrana*. A mechanism for the adequate parameter setting selection and definition of the descriptors that allow classification was developed. Image capture mobile and web application was created for *Lobesia botrana* recognition. The native Android mobile application takes the images and records the GPS position and date, making possible the real-time moth counting. The image recognition system was validated during September 2016 until 2017, in 50 traps deployed in vineyards between V, VI, VII, and metropolitan region in Chile.

The classifier algorithm developed for *L. botrana* recognition included a preprocessing step using a median filter and CLAHE equalization and then a segmentation step using K-means clustering and the generation of two descriptors; one corresponds to the gray scale segment histogram and the other obtained calculating the gradient modulus histogram. A dataset of 360 images was used for classifier construction.

The K-means segmentation algorithm was evaluated in comparison with the sliding window approach. The segmentation algorithm was superior to sliding window in quality of the recognition for the rescaled images at 1, 0.75, 0.5, and 0.25. The segmentation algorithm yielded an average specificity of 95.1.

A Spark distributed version of the system was developed. This system allows fast and scalable cloud-computing analysis of the images, providing an ideal environment for on-field applications. The scalability of the distributed K-means segmentation algorithm was evaluated. The curves show they continue delivering speed-up to 16 executors. When over 16 executors the performance is clearly sublineal. For the dataset with 360 images the deviation from the perfect result on average was 20.6%, for the case of 3600 images 25.6%, and for the case of 36000 images 32.1%. The one that had the best performance in all cases was the one with the lowest resolution, but the effect of the resolution was small.

Using the additional information of timestamps and GPS positioning captured by the implemented application, future work is intended to generate models that allow determining areas of risk of *Lobesia botrana* in addition to conditions that favor the development of the plague. From the point of view of computer science it is interesting to work in a computational intelligence algorithm that allows determining the best parameters setting. This automatic determination of parameters would allow the algorithm to be easily adaptable to similar recognition problems.

