Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 695910, 17 pages
http://dx.doi.org/10.1155/2014/695910

Research Article

Image Classification Using PSO-SVM and an RGB-D Sensor

Carlos López-Franco, Luis Villavicencio, Nancy Arana-Daniel, and Alma Y. Alanis

Computer Science Department, CUCEI, University of Guadalajara, 44430 Guadalajara, JAL, Mexico

Correspondence should be addressed to Carlos López-Franco; clzfranco@gmail.com

Received 10 February 2014; Revised 29 May 2014; Accepted 13 June 2014; Published 10 July

Academic Editor: Francesco Ubertini

Image classification is a process that depends on the descriptor used to represent an object. To create such descriptors we use object models with rich information of the distribution of points. The object model stage is improved with an optimization process by spreading the point that conforms the mesh. In this paper, particle swarm optimization (PSO) is used to improve the model generation, while for the classification problem a support vector machine (SVM) is used. In order to measure the performance of the proposed method a group of objects from a public RGB-D object data set has been used. Experimental results show that our approach improves the distribution on the feature space of the model, which allows to reduce the number of support vectors obtained in the training process.

1. Introduction

Over the past years, there has been an increasing interest in object recognition. Object recognition can be divided in two major tasks: object localization and image classification. *Object localization* detects instances of a given category in the image. *Image classification* can be defined as the task of defining labels to an image, depending on the presence of an object.

In this paper we propose an image classification system based on invariant moment descriptor that includes depth information. The 3D data allows producing small and robust descriptors that will improve the image classification. These descriptors are constructed using object models with rich information of the distribution of points. The model generation stage requires that best points are selected; therefore this stage can be defined as an optimization problem.

Mathematical optimization is the selection of the best element with regard to some criteria. In the simplest case, it is consisted of maximizing or minimizing a fitness function [1]. Metaheuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution [2]. Many metaheuristics implement nature-inspired stochastic optimization. One of these algorithms is particle swarm optimization (PSO) developed by [3] and inspired by social behavior of bird flocking or fish schooling. It has been applied in many fields such as tremor analysis for biomedical engineering, trajectory planning [4], electric power [5], and image processing [6]. Optimization algorithms are often used in computer vision tasks such as image classification, which finds a relation between an input image and a set of previously known models [7].

The sensor used in this work is a Kinect [8] which is an RGB-D sensor providing synchronized color and depth images. This sensor is widely used by the computer vision community due to its capabilities.

In this work we analyze the inclusion of 3D information (provided by an RGB-D sensor) and the use of the PSO algorithm to create robust object models. Using this approach we can construct small and robust descriptors that improve image classification.

The rest of the paper is organized as follows. The next section will present the proposed image classification system. In Section 4 we present the mesh optimization process, with a brief introduction to the PSO algorithm. In Section 5 the invariant moment descriptor and classification are explained. In Section 6 we show the results of the proposed approach. Finally, in Section 8 we give the conclusions.

2. Related Work

In recent years image classification has become a very active research field. The goal of the classification task is to develop an algorithm which will assign a class-label to each one of the samples in the training data set. There are two main approaches for classification in the literature, namely, (a) supervised approach and (b) unsupervised approach, where the former uses a set of samples to train the classification, and the latter performs the classification by exploring the data [9]. Many popular image classification methods [10, 11] are based on local descriptor and support vector machines as basic techniques.

One of the most widely used detectors is the Harris Corner detector [12]. However the Harris detector is not invariant to affine transformations and scale [13]. In [14], an automatic scale detection was proposed. Later in [15], the authors present an algorithm for interest point detection which is invariant to important scale changes. Then in [16], the SIFT detector is proposed; the approach approximates the Laplacian of Gaussians (LoG) by difference of Gaussians (DoG) filter to identify the potential keypoints. Later in [17] the SURF detector was presented, and SURF uses an approach similar to SIFT; however the approach uses integral images instead of DoGs; this allows a fast computation of approximate LoG using a box filter.

Many different feature descriptors have been proposed in the literature: steerable filters [18], moment invariants [19], phase-based local features [20], Gaussian derivatives [21], and descriptors that represent the distribution of smaller-scale features [10]. The SIFT detector produces robust descriptors; however the speed of the detector is not very high. In [17], the SURF detector was presented; this approach is faster than SIFT and it has a similar performance on real images [22].

Histogram of oriented gradients (HoG) was proposed in [23]. The HoG local patters are based on gradient histogram. The HoG and SIFT detectors share the same concept of implementation in the sense that local histograms of oriented patches defined across the image are used to produce the descriptor. In [23] the authors used HoG descriptors as feature vectors for a linear SVM. In [24] the authors propose the use of HoG for object detection and show that HoG descriptors have robust and effective features for such application. In [25] the authors build an efficient detector using AdaBoost to train a chain of progressively more complex region rejection rules based on Haar descriptors for pedestrian detection.

Our Approach. In the last decades, SVMs have been proven to be an effective classifier approach for small sample sets or high dimensional problems [26, 27]. Such result is very important since both problems are too difficult to be solved by classical paradigms.

One of the problems encountered during the design of an image classification technique is data overfitting, which arises when the training samples are small in comparison with the number of features. To overcome this problem we design a descriptor with an optimized 3D point distribution.

The contribution of this work is the development of a small and robust descriptor based on invariant moments and 3D information in order to improve the classification process. The 3D information is incorporated from depth data obtained from an RGB-D sensor. The object model is optimized through the use of a PSO algorithm; this optimization allows improving the classification.

3. Image Classification System

The proposed classification system uses 3D information and is based on local features, invariant moments, contour generation, and a reduction of depth information using a mesh grid. It consists of five main steps: feature extraction, contour creation, mesh reduction, mesh optimization, and invariant moment descriptor formation; see Figure 1.

3.1. Feature Extraction. The first step is the extraction of SURF keypoints from the input image (Figure 5(b)). The speeded up robust features algorithm (SURF) [28] finds features using integral images and Haar-like filters. SURF features provide robustness and speed and are able to be detected despite scale, translation, and rotation of objects or changes in illumination. The following steps rely on this stability to group points and create object models, and they are described in the following subsections.

SURF and SIFT detectors employ slightly different ways of detecting features. SIFT detector builds image pyramids and filters each layer with a Gaussian of increasing sigma values and then it takes the difference. SURF detector uses a box filter approximation. A comparison of the SIFT and SURF detectors is presented in [22]. In such work, the authors conclude that *SURF is as good as SIFT on most tests, except for scaling, large blur, and viewpoint*. With respect to this, we have to mention that the proposed descriptor only uses SURF detector in the first step of the classifier to detect interest points. Then the proposed descriptor is constructed using additional 3D information and invariant moments. In the experiments results show that the proposed descriptor is small and robust. In [22], the authors also mention that *on real image data sets there is little to separate the different SIFTs and SURF except for efficiency*. In this work we choose SURF detector because we are interested in implementing a real time classifier algorithm that could be used by a mobile robot. In addition, as we will see in the results' section, the proposed descriptor is improved thanks to the spreading step in conjunction with the SURF detector, and this provides good results that are comparable and overcome similar approaches.

