International Journal of Intelligent Robotics and Applications (2017) 1:410–428
https://doi.org/10.1007/s41315-017-0037-3

REGULAR PAPER

Combining boosting machine learning and swarm intelligence for real time object detection and tracking: towards new meta-heuristics boosting classifiers

Sidahmed Benabderrahmane[1]

Received: 22 August 2017 / Accepted: 6 November 2017 / Published online: 22 November 2017

Abstract
Artificial vision in robotics involves real time detection of objects for fast decision making. Such intelligent systems require efficient algorithms and big learning database of examples for producing robust classifiers. Several methods of objects detection and tracking have been proposed in the literature. However, even though the detection rates have been improved, the processing time and the complexity of the models still representing a key challenge. In this paper, we present a real time object detection and tracking framework based on Adaboost classification, where a strong classifier is generated using an iterative combination of weak learners. This method is based on the use of discriminative features by analyzing different regions of the input image. Instead of performing a full traversal in the entire search space of all possible visual features, we propose to use intelligent heuristics for accelerating time processing and extracting relevant features in the image that lead to a best detection rate. The meta-heuristics involve the use of genetic algorithms, particle swarm optimization, random walk and a novel hybrid combination of these methods. The obtained results, in a case of intelligent transportation system, have shown considerable improvements in term of computation time, efficiency and accuracy.

Keywords Machine learning · Boosting · Metaheuristics · Intelligent systems · Computer vision in robotics · Object detection

1 Introduction

Computer vision techniques in intelligent real time systems need sophisticated and fast decision making implementations. The bottlenecks in these systems concern the modeling of the external variabilities caused by the scale, the positions and the illuminating conditions during the detection step (Abril et al. 2007; Ho et al. 2003; Mekami et al. 2018). Though these difficulties are still a big challenge, several methods have been proposed in the literature by employing different machine learning paradigms, which have given good performances (Andreopoulos and Tsotsos 2013; Shantaiya et al. 2013; Yilmaz et al. 2006).

One computer vision system for face detection based on ensemble learning, have been proposed by Viola and Jones (2001b). This framework can be generalized to detect any kind of objects (faces, cars, pedestrian...) given as positive examples in the learning database (DB). Their method is based on the combination of a cascade of simple classifiers (weak learners) using the Haar-like features on the image of interest. These weak classifiers are generated using Adaboost algorithm (Schapire 1999a), and they are combined to generate a strong classifier able to detect and track complex objects. The cascade framework in Adaboost allows filtering negative examples away quickly.

The Haar-like features are based on the wavelet representation previously introduced by Papageorgiou et al. (1998). The main idea is the calculation of the difference between two or multiple rectangular regions of the analyzed input image. A sliding window (in general 24×24 pixels) is applied on different positions of the image, and in each position a threshold is calculated for a feature as the sum of pixels intensity in black regions differentiated with the sum of those in white regions. The calculated threshold of each feature in each position of the sliding window is a weighted score on the learning examples.

✉ Sidahmed Benabderrahmane
sidahmed.benabderrahmane@gmail.com

[1] University of Edinburgh, School of Informatics, 10 Crichton Street, Edinburgh EH8 9AB, UK

Even though the calculation of these features seems simple, major drawbacks reside in the computation time during the training and the detection, since it needs to normalize the histogram (the variance and the mean) of each sliding window position prior to applying the features.

In Abramson and Freund (2005) an improved visual learning features called control points have been proposed, to cope with time processing and poor detection rates of complex objects encountered with Viola and Jones' features. Regarding the authors, this new kind of features, totally liberate the detection system from the histogram normalization of the examined sub-window.

Viola and Jones' algorithm perform a complete scan on the input image with different scales, thus it is rather lengthy and computationally expensive. This full search procedure is due to the poor power of discrimination of the weak classifier obtained by the Haar-like features. In other words, Adaboost tries to combine a lot of weak learners by exploring the entire search space. Moreover, expanding the number and the type of visual features (Viola and Jones or control points) will increase drastically the size of the features set. Consequently, the search space would became huge and time processing of Adaboost during the learning and detection will increase exponentially.

To overcome with these inconveniences, we investigated in this paper, the use of meta heuristics search strategies combined with Adaboost for detecting and identifying complex objects learned as positive examples. More specifically, we suggest the introduction of particle swarm optimization (PSO), Genetic Algorithms, and a new hybridization technique for exploring the search space of the visual features instead of a full and exhaustive search initially used in the existing detection frameworks. The obtained results demonstrated that our method increased explicitly the detection rate, and decreased time processing during the training and detection steps.

2 Chronology of the existing methods for real time object detection and tracking

Machine learning is an important research area in artificial intelligence. It refers to implementations, theorems, and methods that allow to a machine to evaluate and perform intelligent tasks throughout a learning process.

Machine learning algorithms can be categorized in two major classes: supervised or unsupervised methods. In the first category, a learning database of labeled examples is used to create an expert decision maker for classifying future examples in the right class of labels. Neural networks, support vectors machine (SVM), Decision Trees are typical examples of supervised machine learning methods. In the unsupervised machine learning category, no expert is needed. The algorithm should discover inherent structures among the data. The Kmeans, PAM (partitions around medoids), or hierarchical clustering are the typical unsupervised algorithms, which learn the global behaviors of the data among the available observations (Mitchell 1997; Bishop 2006; Witten et al. 2011; Marsland 2009; Drew and John 2012; Metidji et al. 2013).

During the past decades there have been an explosion of proposed machine learning methods for object recognition and tracking (Antani et al. 2002; Liu et al. 2013). In fact, real time object detection has become an entire challenging discipline in computer science with the aim of multiple objectives such as: video-surveillance, autonomous robot navigation, intelligent vehicle, drones navigation, biometrics,....

Regarding Yilmaz et al. (2006), the aim of an object detection and tracking is to generate the trajectory of a moving object over time by locating its position. These authors differentiated many methods of tracking as depicted in Fig. 1.

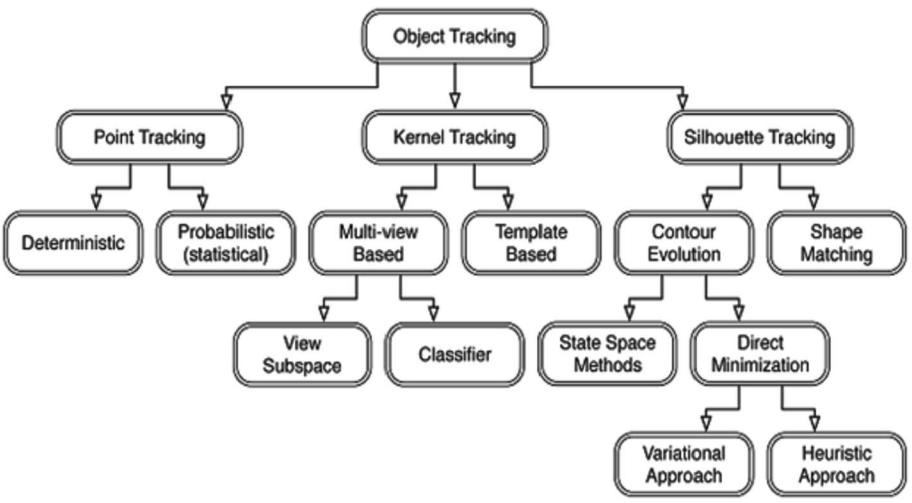

Fig. 1 Different object detection and tracking categories in Yilmaz et al. (2006)

Overall, Object detection methods are based on four categories of methods:

- Knowledge-based methods: Use intuitive representation of the object that we want to detect.
- Feature invariance: Extract discriminative attributes of the detected object.
- Template matching: The detected object is represented with global model templates.
- Appearance model: Learn models from a database of images that contain examples of the detected object. The images should contain different variabilities during the capture for suitable detection.