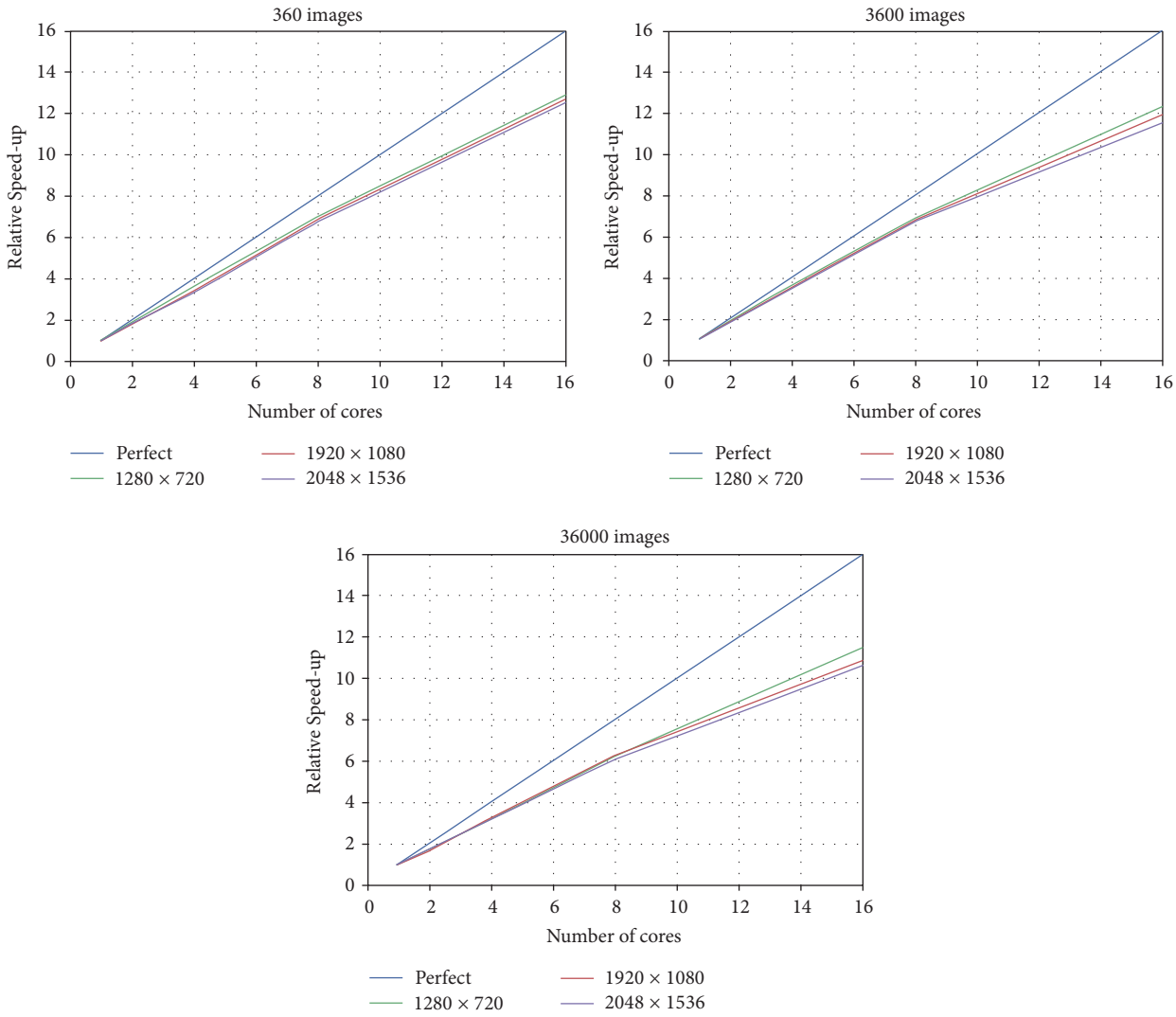


Figure 16: Relative speed-up experiments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Moreau, C. Villemant, B. Benrey, and D. Thiéry, "Species diversity of larval parasitoids of the European grapevine moth (Lobesia botrana, Lepidoptera: Tortricidae): The influence of region and cultivar," *Biological Control*, vol. 54, no. 3, pp. 300–306, 2010.

[2] G. Cozzi, M. Pascale, G. Perrone, A. Visconti, and A. Logrieco, "Effect of Lobesia botrana damages on black aspergilli rot and ochratoxin A content in grapes," *International Journal of Food Microbiology*, vol. 111, pp. S88–S92, 2006.

[3] M. Fermaud and Z. Giboulot, "Influence of lobesia botrana larvae on field severity of botrytis rot of grape berries," *Plant Disease*, vol. 76, no. 4, pp. 404–409, 1992.

[4] C. Ioriatti, A. Lucchi, and L. G. Varela, "Grape berry moths in Western European Vineyards and their recent movement into the new world," in *Arthropod Management in Vineyards*, pp. 339–359, Springer, Dordrecht, The Netherlands, 2012.

[5] C. Ioriatti, G. Anfora, M. Tasin, A. De Cristofaro, P. Witzgall, and A. Lucchi, "Chemical ecology and management of lobesia botrana (Lepidoptera: Tortricidae)," *Journal of Economic Entomology*, vol. 104, no. 4, pp. 1125–1137, 2011.

[6] Servicio Agricola y Ganadero (SAG), Programa nacional de lobesia botrana. Estrategia 2016-2017, 2016.

[7] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: object localization by efficient subwindow search," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, IEEE, June 2008.

[8] S. Ray and R. H. Turi, "Determination of number of clusters in k-means clustering and application in colour image segmentation," in *Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques*, pp. 137–143, 1999.

[9] J. F. S. Gomes and F. R. Leta, "Applications of computer vision techniques in the agriculture and food industry: A review,"

European Food Research and Technology, vol. 235, no. 6, pp. 989–1000, 2012.

[10] D.-W. Sun, *Computer Vision Technology for Food Quality Evaluation*, Academic Press, 2016.

[11] S. Benalia, S. Cubero, J. M. Prats-Montalbán, B. Bernardi, G. Zimbalatti, and J. Blasco, "Computer vision for automatic quality inspection of dried figs (Ficus carica L.) in real-time," *Computers and Electronics in Agriculture*, vol. 120, pp. 17–25, 2016.