3.2. Contour Creation. The RGB-D sensor provides a huge amount of data; thus the information must be reduced into a contour-mesh model to decrease computational cost. To construct the contour-mesh we use keypoints. First, they are scaled and translated; then, we compute the magnitude of the keypoints with respect to their centroid $(\overline{x}, \overline{y})$ as

$$m(x, y) = \sqrt{(x - \overline{x})^2 + (y - \overline{y})^2} \quad (1)$$

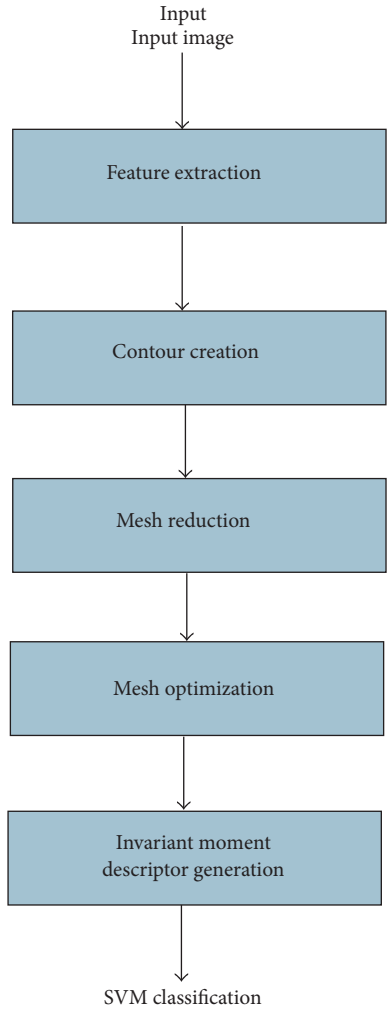

Figure 1: Image classification system.

and its orientation with

$$\theta(x, y) = \tan^{-1}\left(\frac{y - \overline{y}}{x - \overline{x}}\right). \quad (2)$$

After that, we divide the 360 degrees into orientation bins and use a sliding window to take one point for every bin according to their magnitude. If any of the bins remains empty, then its value is linearly interpolated using the previous and next known points in the contour. Examples for this stage of the process are shown in Figures 2 and 3.

In the next sections we will explain the last two steps of the proposed approach, namely, the mesh reduction step and the construction of the descriptor.

3.3. Mesh Reduction. In this step, the 3D data that belongs to the object contour is segmented, and then a cloud of 3D points is obtained. Due to the large number of points in the cloud, a reduction of the data will be required in order to keep a low computational cost. Therefore, in this step a mesh that covers the 3D point cloud is constructed, which allows to reduce the number of 3D points in the cloud, without losing important information.

First, we proceed to extract depth information contained within the boundaries of the contour and reduce the points that will be taken to compute the moments. The reduction of points aims to generate a smaller set with rich information by adjusting a mesh grid over the object. The initial position of the points is obtained sectioning the bounding box, into equally separated cells, generating (x, y) coordinates. This creates a set of bidimensional points, that is,

$$\text{Grid} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, \quad (3)$$

where n is a predefined constant. Besides,

$$x_i \in [x_{\min}, x_{\max}], \qquad y_i \in [y_{\min}, y_{\max}],$$
$$x_1 = x_{\min}, \qquad y_1 = y_{\min},$$
$$x_n = x_{\max}, \qquad y_n = y_{\max},$$

Figure 2: Totem object, with SURF keypoints.

$$x_{i+1} = x_i + x_{\text{inc}}, \qquad y_{i+1} = y_i + y_{\text{inc}},$$
$$x_{\text{inc}} = \frac{x_{\max} - x_{\min}}{n}, \qquad y_{\text{inc}} = \frac{y_{\max} - y_{\min}}{n} \tag{4}$$

minimum and maximum values are defined by the bounding box. The remaining points are tested using the point in polygon (PiP) algorithm, if the points are inside the polygon then they are considered valid.

Then, for each invalid point, we take at random two valid points and move the outlier to a position between them,; this can be seen as a biased migration. Later, we attach the z coordinate (depth) to the (x, y) points. Examples for this stage of the process are shown in Figures 4 and 5(d).

It is important to mention that a point will be considered valid if and only if the coordinates (x, y) of the point are inside the contour, and the depth is not zero.

These steps generate an object model for which we can extract information that we can use for classification; however the simple migration of points produces a model in which points are not equally distributed. This problem is solved by applying evolutionary computation.

4. Mesh Optimization

The mesh reduction step produces an object model; however the points are not equally distributed and thus an optimization step is required. For the optimization step, we have chosen an evolutionary computation (EC) technique, due to all the constraints of the mesh optimization problem.

Evolutionary algorithms (EA) are stochastic search methods inspired by the behavior of swarms' populations or natural biological behavior. In general there are five popular algorithms: genetic algorithms [29], memetic algorithms [30], particle swarm optimization [3], ant-colony optimization [31], and shuffled frog leaping algorithm [32]. In [33], the authors present a comparison study of these five EA; they conclude that the PSO algorithm performs better in general, with respect to the quality of the solution and the success rate. For these reasons, we decided to choose the PSO algorithm among the others.

To solve the mesh optimization problem, the PSO algorithm (Algorithm 1) was adapted in order to spread the set of (x, y) points that conform the mesh and to obtain better models with rich information about the distribution of the points. The purpose is to maximize the distance between points while maintaining them inside the boundaries conformed by the object contour.

In our approach, each PSO particle represents a point in the mesh. The problem has multiple boundaries; every point of the contour is one. Thus, we have to check if particles lie inside the polygon and clamp them after every update. Instead of gathering particles to a global best position they take positions separated uniformly from each other. This is obtained through the fitness function and a modification in the updating rules.

4.1. Description of the Method. The objective of this method is the construction of a mesh with the best distribution of points. The first step in the mesh construction is the determination of the object contour (Section 3.2). This step uses features detected with SURF and a sampling technique to find the object contour. The second step is the mesh reduction step (Section 3.3); in this step the 3D data is used to enhance the object model. However, if we use all the 3D data that belong to the obtained contour, then the computational cost, required to process it, will be high. Therefore, the objective of this step is the construction of a mesh that covers the 3D point cloud, but with fewer points. The third step is the mesh optimization; this process is required since the mesh constructed in the previous step does not distribute efficiently. For this purpose, the PSO algorithm was adapted to spread the points over the object. The adapted PSO minimizes the distance of each particle and its neighbors with respect to the mean distance of all the particles with respect to its neighbors. The PSO particles are initialized with the coordinates of the points. The number of particles is therefore equal to the number of points. Finally, the object model is recovered from the best local particle value (P_i).

4.2. Particle Swarm Optimization Algorithm. PSO is a stochastic search method inspired from the behavior of swarm animals, like bird flocking and fish schooling. In PSO, particles, or solution candidates, move over the search space towards the zone with the best conditions using a cognitive component of the relative particle and a social component generated by the swarm (best local and global positions). This lets PSO to evolve social behavior and relative movement into global optimum solutions [34, 35].

In the iterative process, the position X_i and velocity V_i of particles are updated according to the cognition component P_i and the social component G with

$$V_i(t+1) = \omega V_i(t) + c_1\varphi_1\left(P_i - X_i\right) + c_2\varphi_2\left(G - X_i\right),$$
$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{5}$$

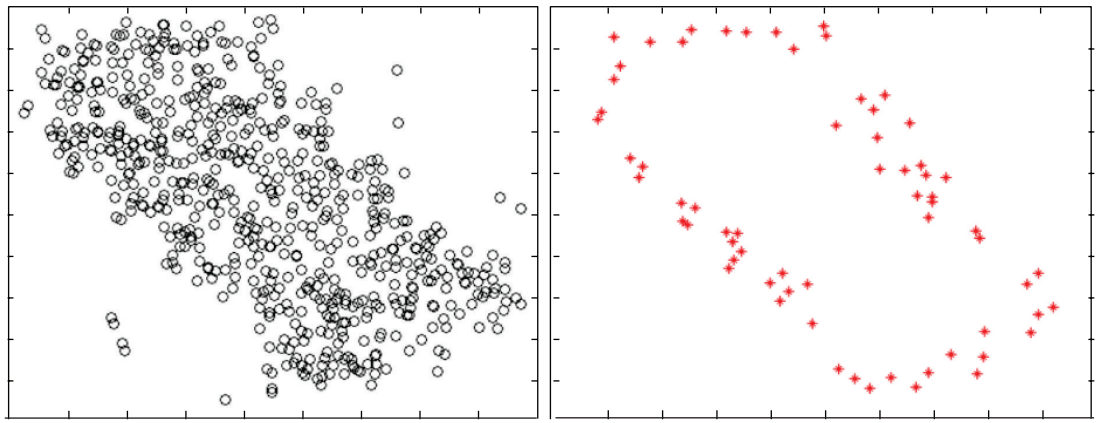

Figure 3: Sample contour for the totem object.