In the knowledge-based methods (Yang et al. 2002), rules are extracted from images using the knowledge and perceptions of the human on the objects. For example in the case of faces detection, it is easy to propose simple rules for describing primitives of a face and their relationships. For instance, a human face in an image often appears with two eyes that are symmetrical. The relative positions of the various components of the face are studied after being detected. The difficulty in these approaches concern the translation by strict rules, the way that the researchers represent the detected objects.

In the features based methods, the objective is to detect invariant characteristics of the object. Boundaries detectors (eg. Canny filter), morphological estimators (erosion and dilatation), shapelets extractor are usually used to isolate the object from the rest of the image (Yang et al. 2002).

The template matching methods aim at using a model that is manually parameterized, to give a standard representation of the detected object. The main issue of these approaches concern the lake of generalization since it is not possible to represent all possible variations an object can have in an image (illumination, scale...) (Brunelli and Poggio 1993).

The final category concerns appearance models, in which statistical analysis and artificial machine learning (SVM, Neural Networks, Deep Learning, Boosting, KNN...) are used to find discriminating characteristics to classify the object which the user wants to detect. The characteristics learned are in the form of probabilistic distributions or discriminatory functions which are therefore used for the detection. In the meantime, the reduction of dimensionality is usually carried out for calculation quickness and detection efficiency (LeCun et al. 2004).

In this paper we are focusing our work on boosting-based classification systems, which are based on learning models from a database of images. We will show how it is efficient in term of rate detection and time processing, when combining Adaboost with metaheuristics models. The proposed object detection framework was implemented and evaluated with real cars database. The fundamental definitions as well as the experimentations are presented in the next sections.

3 I-SwarmBoost: combining metaheuristics, Adaboost and visual features for real time object detection

3.1 Principle of boosting classification

Boosting is a general method that tries to improve the accuracy of a given learning algorithm. The theoretical basis of the method comes from research studies of Ehrenfeucht and Haussler (1989), Valiant (1984). The problem was to check whether a weak learning algorithm (Weak Learner), can do better than the simple chance. The principle is based on the combination of these weak learners (also called hypotheses), by successive iterations in order to generate a final strong learner. This concept is called probably approximately correct learning (PAC learning). A weak learner is an algorithm that provides weak classifier capable of recognizing two classes at least as well as chance does. Each provided classifier is weighted by the quality of its classification: the better it classifies, the more important weight it will have. The misclassified examples are boosted so that they are more important to the weak learner on the next loop of the algorithm.

Several boosting algorithms have been proposed in the literature (Schapire 1999b). For example, Adaboost (for Adaptive boosting) was introduced in Rudin et al. (2004), Freund et al. (2003) for binary classification problems, Textboost (Schapire and Singer 2000) for textual documents classification and categorization, FilterBoost for regression analysis (Bradley and Schapire 2007), in Collins et al. (2002) it was used for logistic regression and distance learning, and finally in human-computer spoken-dialogues systems as in Schapire (2002).

In this paper we are going to concentrate our work on *Adaboost* to propose our classification model for object detection in images and video streams. Adaboost is one of the learning algorithms that were mostly used for object detection and tracking tasks (Wu et al. 2015; Schapire 1999a). As a special case of boosting methods, Adaboost is based on the generation of a strong learner by combining a cascade of weak learners. In each iteration, Adaboost searches the most discriminative feature, by learning a weak classifier that minimizes classification error on a set of training examples. It has been shown that the training error exponentially trends to zero as the number of classifiers improve. Algorithm 1 gives a pseudo code of Adaboost. It takes as input a set labeled training examples $\mathcal{D} = \{(x_m, y_m), n = 1 \dots n, y_n = \{-1, 1\}\}$, where each

example x_i is labeled by a class y_j. Adaboost iterates around a prefixed number of rounds $t = 1 \cdots T$,

ALGORITHM 1: AdaBoost algorithm for Binary datasets

INPUT:
Training data: $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, N\}$, where $x_i \in \mathbf{R}^d$ and $y_i \in \{-1, +1\}$.
The number of sampled examples in each iteration: m
Weak learner: $\mathcal{L}$ that automatically learns a binary classifier $h(x) : \mathbf{R}^d \mapsto \{-1, +1\}$ from a set of training examples.
The number of iteration: T
OUTPUT:
The final classifier: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
Algorithm
Initialize the distribution $D_0(i) = 1/N, i = 1, \ldots, N$
for $t = 1$ to T **do**
  Sample m examples with replacement from $\mathcal{D}$ according to the distribution $D_{t-1}(i)$.
  Train a binary classifier $h_t(x)$ using the sampled examples
  Compute the error rate $\varepsilon_t = \sum_{i=1}^{N} D_{t-1}(i) I(h_t(x_i) \neq y_i)$ where $I(z)$ outputs 1 when z is true and zero otherwise.
  Exit the loop if $\varepsilon > 0.5$.
  Compute the weight α_t as

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

  Update the distribution as

$$D_t(i) = \frac{1}{Z_t} D_{t-1}(i) \exp(\alpha_t I(y_i \neq h_t(x_i))$$

  where $Z_t = \sum_{i=1}^{N} D_{t-1}(i) \exp(\alpha_t I(y_i \neq h_t(x_i))$.
end for
Construct the final classifier $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.

These examples have at the first step, an uniformed initial weight distribution $D_1(i) = \frac{1}{n}$ that will be updated regarding the obtained results in the previous steps. At each step, a binary classifier $h_t(x)$, which is a classification hypothesis, is generated and the error rate ε_t is calculated. A weighting scheme α_t is introduced on each hypothesis to measure its importance. The final classifier (hypothesis) H is calculated with a majority weighted vote using the previously generated weak classifiers $h_t(x)$.

The interest of Adaboost resides in its ability to reduce the learning error. Freund et al. (2003) have proved that if each weak hypothesis having an error slightly better than chance then the learning error of Adaboost decreases exponentially and rapidly with steps t.