[12] S. R. Debats, D. Luo, L. D. Estes, T. J. Fuchs, and K. K. Caylor, "A generalized computer vision approach to mapping crop fields in heterogeneous agricultural landscapes," *Remote Sensing of Environment*, vol. 179, pp. 210–221, 2016.

[13] J. Ma, D.-W. Sun, J.-H. Qu et al., "Applications of computer vision for assessing quality of agri-food products: a review of recent research advances," *Critical Reviews in Food Science and Nutrition*, vol. 56, no. 1, pp. 113–127, 2016.

[14] M. Nagle, I. Kiatkamjon, G. Romano, B. Mahayothee, V. Sardsud, and J. Müller, "Determination of surface color of "all yellow" mango cultivars using computer vision," *International Journal of Agricultural and Biological Engineering*, vol. 9, no. 1, article 42, 2016.

[15] G. P. Moreda, J. Ortiz-Cañavate, F. J. García-Ramos, and M. Ruiz-Altisent, "Non-destructive technologies for fruit and vegetable size determination—a review," *Journal of Food Engineering*, vol. 92, no. 2, pp. 119–136, 2009.

[16] G. P. Moreda, M. A. Muñoz, M. Ruiz-Altisent, and A. Perdigones, "Shape determination of horticultural produce using two-dimensional computer vision - A review," *Journal of Food Engineering*, vol. 108, no. 2, pp. 245–261, 2012.

[17] A. Mizushima and R. Lu, "An image segmentation method for apple sorting and grading using support vector machine and Otsu's method," *Computers and Electronics in Agriculture*, vol. 94, pp. 29–37, 2013.

[18] X. P. Burgos-Artizzu, A. Ribeiro, A. Tellaeche, G. Pajares, and C. Fernández-Quintanilla, "Improving weed pressure assessment using digital images from an experience-based reasoning approach," *Computers and Electronics in Agriculture*, vol. 65, no. 2, pp. 176–185, 2009.

[19] J. D. Luscier, W. L. Thompson, J. M. Wilson, B. E. Gorham, and L. D. Dragut, "Using digital photographs and object-based image analysis to estimate percent ground cover in vegetation plots," *Frontiers in Ecology and the Environment*, vol. 4, no. 8, pp. 408–413, 2006.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[22] LISA Lab University of Montreal, *Deep Learning Tutorial*, Release 0.1, 2013.

[23] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *Proceedings of the 11th International Conference on Document Analysis and Recognition*, pp. 1135–1139, September 2011.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV 2014*, vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, 2014.

[25] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS '14)*, pp. 2627–2635, December 2014.

[26] C. Wen, D. Wu, H. Hu, and W. Pan, "Pose estimation-dependent identification method for field moth images using deep learning architecture," *Biosystems Engineering*, vol. 136, pp. 117–128, 2015.

[27] J. L. Miranda, B. D. Gerardo, and B. T. Tanguilig III, "Pest detection and extraction using image processing techniques," *International Journal of Computer and Communication Engineering*, vol. 3, no. 3, pp. 189–192, 2014.

[28] W. Ding and G. Taylor, "Automatic moth detection from trap images for pest management," *Computers and Electronics in Agriculture*, vol. 123, pp. 17–28, 2016.

[29] F. Zhang, G. Wang, and J. Ye, "An automatic identification system of agricultural pests based on machine vision," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 12, pp. 6231–6236, 2015.

[30] Y. Kaya and L. Kayci, "Application of artificial neural network for automatic detection of butterfly species using color and texture features," *Visual Computer*, vol. 30, no. 1, pp. 71–79, 2014.

[31] F. C. Monteiro and A. Campilho, "Watershed framework to region-based image segmentation," in *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*, pp. 1–4, IEEE, December 2008.

[32] R. V. Patil and K. C. Jondhale, "Edge based technique to estimate number of clusters in k-means color image segmentation," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 2, pp. 117–121, IEEE, July 2010.

[33] W. Cui and Y. Zhang, "Graph based multispectral high resolution image segmentation," in *Proceedings of the International Conference on Multimedia Technology (ICMT '10)*, pp. 1–5, IEEE, October 2010.