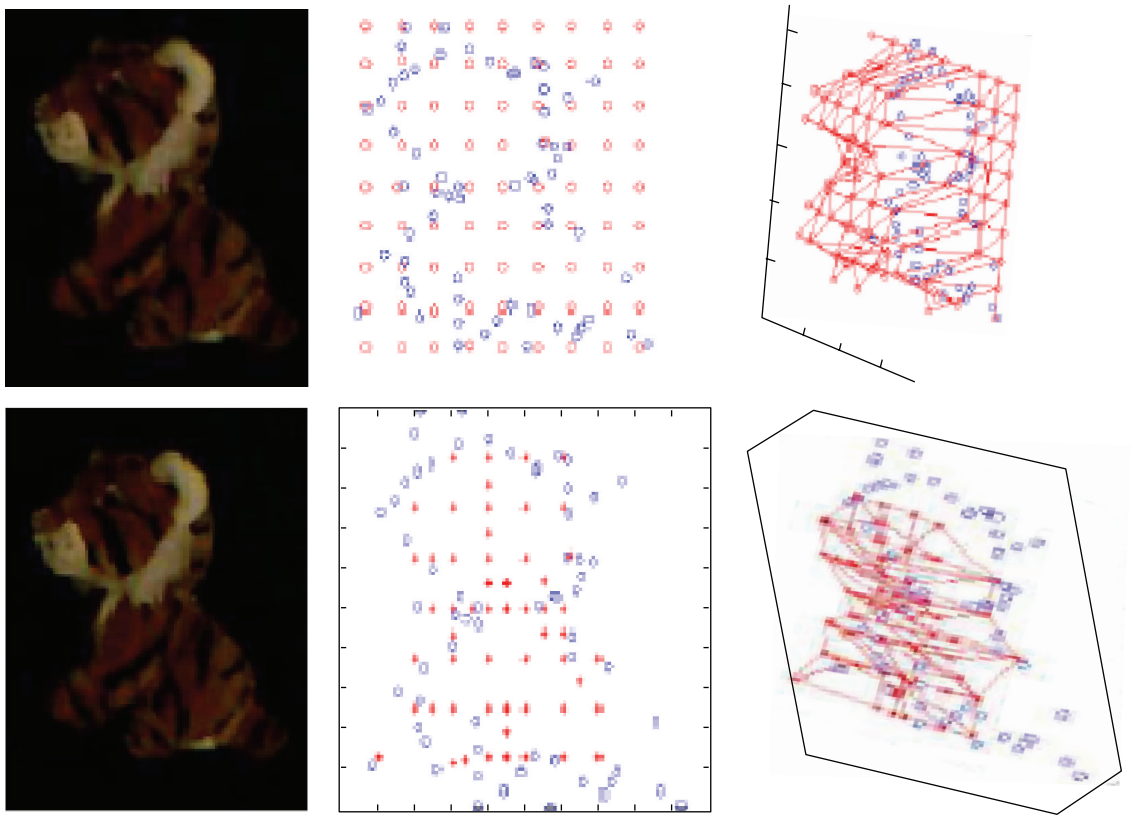

Figure 4: Mesh creation example.

where c_1, c_2 are positive constants, φ_1, φ_2 are two random variables with uniform distribution between 0 and 1, and ω is the inertia weight which balances the effect of the previous velocity vector on the new one. The cognitive component, P_i, is updated by each particle when a better position is obtained. The social component, G, is updated when a new best position within the whole swarm is found. After initializing the swarm, in each iteration, the PSO basic steps are performed until the stop criterion is reached [36].

For more details on PSO, the interested reader is referred to view [2, 3, 36, 37].

4.3. PSO Fitness Function. Instead of a single fitness function we undertake three steps to get a fitness value. First, we measure the distance of each particle to its nearest neighbor; this measure gives information about how separated is every particle. We only take the distance between the current i-particle and a neighbor j-particle with

$$D_i = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}, \tag{6}$$

to keep a low computational load.

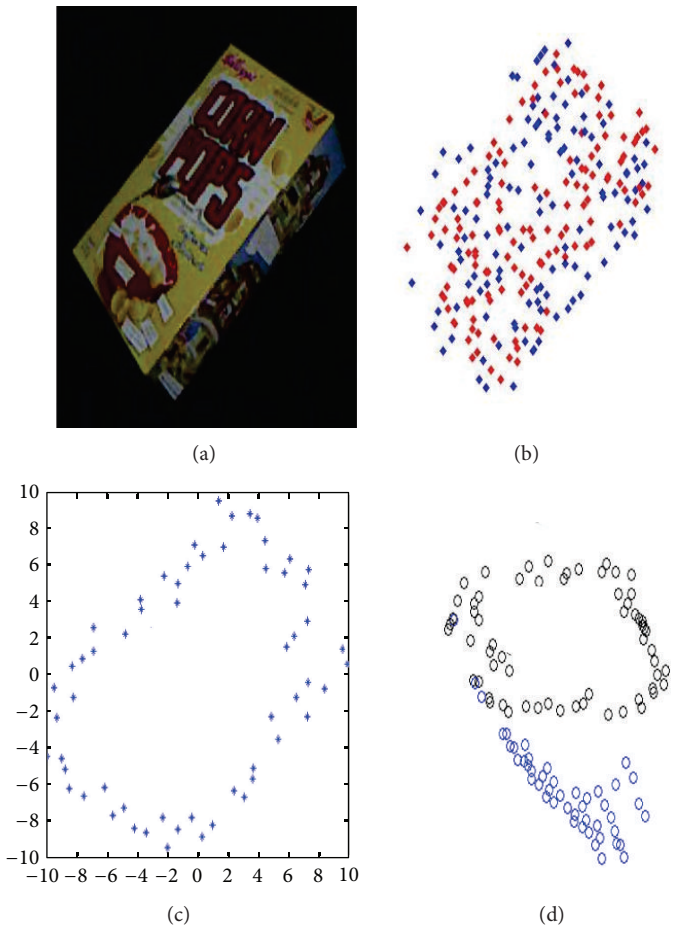

Figure 5: An image through the different stages of the procedure. (a) Original input image. (b) Feature extraction. (c) Contour created for the object. (d) Depth information attached in a mesh grid.

(1) **repeat**
(2) Calculate fitness value of each particle using the fitness function
(3) Update local best if the current fitness value is better
(4) Determine global best: take the best fitness particle and compare it to current best
(5) For each particle
(6) Calculate particle velocity according to (1)
(7) Update particle position according to (2)
(8) **until** Stop criteria is met

Algorithm 1: PSO algorithm.

Then we calculate the mean of distances

$$\overline{D} = \frac{1}{N}\sum_{k=1}^{N} D_k, \tag{7}$$

where N is equal to the swarm size. This can be seen as the global value to maximize, since we want the points to be uniformly separated and yet lie inside the boundaries; that is, they have the same distance to each other and cover the space uniformly.

Next, we compute the difference between the local distance of the particle to its neighbor and the mean global distance. Thus, the fitness of ith particle is defined as

$$f_i = \sqrt{\left(D_i - \overline{D}\right)^2}. \tag{8}$$

This is the fitness value, by minimizing (8) every particle tries to minimize its own distance with respect to the mean distance of the swarm and by doing this we would obtain an approximation to a geometric uniform distribution of the swarm on the space; that is, we will obtain a swarm in which

every particle is separated from its neighbors by a distance of $\overline{D}$. Thus, the distance of the particle to its neighbor is the closest value to the mean estimated distance. This value is stored in by the cognitive component, P_i.