3.2 Visual learning

As reported above, Adaboost is a binary classification algorithm. It has been widely used in a lot of computer vision problems, more precisely for detecting faces/non-faces images. In these systems, Adaboost was combined with visual features extraction methods for identifying human faces in image sequences. Papageorgiou et al. in (1998) were among the pioneers in the utilization of visual features for object detection in static images of cluttered scenes. Their proposed approach was based on Haar-like wavelet representation of an object belonging to a class. Figure 2-Top gives an example of the learning database used by these authors, and Fig. 2-Bottom represents Haar-like wavelets that were applied on the images for visual features extraction.

Fig. 2 Top: An example of the learning database used in Papageorgiou et al. (1998) for pedestrian detection. Bottom: The results of Haar wavelets on the images for features extraction

Thanks to Papageorgiou et al. works, Viola and Jones (2001a, b) proposed a new visual object detection framework that has the ability of identifying an object in an image very rapidly. The main contributions of their work concerned in three points: (i) the application of a new image representation called the Integral Image (also called Rectangular Features RF, described in next sub section) which is similar of the wavelet representation inspired from Papageorgiou et al. (1998), (ii) the utilization of Adaboost for the selection of a small number of discriminative visual features from a larger set to enhance the classification, and (iii) the combination of complex classifiers in cascading way, which allows background regions of the image to be quickly eliminated while spending more computation on promising object-like regions. The important results obtained by Viola and Jones have helped the reputation of their rectangular features and facilitated the production of several libraries that implement this technique. Some future contributions have been proposed to improve this Adaboost-based object recognition system, for instance: Lienhart et al. (2003), Lienhart and Maydt (2002) with their new set of visual features, and Ghimire and Lee (2013), Wu et al. (2015) for their multi-classes Adaboost classifiers to the aim of facial expression recognition in image sequences. The following subsections give more details on the visual features that are used within Adaboost for object detection.

3.3 Rectangular features

Viola and Jones proposed an Adaboost object detection system (Viola and Jones 2001a, b; Lienhart et al. 2003) using a set of basic Haar-like wavelet as rectangular features, in a cascading way, as illustrated in Fig. 3a. These features indicate the signal intensity difference of pixels in local rectangular regions neighboring each other. The response of each feature is considered as a simple classifier, and is calculated as the difference between the sum of pixel

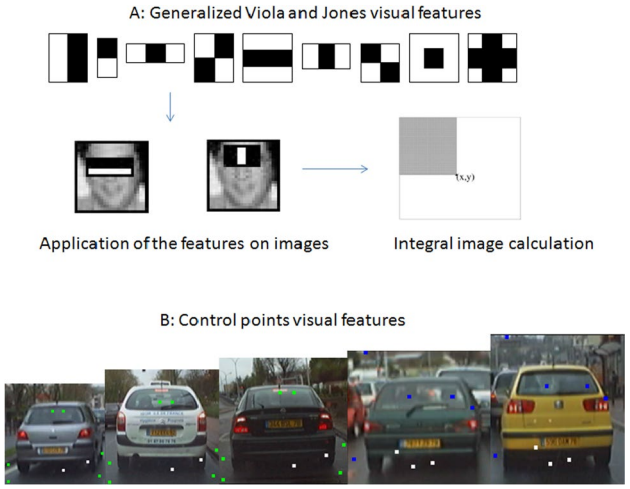

Fig. 3 Generalized Haar-like visual features used by Viola and Jones (**a**). The features are applied on an input image, and the integral image is obtained by the calculation of the difference between two rectangular regions. Control points features (**b**) are based on individual pixel values examination

intensity in gray and white regions. The advantage of using the rectangular features is the simplicity of their calculation within the Integral Image representation (see Fig. 3a): $II(x, y) = \Sigma_{x' \leq x, y' \leq y} I(x', y')$. Further, Lienhart et al. in Lienhart et al. (2003) proposed an extension of the rectangular features, as in displayed in Fig. 4. By considering the positions of the features in the image, and their types, we can have five degrees of freedom. Consequently, by varying the positions of the features in the analyzed image, the width, the height, and the type, we can generate a huge set of features (260,000 according to the authors using 24×24 window in the image). This number can grow up with the extension of novel features. Hence computing complexity can be raised up. The role of Adaboost is to extract the discriminative features, i.e., those generating low classification errors that are the total weight of all misclassified examples. Figure 5 represents an example of good features kept by Adaboost during the learning. They represent the selected week learners.

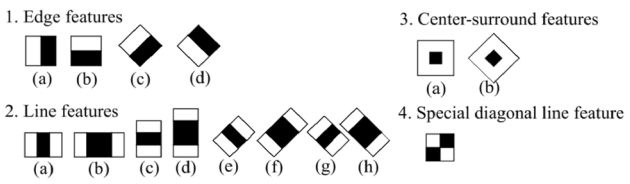

Fig. 4 Feature prototypes of simple haar-like and center-surround features. Black areas have negative and white areas positive weights (Lienhart et al. 2003)

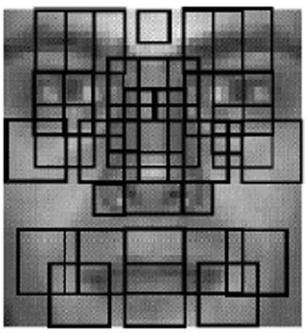

Fig. 5 Example of good features kept by Adaboost during the learning. They represent the selected week learners

3.4 Control points features

To overcome with the high processing complexity encountered with rectangular features of Viola and Jones, authors in Abramson et al. (2007) proposed an improvement to the previous object detection schemes, by introducing a new kind of visual features called *Control points* (CP), which are based on sampling individual pixels within the examined sub-window, instead of comparing the sums of pixel values in rectangular regions. It has been shown that these features are faster and do not demand the preparation of an integral image, nor variance-normalization of each sub-window. Figure 3-B gives an example of control points features that are applied on car learning database. On each image, and in different resolutions, a set of positive $\{x_1, x_2, \ldots, x_n\}$ and negative points $\{y_1, y_2, \ldots, y_m\}$ are generated on the image. The feature answers positively if and only if for every control point $x \in \{x_1, x_2, \ldots, x_n\}$ and every control point $y \in \{y_1, y_2, \ldots, y_m\}$, we have $val(x) \geq val(y)$.

Each CP feature tests the relations between pixels. It tries to look for a predefined order of pixels in the image, and if such is not found, it labels it negatively (Abramson et al. 2007). Thus, CP features are order-based and not intensity-based, they do not consider the difference between two pixels values, but the sign of that difference. It is therefore insensitive to variance normalization.

3.5 The global architecture of I-SwarmBoost

Motivations We have shown that AdaBoost needs a weak learner, i.e., an algorithm which provides a good feature in each iteration. Doubtlessly, choosing the best simple classifier at each step cannot be done by testing all visual features possibilities (varying the positions of the features in the image, the resolution, and the class of the features), since there are more than 260,000 rectangular features, and about 10E32 Control Points features. Therefore, the use of a metaheuristic algorithm became evident, to explore the research

space of the features in an intelligent way, and generating simple classifiers and iteratively improving them. Thus we are proposing here *I-SwarmBoost*, an improved boosting classification method, by considering both rectangular features (Viola and Jones 2001b), and control points (Abramson et al. 2007) within Adaboost, combined with several meta-heuristic methods for the exploration of the search space of the visual features. The general architecture of our proposed model is illustrated in Fig. 6. As input we dispose of a learning database containing positive and negative examples. Adaboost uses the database by applying a sliding window on the images, with the visual features to extract the best weak learners. In each iteration it uses a meta-heuristic algorithm to explore the search space. The output of I-SwarmBoost is a ROI (region of interest) that is plotted in the image, to identify the detected object.