[34] A. S. Abutaleb, "Automatic thresholding of gray-level pictures using two-dimensional entropy," *Computer Vision, Graphics and Image Processing*, vol. 47, no. 1, pp. 22–32, 1989.

[35] P.-Y. Yin and L.-H. Chen, "A new method for multilevel thresholding using symmetry and duality of the histogram," in *Proceedings of the International Symposium on Speech, Image Processing and Neural Networks (ISSIPNN '94)*, pp. 45–48, IEEE, Hong Kong, 1994.

[36] A. Xu, L. Wang, S. Feng, and Y. Qu, "Threshold-based level set method of image segmentation," in *Proceedings of the 3rd International Conference on Intelligent Networks and Intelligent Systems (ICINIS '10)*, pp. 703–706, IEEE, November 2010.

[37] W. Kaihua and B. Tao, "Optimal threshold image segmentation method based on genetic algorithm in wheel set online measurement," in *Proceedings of the 3rd International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 20)*, vol. 2, pp. 799–802, IEEE, January 2011.

[38] F. Zhang, S. Guo, and X. Qian, "Segmentation for finger vein image based on PDEs denoising," in *Proceedings of the 3rd International Conference on BioMedical Engineering and Informatics (BMEI '10)*, vol. 2, pp. 531–535, IEEE, October 2010.

[39] C. Yuan and S. Liang, "Segmentation of color image based on partial differential equations," in *Proceedings of the 4th International Symposium on Computational Intelligence and Design (ISCID '11)*, vol. 2, pp. 238–240, IEEE, October 2011.

[40] J. Xiao, B. Yi, L. Xu, and H. Xie, "An image segmentation algorithm based on level set using discontinue pde," in *Proceedings*

of the 1st International Conference on Intelligent Networks and Intelligent Systems (ICINIS '08), pp. 503–506, IEEE, November 2008.

[41] C. Lynch, "Big data: how do your data grow?" *Nature*, vol. 455, no. 7209, pp. 28-29, 2008.

[42] M. Minelli, M. Chambers, and A. Dhiraj, *Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses*, John Wiley & Sons, 2012.

[43] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[44] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality in mapreduce," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '12)*, pp. 419–426, IEEE, May 2012.

[45] K. Grolinger, M. Hayes, W. A. Higashino, A. L'Heureux, D. S. Allison, and M. A. Capretz, "Challenges for mapreduce in big data," in *Proceedings of the IEEE World Congress on Services (SERVICES '14)*, pp. 182–189, IEEE, June 2014.

[46] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud '10)*, p. 10, June 2010.

[47] R. Shyam, S. Kumar, P. Poornachandran, and K. P. Soman, "Apache spark a big data analytics platform for smart grid," *Procedia Technology*, vol. 21, pp. 171–178, 2015.

[48] S. Sarraf and M. Ostadhashem, "Big data application in functional magnetic resonance imaging using apache spark," in *Proceedings of the Future Technologies Conference (FTC '16)*, pp. 281–284, IEEE, December 2016.

[49] W. Zhou, R. Li, S. Yuan et al., "MetaSpark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes," *Bioinformatics*, vol. 33, no. 7, pp. 1090–1092, 2017.

[50] J. García, B. Crawford, R. Soto, and P. García, "A multi dynamic binary black hole algorithm applied to set covering problem," in *Proceedings of the International Conference on Harmony Search Algorithm*, vol. 514, pp. 42–51, Springer, 2017.

[51] Z. Zhang, K. Barbary, F. A. Nothaft et al., "Kira: processing astronomy imagery using big data technology," *IEEE Transactions on Big Data*, 2016.

[52] L. Hao, S. Jiang, B. Si, and B. Bai, "Design of the research platform for medical information analysis and data mining," in *Proceedings of the 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI '16)*, pp. 1985–1989, IEEE, Datong, China, October 2016.

[53] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*, O'Reilly Media, Inc., 2015.

[54] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 530–533, ACM, March 2011.

[55] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012.

[56] R. Boyle and R. Thomas, *Computer Vision: A First Course*, Blackwell Scientific Publications, 1988.