Since our function is multiconstrained, we add a penalty term to the fitness function. Particles that go out of the boundaries are penalized according to the distance that they have to the centroid. We add a constant α which determines the influence of the penalization in the fitness value computation [36]; the fitness function with the penalization term is defined as

$$f_i = \sqrt{\left(D_i - \overline{D}\right)^2} + \alpha T(X) \left|X - \overline{X}\right|, \tag{9}$$

with

$$T(X) = \begin{cases} 1 & \text{Particle is out side contour} \\ 0 & \text{Particle is inside contour,} \end{cases} \tag{10}$$

where X is a point (x, y) in the mesh represented by a particle, $\overline{X}$ is the centroid $(\overline{x}, \overline{y})$, and $T(X)$ is a point-in-polygon function added so only points that got outside the contour are penalized.

Additional to the penalization term, we also use a preserve feasibility approach as explained by [36], since all our initialization particles are feasible we want to preserve the final result like that; thus P_i is only updated when the particle lies inside the contour.

4.4. PSO Update Formulas. The position update includes a variable β that multiplies velocity:

$$X_i(t+1) = X_i(t) + \beta V_i(t+1). \tag{11}$$

β is a term that defines how close are particles from boundaries; we want particles to be sufficiently spread, but we also want some particles close to the boundaries. Clamping and updates may cause particles to go out or be repositioned. If the particle is close to boundaries, β will be small and the update effect will be less. This value is thresholded so after certain value it becomes one and the effect of velocity applies normally; this way, particles far from boundaries are not affected. In conclusion, β can be seen as a function of the distance of the particle to the boundaries with

$$D_b = \sqrt{\left(x_i - x_{\text{boundary}}\right)^2 + \left(y_i - y_{\text{boundary}}\right)^2} \tag{12}$$

the distance of a point to the object boundary, and

$$\beta = \begin{cases} D_b & D_b < \text{Threshold} \\ 1 & \text{Otherwise.} \end{cases} \tag{13}$$

The threshold value depends on the range of the data set; it establishes how far to the border the data can be without being affected. In our case feature points are scaled to $[-10, 10]$, and β values in the range $[0.2, 1.0]$ were tested, resulting in 0.7 as the value that best performed.

In the velocity update, the global coefficient has the effect on particles to move towards the best position of the swarm. Since in this application we do not need that effect, the term is replaced by one that makes particles get closer or farther from their neighbor as needed. This behavior is accomplished by using the sign of the distance between particles, and a constant value ρ to determine how fast particles are going to move towards or away from others; that is,

$$V_i(t+1) = \omega V_i(t) + c_1 \varphi_1 \left(P_i - x_i\right) + \rho \operatorname{sign}\left(X_i - X_j\right), \tag{14}$$

where X_i, X_j are two particles (x_i, y_i), (x_j, y_j), with X_j being the closest particle to X_i. Another option is the inclusion of a scale factor of the current fitness value, since it gives a value of how much the particles are separated.

4.5. Details of the Modified PSO. The fitness function of the algorithm is defined to minimize the distance between the particle i, with respect to the mean distance of all the particles with respect to their corresponding neighbors; therefore at the end of the PSO iterations we obtain the best spread particles found by the algorithm. Although many different fitness functions can be defined, for this particular application the chosen fitness function shows a good performance, as it was demonstrated on the experimental results.

With respect to the stop criteria of the algorithm, we cannot force PSO to stop when only the best value G is close to zero, since it does not guarantee that the distances between each particle with respect to their neighbors are close to the mean distance. Instead, the algorithm stops when the particle best value P_i of each particle is close to zero, or in practice smaller than a certain threshold. In addition we can stop the algorithm if a certain threshold of iterations has been met.

4.6. Point Spreading Algorithm (PSA). The inclusion of PSO in the mesh reduction step of the image classification system aims to spread points and create models that describe the entire object surface better. After the initial generation of points in a grid over the object and the migration of points that lie outside the object, the PSO variation is applied to the mesh of points. Particles are initialized with the coordinate values of points. The number of particles is therefore determined by the number of points that we want over the object. A maximum number of iterations is set and the algorithm is executed, until the stop criterion is reached. An object model is recovered from the local best position (P_i) that each particle generated. The main steps of the algorithm are portrayed in Algorithm 2.

In Figures 6 and 7 we can see an example of this procedure. The first picture shows the original image. The second image shows the initial coordinates of the mesh. The third image shows migration of points; some points are migrated to very close places and thus they are covering the object poorly. The fourth image shows the points after the PSO algorithm is applied; we can see that points are moved and the object surface is covered in a better way.

(1) Obtain object contour
(2) Generate starting point coordinates from bounding box
(3) Migrate non-valid points
(4) PSO spreading

Algorithm 2: Mesh construction algorithm.

5. Invariant Moment Descriptor and Classification

Moments provide useful and compact information of a data set, such as its spread or dispersion. A pattern may be represented by a density distribution function, moments can be obtained for a set of points representing an object, and they can then be used to discriminate between objects [38]. The first order moments can be used to locate the centroid of the points' distribution. If we compute the moments considering a translation to the centroid, we generate central moments which can be made scale invariant [39, 40]. The general equation for three-dimensional central moment (for short, 3D moment) is defined as

$$\mu_{pqr} = \sum_x \sum_y \sum_z (x - \overline{x})^p (y - \overline{y})^q (z - \overline{z})^r f(x, y, z), \quad (15)$$

where $f(x, y, z)$ is a distribution function of the variables and $(\overline{x}, \overline{y}, \overline{z})$ is the centroid. The scaling is performed using

$$\eta_{pqr} = \frac{\mu_{pqr}}{\mu_{000}^{\gamma}}, \quad \gamma = \left[\frac{p + q + r}{3}\right] + 1. \quad (16)$$

In particular, [38] defines seven values, computed by normalizing central moments through order three that are invariant to object scale, position, and orientation. Tests were performed using different combinations of these seven values and then using the moments of order one to four. Similar results were obtained using the three first values defined by Hu and the moments of order one to four. Therefore, invariant moments can be computed from the reduced 3D set. In the proposed method we use the first four moments, over the three dimensions, then the values for p, q, and r are defined over the interval from 1 to 4, with these moments we define a descriptor of 12 elements to represent each object model. Such vector has the form

$$d = [\eta_{100}, \eta_{200}, \eta_{300}, \eta_{400}, \eta_{010}, \eta_{020}, \eta_{030}, \eta_{040}, \eta_{001}, \eta_{002}, \eta_{003}, \eta_{004}]. \quad (17)$$

Finally, the descriptor is given as input to a SVM and we get the result on whether the image contains the target object. The SVM [41] is an algorithm to solve classification and regression problems. It defines a subset on the train data composed by those samples that are closer to the decision area. It is based in the maximization of the margin of separation between classes. The SVM algorithm is able to perform nonlinear classification thanks to the use of kernel functions.

6. Results

In the following, we validate the proposed approach and compare it with histograms of oriented gradients (HOG) [42], scale invariant feature transform (SIFT) detector [10], and a detection system using cascades of HAAR-like features [43].

First, a data set composed by 5 objects was defined (cups, hair dryers, irons, cereal boxes, and soda cans) with around 50 images for each object; see Figure 8. The images included changes in the scene conditions such as illumination, object orientation, and position, besides partially occluded objects.

In addition to the house made data set, the RGB-D object data set from the University of Washington [44] was used to perform tests. We used a set consisting of eight different objects (bowl, cap, flashlight, coffee mug, cereal box, soda can, camera, and pitcher) similar to [45] (Figure 9). This data set was selected since it contains similar objects to our set and they were acquired through the same range sensor. For each of the classes, the set includes a group of around four objects that belong to the same class with three different views and 200 images for each view. The bounding box, cropped image, and object mask are provided.