Each category of the visual features was combined with a meta-heuristic algorithm. For clarity each optimization method will be presented in a separate section.

4 Description of the meta-heuristics boosted weak learners

The application case of this study is the detection of cars for helping the autonomous driving system to make decisions by detecting obstacles. The project has been done at Paris robotics center (CAOR), with the aim of evaluating a precedent work in this laboratory (Abramson et al. 2006), and for proposing improvements by developing and implementing other types of weak learners. The obstacles here are in-front cars, through a view of their backside. To that aim we used a learning database (see Fig. 7) in the input of the algorithm, containing positive examples (5000 car images), and negative examples (2500 non cars images). To ensure diversity in the database, we captured the images in different lighting conditions, resolutions, and positions in the road. Note that the detection system could be generalized on other objects, it is only necessary to create positive examples on the class of these objects (human faces, pedestrians, bikes, road signs,...). The video stream should be analyzed in real time to produce a ROI on the positive detected object, as in Fig. 8.

Fig. 7 The learning database used for training Adaboost classifiers. Top: positive examples of cars images, Bottom: negative examples that are other objects than cars

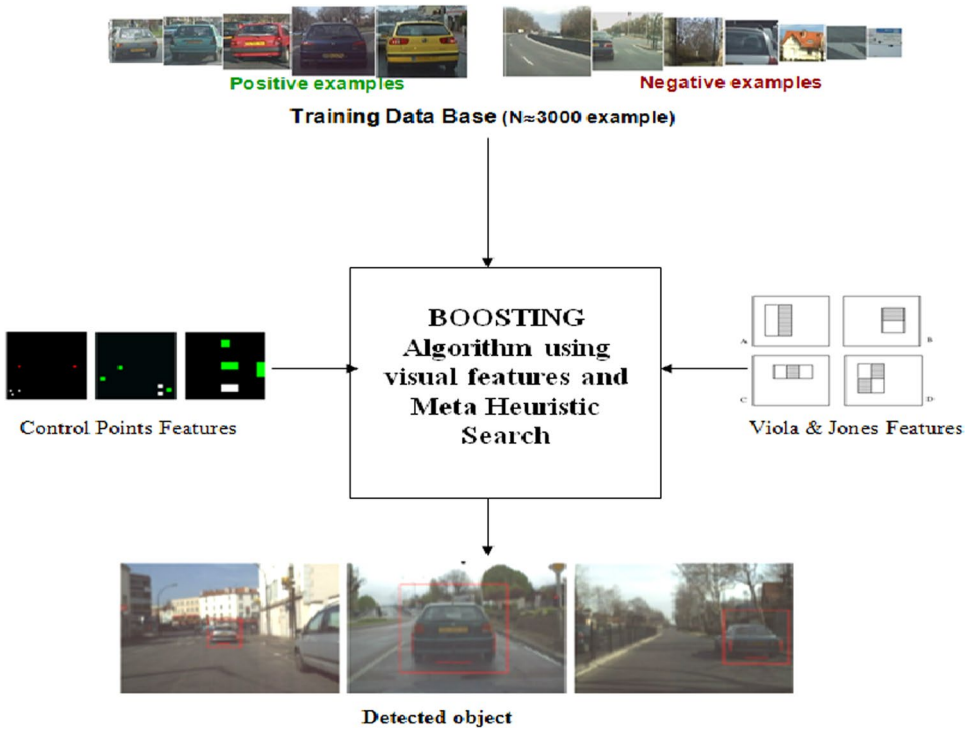

Fig. 6 An overview of the proposed framework. An input database serves Adaboost with learning examples. Meta-heuristic algorithms are then used to explore the search space of the most discriminative visual features

Fig. 8 A region of interest (ROI) is drawn on the image for identifying the positive detected object (car)

Adaboost algorithm tries to identify in each iteration, the best features that are the most discriminative. This procedure produces a weak-learner. In the case of Adaboost, the weak-learner is an algorithm that explores the space of the visual features. We propose here the evaluation of some exploration methods, using meta-heuristics and intelligent swarm optimization. In the literature, we can have a lot of techniques that were proposed for optimization issues. We advise reading this excellent book for a deep review study (Talbi 2009). The techniques can be classified as either local (typically gradient-based) or global (typically non-gradient based or evolutionary) algorithms (Venter 2010). Among the existing techniques there are: gradient descent (GD) (Bottou 2012), Ant colony (Hajjem et al. 2017), genetic algorithms (GA) (Goldberg 1989; Oliver 2017), evolutionary hill climbing (Abramson et al. 2006), particle swarm optimization (PSO) (Poli et al. 2007), taboo search (Glover and Laguna 1997). All these optimization algorithm can be used to find the values of parameters (coefficients) of a function (*f*) that minimizes a cost function (*cost*). They can be best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

Control points and rectangular features were evaluated with only one optimization algorithm, which is Hill Climbing as explained in Abramson et al. (2006). In our paper, we propose a continuity of the precedents authors' work (Abramson et al. 2006), by proposing an exhaustive scheme for the evaluation of other optimization algorithm to discover weak learners with the aim of detection's rate improvement.

The evaluation protocol that we propose here, involves the comparison of five strategies:

- Evolutionary hill climbing (Evo-HC-BOOST): we evaluate here the model proposed in Abramson et al. (2006) by fine tuning the parameters of their algorithm. This type of algorithm is used to remedy with the problem of total exploration, and it uses a principle close to genetic algorithms that is to say it begins with a population of features, select the best elements among theme, i.e., those that reduce the classification error, and finally makes mutations between these best elements.
- Random walk (RW-BOOST): in this case we developed a purely random search that is made in the features' space with the Max Trials parameter (number of iterations and features) and then the best features found are refined by successive mutations.
- Particle swarm optimization (PSO-BOOST): in this algorithm, we developed and implemented a new way technique for the exploration of the search space using bio-inspired intelligent methods.
- Hybrid (H-BOOST): it combines both genetic algorithms and PSO in a global model.
- Full search (FS): it is expected to perform an exhaustive exploration via Adaboost, of all possible rectangular and control points features, with different scales of the input image. It is the most computationally expansive.

Each method has been evaluated regarding the intrinsic parameters (tuning), time processing during the learning, ROC and Precision/Recall score for the validation. They are presented in the same order in the next subsections.