6.1. Algorithm Parameters. We worked with images containing a single object with a discriminative background. As explained before, we start by extracting key points from the image. The contours were formed by 72 bins of 5 degrees each. The mesh was composed by a 9×9 grid, generating 81 points. The algorithmic control parameters of PSO, coefficient of cognition (c_1), and inertia weight (ω) (14) were set to $\omega = 0.729844$ and $c_1 = 1.49618$ as suggested in [46]. Different values for the parameters α (9) and ρ (14) were tested; the best results were obtained with $\alpha = 1.230$ and $\rho = 3.751$, though this might be application dependent.

In our case, for ρ, values greater than 5 resulted too aggressive bringing neighbor particles too close rapidly, and values smaller than 2 had little effect on the movement of particles. Values for α within the range $[0.5, 1.5]$ caused the expected penalty result while other values caused fitness values to boost or to be decreased abnormally. Finally, as explained before β (11) was set to 0.7 defining that particles closer to the boundaries will be less affected by the velocity update.

6.2. Training. The cross-validation method was used to validate the training process [47]. Using a 4-fold approach in the five-object set and a 5-fold approach in the six-object set, the data was randomly separated in different subsets with equal number of elements; three of the subsets were used to train different SVMs and the rest were used for validation. The classifiers were trained and tested 5 times rotating the subsets used for training; the estimate of accuracy is the overall number of correct classifications divided by the number of instances in the dataset; finally, the classifier performance is measured with the average of the accuracy throughout the rotation process.

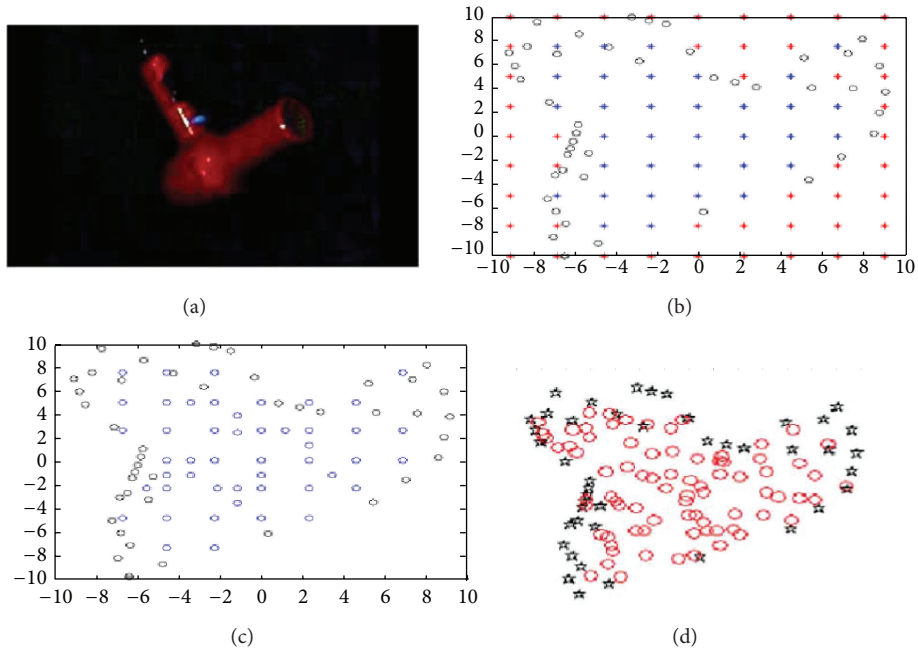

Figure 6: Different stages of the mesh grid procedure for the hair dryer object. (a) Original input image. (b) Feature extraction. (c) Contour created for the object. (d) Depth information attached in a mesh grid.

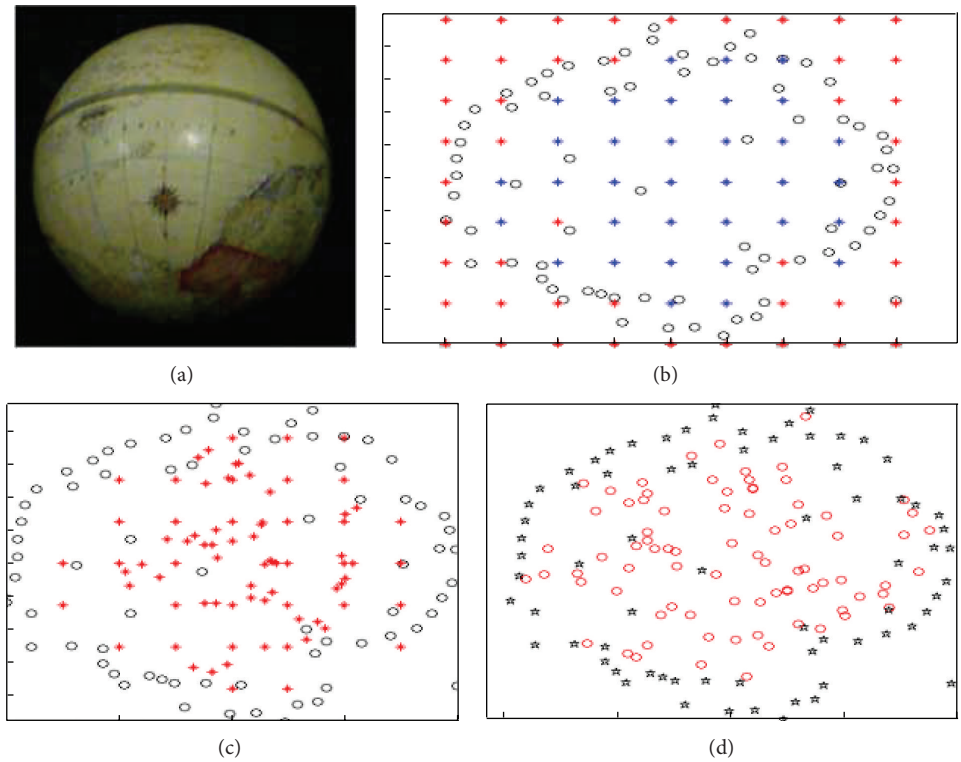

Figure 7: Different stages of the mesh grid procedure for the earth globe object. (a) Original input image. (b) Feature extraction. (c) Contour created for the object. (d) Depth information attached in a mesh grid.

Figure 8: Objects conforming the data set.

To define the SVM model, different SVMs with linear, $\mathbf{x}_i' \cdot \mathbf{x}_j$, and polynomial, $(\alpha \mathbf{x}_i' \cdot \mathbf{x}_j + b)^d$, kernel functions were tested varying the cost parameter C using values in the range $(10^2, 10^8)$, and the degree of the polynomial d using values in the range $(3, 10)$; cross-validation was used to define which value is the best. A supervised learning approach was considered in the whole training and validation processes; the classifiers were trained using a labeled dataset of images, in each image the information about which object is present and the bounding box of the object was also provided. Therefore, the class to which each descriptor belongs is known. To classify each object 100 tests were performed.

6.3. Classification. The first experiment consisted of binary classification of one object being discriminated from another, 1 versus 1 classification; see Table 1. The results are summarized in a result table. Each row indicates the percentage of (a) correct recognitions (CR) and (b) class one errors (C1E); objects of class one are classified as objects of class two and (C2E) class two errors; objects of class two are classified as objects of class one.

The second test consisted of binary classification using the 5 objects (an object being discriminated from the rest); see Table 2. In this case, result table contains the percentages of (a) correct recognitions (CR), (b) false positives (FP), a different object is classified as the target object, and (c) false negatives (FN), the target object is classified as a different object.

Table 1: Binary classification using 2 objects.

1 versus 1 classification percentages			
Test	PSA-moment descriptor		
Name	CR	C1E	C2E
Dryer and box	100%	0%	0%
Can and irons	94%	3%	3%
Cup and can	86%	10%	4%
Iron and dryer	88%	4%	8%
Box and cup	90%	0%	10%

Table 2: Binary classification using 5 objects.