4.1 Genetic algorithms and evolutionary hill climbing-boosting classifiers: Evo-HC-BOOST

The objective here is to select in each iteration of Adaboost, a simple weak-learner that classifies better than chance for a given distribution of examples. The selection is made here using genetic algorithms, whose common characteristics are to be based on the manipulation of an artificial population, that are elements of a search space (Potential solutions to the problem). Figure 9 illustrates the pseudo-code of a GA

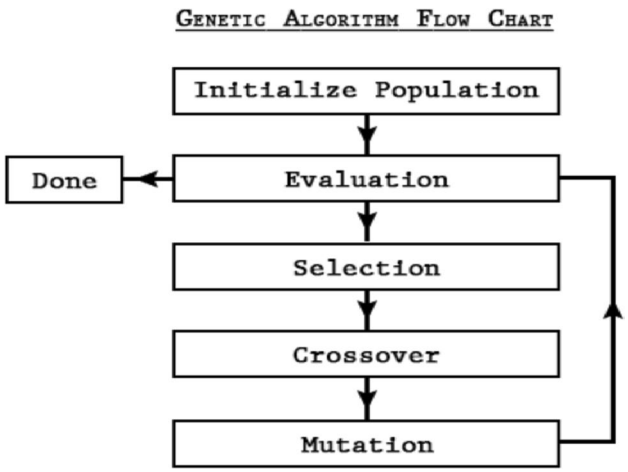

Fig. 9 Genetic algorithm simple flow chart

algorithm. The evolution of the population is simulated with some operations that can be repeated in a loop, called generations, until the optimization converges. If the parameters are correctly calibrated, than the convergence to a solution can be obtained with small number of iterations.

As it is explained in algorithm 2 pseudo-code, the hill climbing concept consists of allowing a succession of mutations if the learning error decreases.

ALGORITHM 2: Genetic Algorithm Hill Climbing optimization pseudo-code (Evo-HC)

Ensure: h a weak-learner which has the smallest error within the examined features.
Begin
Let $\epsilon(h_i)$ be the weighted classification error for each feature h_i.
Let $N = N_{best} + N_{rand}$ the total size of the population of features.
Initialize a random population of visual features h_i^1, with $i = 1 \rightarrow n$
Initialize $errMin \leftarrow +\infty$, and $noImpr \leftarrow 0$
for g in $1 \rightarrow N_{generation}$ **do**
  Calculate $\epsilon(h_i^g)$ the classification error of each feature h_i^g in the considered generation
  Rank the feature h_i with $Ascendant(\epsilon(h_i^g))$
  Select h_j^g best features, with $j = 1 \rightarrow N_{best}$
  if $\epsilon(h_j^g) \leq errMin$ **then**
    $\epsilon(h_j^g) \leftarrow errMin$ and $noImpr \leftarrow 0$
  else
    $noImpr++$
  end if
  if $noImpr \geq N_{generation}$ **then**
    return h_j^g
  end if
  for k in $1 \rightarrow N_{mut}$ **do**
    $\overline{h_j^g} \leftarrow Mutate_k(h_j^g)$
    if $\epsilon(\overline{h_j^g}) \leq \epsilon(h_j^g)$ **then**
      $h_j^g \leftarrow \overline{h_j^g}$
    end if
  end for
  Generate N_{Rand} new features
end for
End

The algorithm starts by generating for a given number of generations, random population of features on the input image. The error is calculated on each feature $\epsilon(h_i^g)$, and best features with low error are selected for mutation. The mutation operator aims at diversifying the population and avoiding exploring around local minimum. But the most important parameter in Evo-HC, is the hill climbing parameter (*noImpr*). Conversely to gradient descent, it allows exploring the research space even though the gradient increases.

$Evo - HC$ algorithm has fours important parameters: $N_{generation}$ which represents the number of generations of the genetic algorithm, N_{best} representing the best selected features, N_{mut} the number of mutations applied on N_{best} visual features, and N_{Rand} the randomly added features. Our goal here is to evaluate the impact of each of these parameters on Evo-HC algorithm and to find the best tuning regarding the detection rate with our cars learning database. At each time we varied a parameter by keeping unvaried the remaining parameters, and then we calculated the ROC (Receiver Operating Characteristic), the Precision-Recall curves, and also the learning processing time. Results are illustrated in Sect. 5.1.

4.2 Random walk-boosting classifiers: RW-BOOST

Random search (or select and refine) weak-learner is the second proposed algorithm that is combined with Adaboost in our study for object detection purposes. The general principle of this algorithm as its name suggests is based on the generation of a huge number of random features (*MaxTrials*) from which are retained the bests, then a succession of mutations (*MaxMut*) are applied only for these selected features. Hence, this algorithm has two parameters that we would like to evaluate: *MaxTrials* and *MaxMut*. Algorithm 3 gives an overview of the RW-BOOST pseudo-code. Following the same scheme of evaluation, we will show the impact of each parameter in Sect. 5.2.

ALGORITHM 3: Random Walk-Boosting pseudo-code (RW-BOOST)

Ensure: h a weak-learner which has the smallest error within the examined features.
Begin
Let $\epsilon(h_i)$ be the weighted classification error for each feature h_i.
Initialize a random population of visual features h_i^1, with $i = 1 \rightarrow n$
Read $MaxMut$, and $MaxTrials$.
Initialize $errMin \leftarrow +\infty$
for g in $1 \rightarrow MaxTrials$ **do**
  Calculate $\epsilon(h_i^g)$ the classification error of each feature h_i^g
  Rank the feature h_i with $Ascendant(\epsilon(h_i^g))$
  Select top h_j^g best features, with $j = 1 \rightarrow NMaxMut$
  for $j = 1 \rightarrow NMaxMut$ **do**
    $\overline{h_j^g} \leftarrow Mutate_k(h_j^g)$
    if $\epsilon(\overline{h_j^g}) \leq errMin$ **then**
      $h_j^g \leftarrow \overline{h_j^g}$
      return h_j^g
    end if
  end for
end for
End

4.3 Particle swarm optimization-boosting classifiers: PSO-BOOST

In this subsection, we introduce a novel weak-learner that we have developed and implemented using boosting paradigm, and which belongs to meta-heuristic techniques. More precisely, it is based on particle swarm optimization (PSO) family of algorithms (Pai and Michel 2017; Tavares et al. 2017).

Definitions Meta-heuristics are of family of deep optimization algorithms aimed at solving difficult problems arising from operational research for which no more conventional method is known to be more effective. Appeared from early 80's of the last century, metaheuristics are generally iterative stochastic algorithms, which progress towards an optimum by sampling an objective function. The metaheuristics methods, try to find the global optimum of a difficult optimization problem (with possible discontinuities) without being trapped by the local optima. They manipulate one or more solutions, when searching for the optimum that is the best solution to the problem. The successive iterations must make possible to pass from a solution of poor quality to the optimal solution. Usually, the algorithm stops after reaching a

stop criterion, which consists in reaching the execution time or in a requested precision.

Characteristics Each metaheuristic algorithm has set of characteristics:

- They do not require necessary special knowledge of the optimized problem to operate.
- They are not necessarily used for problems that cannot be optimized by mathematical methods.
- They are often used in combinatorial optimization.