1 versus all classification percentages			
Test	PSA-moment descriptor		
Name	CR	FP	FN
Dryer	96%	2%	2%
Iron	88%	0%	12%
Cup	92%	2%	6%
Can	90%	10%	0%
Box	96%	2%	2%

Figure 9: Objects from the RGB-D data set of the University of Washington [44].

Finally, a test was made in multiclass classification where objects were classified all at once. Each object was set as a different class and the classification delivered the class to which the objects belong. This test generated a 79% of correct classification. An 8% of incorrect classification was present in class 1 (cups) and 9% in class 2 (irons); all objects of classes 3 and 4 (irons and cans) were correctly classified, and 4% of incorrect classification was in class 5 (box).

After testing the descriptors with the small data set, classification tests were performed using a group of objects from the RGB-D object data set of the University of Washington. All objects were discriminated from the rest in binary classification. Tests were performed using first the moment descriptor obtained when points in the net are simply migrated, then the same tests were done using the point spreading approach. Later, these results were compared with histograms of oriented gradients (HOG) [42], scale invariant feature transform (SIFT) detector [10], and a detection system using cascades of HAAR-like features [43].

The classification has been made for each descriptor (moment, moment-PSA, HOG, and SIFT) using SVMs. From cross-validation the best results obtained were a linear kernel function with a cost parameter $C = 10^3$. Results of the tests for the four objects are summarized in Table 3: object one is the bowl, object two is the pitcher, object three is the cap, and object four is the soda can.

We can see from these results that spreading the points over the object surface improves the descriptor computed and then the classification of the objects. With this approach we have similar results to the well-known HOG descriptors

Table 3: Tests using objects from the Washington University data set.

Classification percentages												
Object	Object 1			Object 2			Object 3			Object 4		
Result	CR	FN	FP	CR	FN	FP	CR	FN	FP	CR	FN	FP
Moment descriptor	80	8	12	96	0	4	92	4	4	79	7	14
Moment-PSA descriptor	85	5	10	98	0	2	97	1	2	82	8	10
HOG descriptor	99	1	0	98	2	0	90	8	2	85	10	5
SIFT descriptor	74	22	4	85	8	7	76	10	14	70	27	3
HAAR cascade	90	9	1	89	7	4	75	10	15	72	27	1

Table 4: Average and minimum number of support vectors.

Support vectors								
Object	Object 1		Object 2		Object 3		Object 4	
Support vectors	avg.	min.	avg.	min.	avg.	min.	avg.	min.
Moment descriptor	20	15	25	17	101	73	27	20
Moment-PSA descriptor	8	5	24	12	73	52	12	10
HOG descriptor	300	23	400	60	370	43	418	70
SIFT descriptor	310	22	404	65	401	84	450	72

Table 5: Descriptor construction time. Where the steps are feature extraction (FE), contour creation (CC), mesh reduction (MR), mesh optimization (MO), and invariant moments (IM).

Step	FE	CC	MR	MO	IM
Time	0.0929 s	0.0419 s	0.0202 s	0.0566 s	7.85×10^{-4}

which are used in many of the state-of-the-art recognition systems. These two descriptors achieved the highest correct recognition percentages in all the tests.

The accuracy of the classification system has been assessed in terms of receiving operating characteristics curves (ROC) which relate the positive and false acceptance rates according to an acceptance threshold δ varying in the rate $[0, 1]$. Figure 10 shows the ROC curves for the classification of the bowl and Figure 11 the ROC curves for the classification of the coffee mug.

With the ROC curves, we can also note that the moments-PSA classifiers show improvement over the one with the grid that is not optimized, and also it shows similar performance to the classification through HOG descriptors.

The efficiency of the descriptors using a SVM classifier was also evaluated in terms of the number of support vectors obtained by the SVM algorithm. This was done in the cases of moment, moment-PSA, HOG, and SIFT descriptors. Table 4 shows the average and minimum number of support vectors for 100 tests on the classification of the four objects used in Table 3.

With these results we can see that although HOG and our approach have similar classification results, the number of support vectors needed in the training process is smaller for our approach, and when the distribution of the points is optimized it also results in patterns that are better distributed in the feature space so we need even less support vectors.

In this paper we propose the design of a new small and robust feature descriptor based on SURF features, 3D data obtained by RGB-D sensor, an optimization based on a modified PSO, and moment invariants. The experimental results presented in this section show that even that the new descriptors are small the addition of 3D data is advantageous and provides robust features. The importance of the mesh optimization step lies on the reduction of points and more distributed points over the object. Table 4 shows that the proposed approach requires less support vectors; therefore the classification is faster. From the ROC curves we can also note that the proposed approach has a performance compared with the HOG approach, but as we mention, with fewer support vectors.

In order to determine the time required to construct the descriptor, we have tested each step 100 times for each object to estimate its average processing time, a resume of this results is presented in Table 5. In this table, we can note that the most time consuming process is the feature extraction process and that the computation of the invariants moments of the optimized mesh is insignificant. With respect to the classification time, we use the same process used to estimate the descriptor construction time, and we obtained an average of 0.007565 s. The implementation of the algorithm has been made in Matlab, and therefore it can be improved by a C or C++ implementation.

7. Discussion

7.1. Difficulties of Each Step of the System. In the previous section the different steps of the proposed approach had been presented. The approach produces a descriptor based on moment invariants which has been computed using 3D data provided by an RGB-D sensor. A new method has been introduced for the extraction of object models from the 3D data.

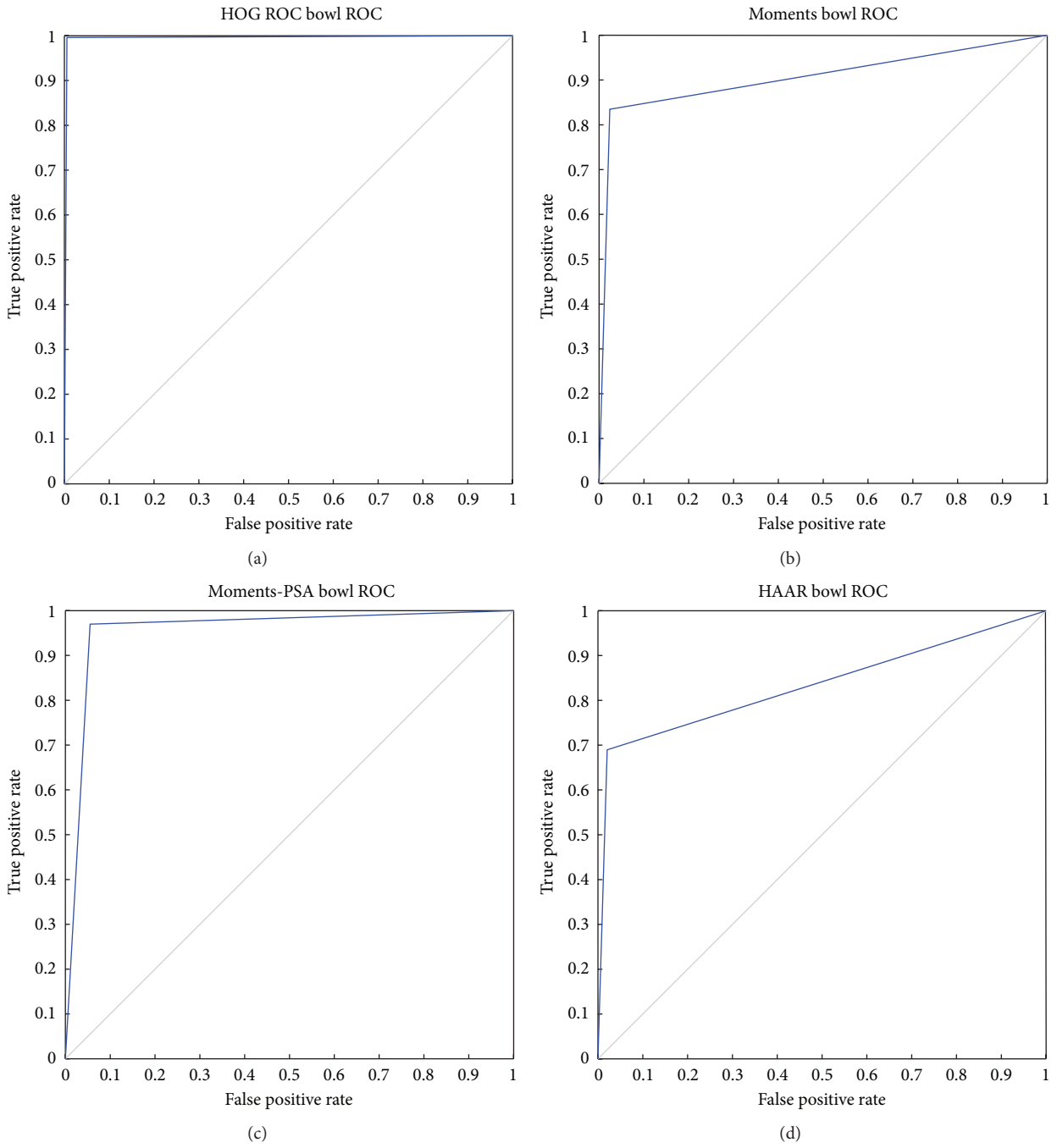

Figure 10: ROC plots for the classification of the bowl. (a) HOG ROC, (b) moments ROC, (c) moments-PSA ROC, (d) HAAR ROC.

The first step of the procedure is the feature extraction; in this approach the SURF detector was used, and the difficulties of obtaining the SURF features are the traditional of any image feature detector. The second step is the contour creation: this step takes as input the features detected in the previous step and extracts the contour of the object. In this step we must take into account that some interest points do not belong to the object; they are close to the contour but they are outside of it. This point can be seen as noise, and they are eliminated if their distance to the centroid is the double of the mean distance.

After the construction of the object contour, the depth information is added. If the area occupied by the object is big, then there would be a lot of data; thus the next steps will require a lot of computational process. For this reason, the third step of the proposed approach tries to optimize the mesh in order to reduce the computational time. In this step a modified version of PSO has been used to optimize the points

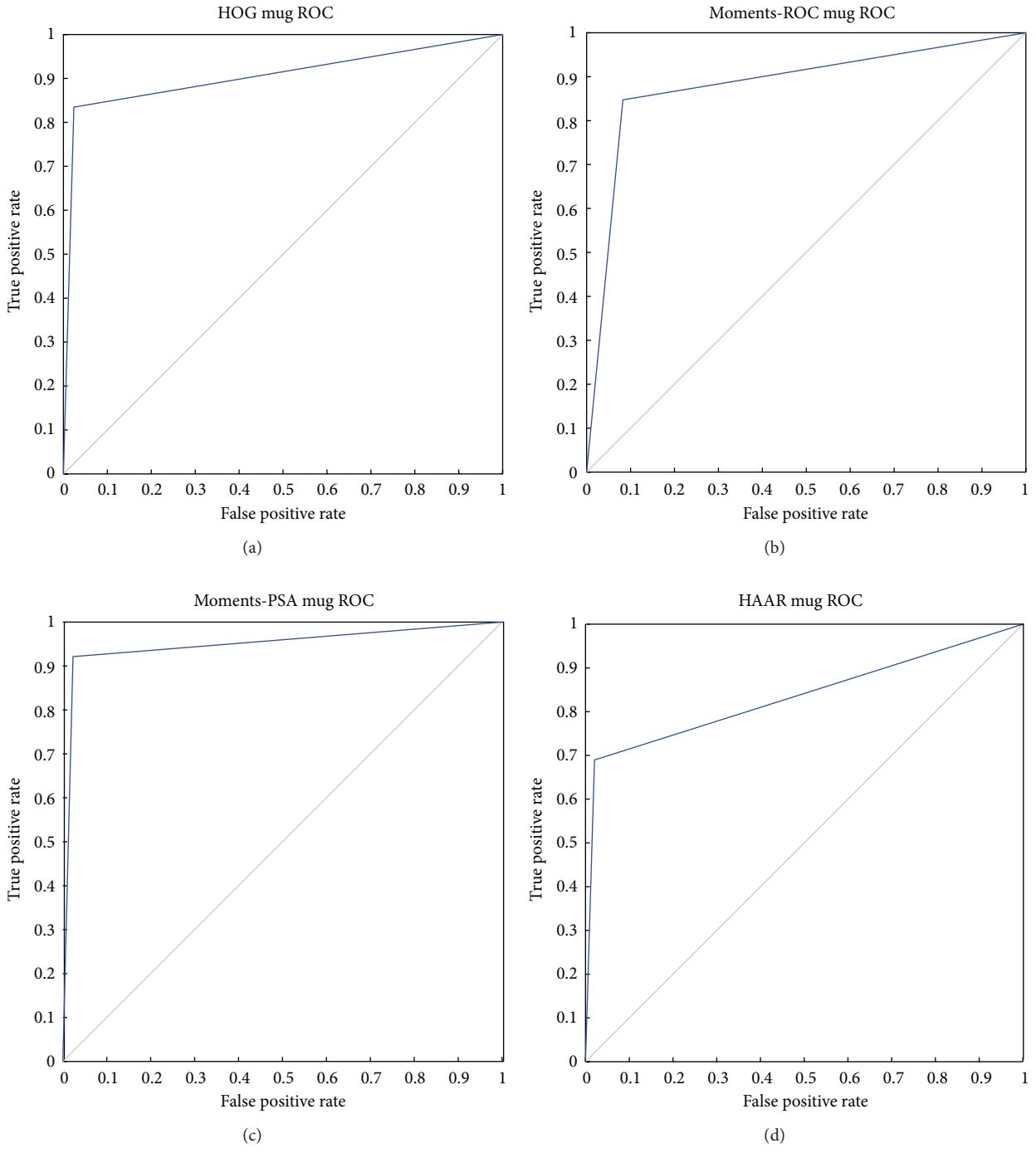

Figure 11: ROC plots for the classification of the coffee mug. (a) HOG ROC, (b) moments ROC, (c) moments-PSA ROC, (d) HAAR ROC.

on the mesh. One of the challenges encountered on this step was the stop criteria; we cannot force PSO to stop when the best value is zero, and therefore the algorithm stops when the best value of each particle i (P_i) is smaller than a certain threshold. With respect to the SVM, the only problem was the estimation of the cost parameter C, but this was solved by using cross-validation.

7.2. Computation Complexity of the System. Taking into account the steps illustrated in Figure 1, next, a complexity analysis of the classification system per step is presented.

With respect to the feature extraction step, as we mentioned before, it is made by using a SURF detector. The theoretical complexity of SURF was determined and validated through experimentation in [48]; this is $O(MN + K)$,

where M and N are the image height and width, respectively, and K stands for the number of detected points. It was proven that SURF is a very attractive algorithm concerning its performance when usual size and high rates of frames by seconds are used; this is our case; the images' size is around 640×480 pixels. This step produces a dense 3D point cloud that is greatly reduced in the next steps.

The contour creation step is a linear process with respect to the number of points of the cloud produced by the SURF algorithm; nevertheless when the contour is created this number is reduced and only the points which belongs to the silhouette of the object are taken to be processed by the mesh reduction step, which is also linear with respect to this reduced number of points.

The mesh optimization step is performed by using the PSO algorithm. The number of computations to complete a PSO run is the computations of the cost function and the position and velocity updates, which are directly proportional to the number of particles and iterations of the algorithm. The computational complexity of evaluating the cost function depends basically on the Euclidean distance of the particles, which requires $2N$ additions, $2N$ multiplications, and N square roots, where N represents the swarm size. Whereas the position update requires $2N$ multiplications and $2N$ additions, and the velocity update requires $7N$ multiplications and $4N$ additions. Therefore in total the PSO requires $6N$ additions, $11N$ multiplications, and N square roots. Again this step reduces even more the number of points which belong to the contour of the object.