In a general manner, metaheuristics do articulate around three main notions:

- Diversification: It refers to the process of gathering information about the optimized problem.
- Intensification/exploration: It aims at using the information already gathered to define and navigate in the areas of interest of the research space.
- Memory and learning: It is the support of learning, which allows the algorithm to take into account only the areas where the global optimum is likely to be, thus avoiding the local optima.

Metaheuristics progress in an iterative way, alternating at each step the phases of intensification, diversification and learning, or by combining these notions more closely. The initial state is often chosen randomly, the algorithm then proceeds until a stop criterion is reached. Figure 10 illustrates the process of diversification and intensification.

Metaheuristics are often inspired by natural systems, whether they are taken in physics (simulated annealing), or in biology (case of ant colony algorithms or optimization by particle swarms).

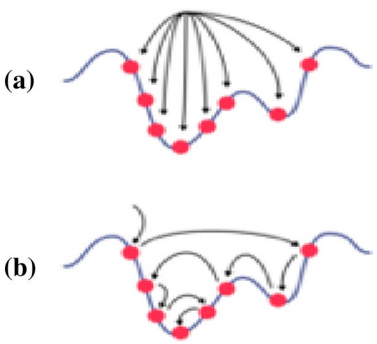

Fig. 10 The diversification (**a**) and intensification (**b**) notions of a metaheuristics algorithm

Swarm intelligence Particle swarm optimization (PSO) is a metaheuristic of optimization, invented by Russel Eberhart (electrical engineer) and James Kennedy (socio-psychologist) in 1995. The guiding idea for the authors was the simulation of Collective behavior of birds inside a cloud (Kennedy et al. 2001). Initially J. Kennedy and R. Eberhart sought to simulate the ability of birds to fly synchronously and their ability to change direction abruptly while remaining in optimal formation. The model they proposed was then extended into a simple and efficient optimization algorithm. The particles are individuals that move into the hyper-space of research. The exploration process is based on two rules: (i) Each particle has a memory that allows it to memorize the best point by which it has already passed and tends to return at this point (called *Pbest* for personal best). (ii) Each particle is informed of the best known point in its neighborhood and it will also tend to go towards this group best *Gbest* point.

Neighborhood The neighborhood of the particles is the structure of the biological social network. The particles within a neighborhood communicate to each other. Different neighborhoods have been studied in the literature.

- Star topology (Fig. 11middle): the social network is complete, each particle is attracted to the best group particle noted *gbest*, and communicates with the others. It is this topology that we have chosen for our PSO-BOOST WeakLearner.
- Ring topology (Fig. 11left): each particle communicates with n immediate neighbors, and tends to move towards the best position in the local neighborhood called *lbest*.
- Radius topology (Fig. 11right): a central particle is connected to all others. Only this central particle adjusts its position towards the better, if this causes an improvement the information is propagated to the others.

PSO algorithm Each particle represents a potential solution in the features space search. The new position of a particle is determined according to its own value and that of its neighbors. Let $\overrightarrow{x_i(t)}$ be the current position of a particle P_i at time t. This position is modified by adding a speed vector

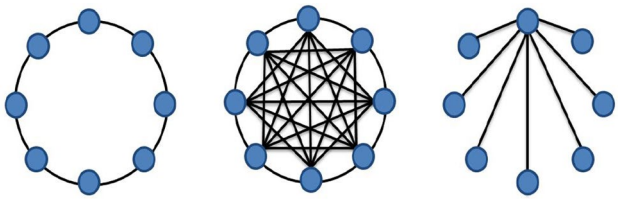

Fig. 11 The different topologies of particle swarm algorithms. (left): ring topology, (middle) star topology, (right) radius topology

(velocity) $\overrightarrow{v_i(t)}$ to its past position: $\overrightarrow{x_i(t)} = \overrightarrow{x_i(t-1)} + \overrightarrow{v_i(t)}$. It is the velocity vector that directs the research process and reflects the "sociability" of the particles. If N particles are considered and each particle compares its new position to its best obtained position, this gives the simple PSO algorithm as in 4, where *Fitness* is the fitness function.

ALGORITHM 4: Simple Particle Swarm pseudo-code

```
Let N be the number of particles.
Let x_i be the position of particle P_i.
Let v_i be the velocity of particle P_i.
Let Pbest_i be the best fitness of the particle P_i.
Let x_Pbesti be the best position of particle P_i.
Let ρ be a random positive value.
Begin
repeat
  for i in 1 to N do
    if Fitness(x_i) ≥ Pbest_i then
      Pbest_i ← x_i
      x_Pbesti ← x_i
    end if
    v_i ← v_i + ρ(x_Pbesti - x_i)
    x_i ← x_i + v_i
  end for
until The process converges
End
```

This first algorithm does not take into account the neighborhood structure and the topology of the swarm network, since it uses only the improvement obtained on the particle itself. If we consider a star neighborhood, the Algorithm 4 becomes 5. This is the one we have implemented, and tested with Adaboost for object detection.

ALGORITHM 5: Star-Topology Particle Swarm pseudo-code

```
Let N be the number of particles = Popsize.
Let x_i be the position of particle P_i.
Let v_i be the velocity of particle P_i.
Let Pbest_i be the best fitness of the particle P_i.
Let x_Pbesti be the best position of particle P_i.
Let x_Gbesti be the best position of in the group.
Let ρ_1, ρ_2 be two random positive values.
Begin
repeat
  for i in 1 to N do
    if Fitness(x_i) ≥ Pbest_i then
      Pbest_i ← Fitness(x_i)
      x_Pbesti ← x_i
    end if
    if Fitness(x_i) ≥ Gbest_i then
      Gbest_i ← Fitness(x_i)
      x_Gbesti ← x_i
    end if
  end for
  for i in 1 to N do
    v_i ← v_i + ρ_1(x_Pbesti - x_i) + ρ_2(x_Gbesti - x_i)
    x_i ← x_i + v_i
  end for
until The process converges or N greater than Epoch
End
```

This algorithm starts by generating a N particle on random positions $\overrightarrow{x_i}$ in the search space. Then each particle (which is a visual feature in our case) is evaluated through a fitness function. At each step the best precedent personal fitness as well as the group best fitness are memorized. The more a particle is far from the best position in the swarm, the greater will be the variation of its speed in order to move towards the best solutions. The random variables in this algorithm ρ_1 and ρ_2 represent the compromise between the inertia of the particle, its nostalgia, and its imitation of others. They are chosen rigorously to have a convergence and can be: $\rho_1 = r_1 c_1$, $\rho_2 = r_2 c_2$, with r_1 and r_2 follow a uniform distribution in [0,1] and c_1, c_2 are constants representing a positive acceleration. The algorithm executes as long as a convergence criterion has not been reached. In our case, this can be done by three ways:

- Reaching a certain number of iterations, called in our case *Epoch*.
- The fitness (optimization function) reaches an acceptable value.
- The velocity variation tends to 0.