The invariant moment descriptor generation seems to be the more expensive step which is executed in real time by the system because it is a process with a computation complexity of $O(N^3)$, with N the number of points of each contour, but this number is small (because it was reduced by the above three steps of the system) and the descriptors are very fast computed as it is shown in Table 5.

The last step of the system is the classification of the descriptors obtained; it is made using a SVM. A SVM solves a quadratic programming problem (QP) and therefore its computation complexity depends on the QP solver used. The computation complexity of the traditional algorithm of the SVM classifier is $O(N_{sv}^3 + LN_{sv}^2 + d_l LN_{sv})$ [49], where d_l is the dimension of the input space, L is the number of training points, the number of the samples in the support vector set is N_{sv}. Nevertheless there are several efficient implementations, such as incremental learning algorithm [50] which reports a computation complexity of $O(N_{sv}^3 + d_l N \text{sv}^2)$, or sequential minimal optimization (SMO) [51] which for a real-world test sets are 1200 times faster for linear SVMs and 15 times faster for nonlinear SVMs against another quadratic programming solver techniques. SMO was the training method used in our system. It is important to note that, even when the system includes a training step with the complexity that was already determined for the SVM algorithm, this step is made offline and, in the real time process, the classification step consists of evaluating the classification function obtained with the training of the SVM and this is a linear process with respect to the number of support vectors N_{sv}, which is usually related with the number of training data N as follows $N_{sv} \ll N$.

8. Conclusions

In this paper we have presented an image classification technique based on an invariant moment descriptor that includes depth information. The inclusion of 3D data enables invariant moments to produce small and robust descriptors improving image classification. To create such descriptors, we used object models with rich information of the distribution of points. The application of the optimization algorithm PSO to the model generation stage improved the computed descriptor and the object recognition. From experimental results, it is clear that these descriptors have achieved high correct recognition percentages. Furthermore, the number of support vectors obtained in the training process is smaller for our approach, due to the fact that the points are optimized and thus the patterns are better distributed in the feature space.

Abbreviations

X_i: Position of PSO particle i
P_i: Best position found by PSO particle i
V_i: Velocity of PSO particle i
G: Best position found by the swarm
ω: PSO inertia weight
φ_1, φ_1: Uniform random numbers $[0, 1]$
c_1, c_2: PSO acceleration constants
f_i: PSO fitness function
α: Fitness penalization gain
β: Particle closeness to boundaries
ρ: Particle acceleration with respect to closest particle
d: Moment descriptor
D_i: Distance of particle i with respect to its neighbors
$\overline{D}$: Mean of particles distances.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank CONACYT and the University of Guadalajara. This work has been partially supported by the CONACYT projects CB-156567, CB-106838, CB-103191, and INFR-229696.

References

[1] M. Minoux, *Mathematical Programming: Theory and Algorithms*, Wiley-Interscience, 1986.

[2] K. Parsopoulos and M. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*, IGI Global, 2010.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[4] M. Jin and D. Wu, "Collision-free and energy-saving trajectory planning for large-scale redundant manipulator using improved

pso," *Mathematical Problems in Engineering*, vol. 2013, Article ID 208628, 8 pages, 2013.

[5] A. Y. Alanis, E. Rangel, J. Rivera, and C. Lopez-Franco, "Particle swarm based approach of a real-time discrete neural identifier for linear induction motors," *Mathematical Problems in Engineering*, vol. 2013, Article ID 715094, 9 pages, 2013.

[6] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[7] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, 1995.

[8] Kinect sensor, http://www.xbox.com/en-US/Kinect.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 316–323, 1999.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)*, vol. 2, pp. 1458–1465, October 2005.

[12] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

[13] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151–172, 2000.

[14] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[15] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proceedings of the 8th International Conference on Computer Vision*, vol. 1, pp. 525–531, July 2001.

[16] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, pp. 1150–1157, Kerkyra, Greece, September 1999.

[17] H. Bay, T. Tuytelaars, and L. Gool, "Surf: speeded up robust features," in *Proceedings of the Computer Vision Conference (ECCV '06)*, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951 of *Lecture Notes in Computer Science*, pp. 404–417, Springer, Berlin, Germany, 2006.

[18] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[19] J. Flusser, B. Zitova, and T. Suk, *Moments and Moment Invariants in Pattern Recognition*, John Wiley & Sons, Chichester, UK, 2009.

[20] G. Carneiro and A. D. Jepson, "Phase-based local features," in *Proceedings of the European Conference on Computer Vision (ECCV '02)*, pp. 282–296, 2002.

[21] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "General intensity transformations and differential invariants," *Journal of Mathematical Imaging and Vision*, vol. 4, no. 2, pp. 171–187, 1994.

[22] N. Y. Khan, B. McCane, and G. Wyvill, "SIFT and SURF performance evaluation against various image deformations on benchmark dataset," in *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA '11)*, pp. 501–506, December 2011.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 886–893, San Diego, Calif, USA, June 2005.

[24] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[25] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, vol. 2, pp. 734–741, October 2003.

[26] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[27] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, New York, NY, USA, 2000.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[29] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1992.

[30] P. Merz and B. Freisleben, "A genet ic local search approach to the quadratic assignment problem," in *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 465–472, Morgan Kaufmann, 1997.

[31] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[32] M. Eusuff and K. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[33] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[34] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 315–346, 2013.

[35] Q. Ni and J. Deng, "Analysis of population diversity of dynamic probabilistic particle swarm optimization algorithms," *Mathematical Problems in Engineering*, vol. 2014, Article ID 762015, 9 pages, 2014.

[36] N. A. A. Aziz, M. Y. Alias, A. W. Mohemmed, and K. A. Aziz, "Particle swarm optimization for constrained and multiobjective problems: a brief review in," in *Proceedings of the International Conference on Management and Artificial Intelligence (IPEDR '11)*, pp. 146–150, Bali, Indonesia, 2011.

[37] M.-P. Song and G. C. Gu, "Research on particle swarm optimization: a review," in *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 2236–2241, August 2004.

[38] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[39] M. Mercimek, K. Gulez, and T. V. Mumcu, "Real object recognition using moment invariants," *Sadhana*, vol. 30, pp. 765–775, 2005.

[40] M. Rizon, H. Yazid, P. Saad et al., "Objectdetection using geometric invariant moment," *American Journal of Applied Sciences*, vol. 2, pp. 1876–1878, 2006.

[41] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[42] N. Dalal and B. Triggs, "Histograms of oriented gradients f or human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 886–893, IEEE Computer Society, Washington, DC, USA, 2005.

[43] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. I511–I518, December 2001.

[44] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 1817–1824, May 2011.

[45] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3D scenes," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '12)*, pp. 1330–1337, Saint Paul, Minn, USA, May 2012.

[46] L. Mussi, F. Daolio, and S. Cagnoni, "Evaluation of parallel particle swarm optimization algorithms within the cuda architecture," *Information Sciences*, vol. 181, no. 20, pp. 4642–4657, 2011.

[47] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI '95)*, vol. 2, pp. 1137–1143, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1995.

[48] P. Drews, R. de Bem, and A. de Melo, "Analyzing and exploring feature detectors in images," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN '11)*, pp. 305–310, Lisbon, Portugal, July 2011.

[49] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1st edition, 1998.

[50] Y. Wang, F. Zhang, and L. Chen, "An approach to incremental SVM learning algorithm," in *Proceedings of the ISECS International Colloquium on Computing, Communication, Control, and Management (CCCM '08)*, vol. 1, pp. 352–354, August 2008.

[51] J. C. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Tech. Rep., Advances in Kernel Methods-Support Vector Learning, 1998.