Implementation In this paper, each particle represents a visual feature (control point or rectangular feature). As reported antecedently, we followed a start topology, and the different parameters that we want to evaluate here are:

- The number of weak learners.
- The number of iteration of the algorithm *Epoch*.
- The initial number of randomly generated particles *Popsize*)
- Features' resolutions in the images (full, half, quarter).

The evaluation protocol is the same as in the precedent methods. Results are presented in Sect. 5.3.

5 Experimental evaluation

We present here the evaluation results on the entire previously described algorithms. Recall that one important parameter of Adaboost is the number of the weak-learners, which are generated at the end of the learning process. The more important is the number of the weak learners, the more detection rate increases. This observation is illustrated and asserted in Figs. 12 and 13, where in the first picture we can observe decreasing error rates are visible with increasing number of weak learners, whereas in the second figure we can observe increasing detection rates with high number of weak learners. After a series of experiences we obtained good results with 500 weak-learners for the majority of the algorithms.

5.1 Evo-HC-BOOST evaluation

Here we present the evaluation of Evo-HC-BOOST algorithm. Table 1 illustrates the values of the different parameters that have been chosen during the tests with a certain

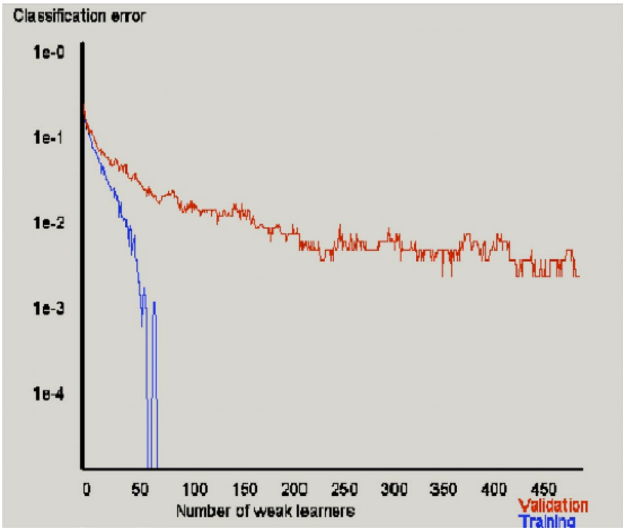

Fig. 12 Variation of training and validation error when varying the number of weak learners. The more number of weak learners increases, the more error decreases

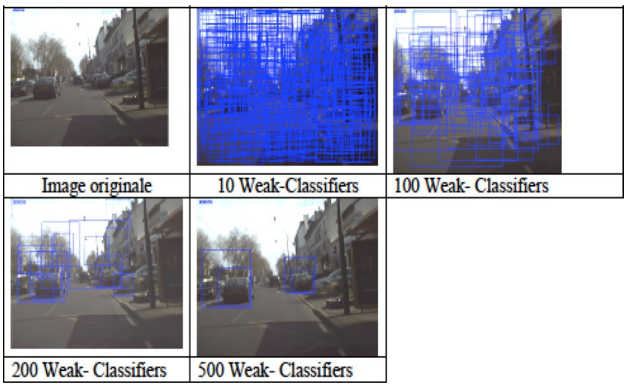

Fig. 13 The more important is the number of weak learners, the more detection rate increases

manner for allowing the exploration of each parameter separately.

The first observation that we can made concerns time processing. Greater values are observed when the product $N_{best} \times N_{mut}$ increases. For example the couples (45,80), (45,50), (30,100) generated about 58, 41 and 31 h respectively. Obviously, important successive mutation operations on best features are very time consuming (see Fig. 14 for more details).

To go deeper in our analysis, we evaluated each combination's classifier with a ground truth database. Figure 15 depicts the variation of the precision/recall curves within the previously used parameters of Evo-HC-BOOST. For instance, we can see that the good classifiers were generated using great mutation values (eg:

Table 1 Evo-HC-BOOST parameters evaluation. Combinations are ranked regarding time consumed during the learning process

N_{best}	N_{rand}	N_{mut}	N_{gen}	Time processing
45	25	80	50	58h01
45	25	50	50	41h09
30	40	100	30	31h32
30	40	40	40	15h02
30	40	20	40	11h01
30	40	10	80	10h26
5	65	10	30	10h02
30	40	20	30	8h25
30	40	10	30	5h15
30	100	10	30	5h19
30	60	10	30	5h00
30	50	10	30	4h52
30	0	10	30	4h41
30	40	10	30	4h37
30	30	10	30	4h23
30	40	10	20	3h17
30	40	05	30	2h52
30	40	10	10	2h18

*EVO-HC*_45*best*25*rand*80*mut*50*gen* classifier). as a further analysis we will elaborate evaluations on each parameter separately, in order to have a good calibration of Evo-HC-BOOST.

Analyzing the effect of the mutation operator Here we present the evaluation of the mutation effect that is applied on the visual features. More precisely, the implemented mutation operator can either move a feature on the image with a specific radius (from 1 to 5 pixels), add randomly a feature at a random position in the image, delete an existing feature, change the resolution of the image, or change the gray-scale of the image (see Fig. 16). We decided to fix the remaining parameters of EVO-HC-BOOST and modify only N_{mut} with a range in (10,20,40 and 100) as it is illustrated in Table 2. For each tuple of values we calculated the AUC (area under curve) with the ground truth. Firstly, we can confirm past observations about time processing that raises with large mutations. Secondly, we observed that modifying N_{mut} from 10 to 40 allowed the augmentation of AUC of the generated weak learners, with a max value of 0.95 observed for $N_{mut} = 40$. However, we also observed that with $N_{mut} \geq 100$, the AUC starts diminishing, which could be an over-learning effect or a well local minimum. Consequently, it is not recommended to augment indefinitely the size of N_{mut} since it is in one side very time consuming, and in the other side can bring to over-fitting.

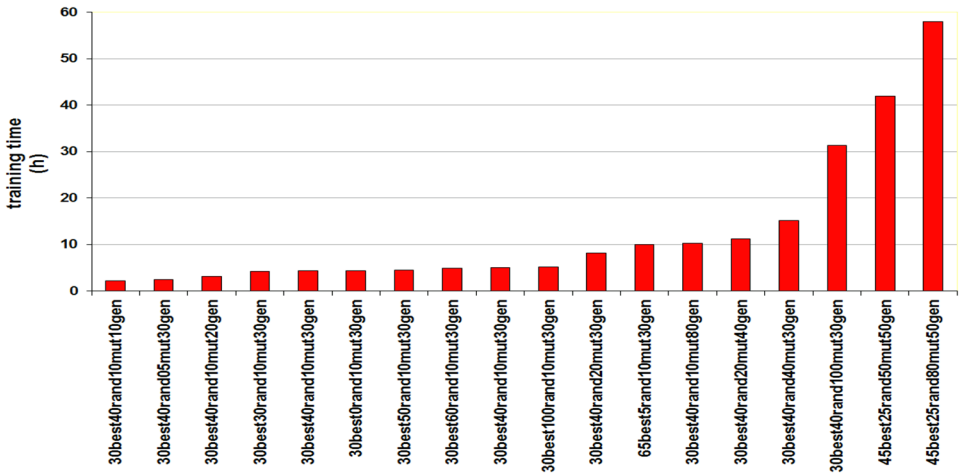

Fig. 14 Variation of the processing time (in hours) during the learning of EVO-HC-BOOST. Important time values are observed within greater number of mutations that are applied on the best features

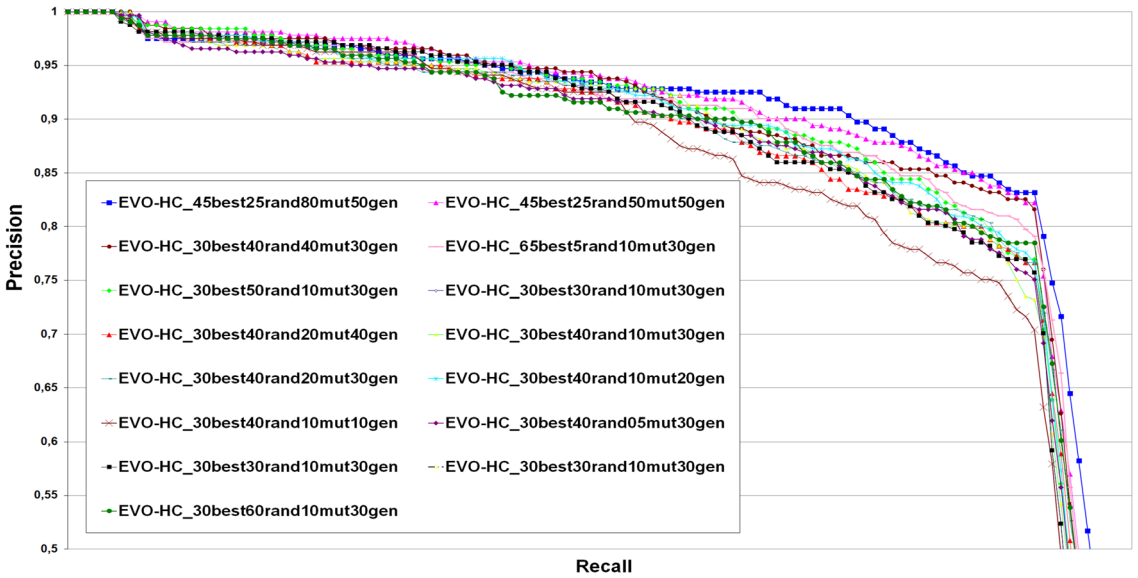

Fig. 15 Precision Recall curves of different parameters of EVO-HC-BOOST algorithm

Table 2 Evaluation of the mutation parameter of EVO-HC-BOOST

N_{best}	N_{rand}	N_{mut}	N_{gen}	Time proc.	AUC
30	40	**10**	30	4h37	0.90
30	40	**20**	30	11h02	0.92
30	40	**40**	30	31h02	**0.95**
30	40	**100**	30	31h32	0.91

Bold values represent the parameter which is still unchanged during the evaluation

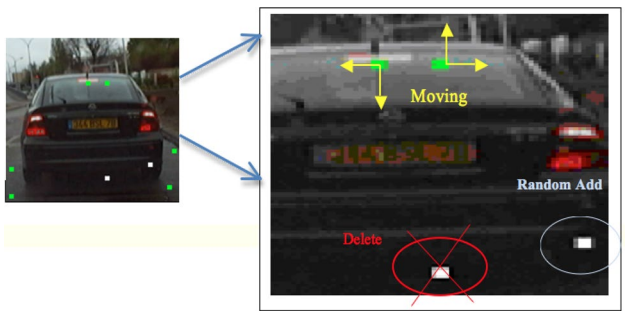

Fig. 16 Example of some mutation operators with control point features in EVO-HC-BOOST algorithm

Analyzing the effect of the random operator Here we present the evaluation of the random effect that is applied on the visual features. This parameter represents the number of randomly added individuals (visual features) by the genetic algorithm during the training. It is very important since it allows the expansion of the research space, and avoiding the local optimum.

Table 3 Evaluation of the random parameter of EVO-HC-BOOST

N_{best}	N_{rand}	N_{mut}	N_{gen}	Time proc.	AUC
30	**0**	10	30	4h41	0.91
30	**30**	10	30	4h23	0.93
30	**40**	10	30	4h37	0.93
30	**50**	10	30	4h52	0.91
30	**60**	10	30	5h00	0.90

Bold values represent the parameter which is still unchanged during the evaluation

Table 4 Evaluation of the number of generations parameter of EVO-HC-BOOST

N_{best}	N_{rand}	N_{mut}	N_{gen}	Time proc.	AUC
30	40	10	**10**	2h18	0.80
30	40	10	**20**	3h17	0.9
30	40	10	**30**	4h37	0.93
30	40	10	**80**	4h37	0.94

Bold values represent the parameter which is still unchanged during the evaluation

In a similar way, we have varied values of N_{rand}, fixed the rest of parameters, and calculated AUC of each tuple as illustrated in Table 3. The first observation concerns time processing which is varying in a nonlinear way. Secondly, the real impact of this parameter on the AUC is unforeseeable regarding the tested detectors since for $N_{rand} = 0$ and 50 the AUC is practically the same which can be hardly interpreted.

Analyzing the effect of the number of generations operator Here we present the evaluation of the number of generation effect that is applied on the visual features. This parameter represents the stop criteria of the algorithm, that is to say, the number of successive generations without reducing the minimum error of the population. By always keeping the same principle for the other parameters, we did the realizations depicted in Table 4. The first remark that we can make concerns the time processing which is exponentially related to the number of generations. Secondly, large size of generations allow having great AUC values. Likewise N_{mut}, N_{gen} seems to be a good parameter for the search space global exploration.

Discussions Throughout these experiments on the EVO-HC algorithm, we had some intermediate results on the effect of each parameter:

- For the N_{rand}: it seems useless to choose a large value.
- For For N_{mut}: seems very effective at least for values between 40 and 60.
- For the N_{gen}: seems also effective when exceeding 20 generations.

It seems however that none of these varied parameters has an absolutely systematic impact. This may be an effect hidden by the random variations during the training, and/or the over-fitting problem. It may also be necessary to ideally varying certain parameters together (for example, changing the N_{mut} and N_{gen} at the same time). However, despite these difficulties, we have generated very good detectors up to 95% of detection rate. These results are visible on the ROC curves of Fig. 17 in which can be seen the new best combination detectors: (N_{best}, N_{rand}, N_{mut}, N_{gen}) = (30, 40, 40, 30) and (45, 25, 50, 50).

5.2 RW-BOOST evaluation

There are two parameters of RW-BOOST that we want to evaluate, namely: *MaxTrials* for the number of iterations, and *MaxMut* for the number of mutations. Table 5 gives

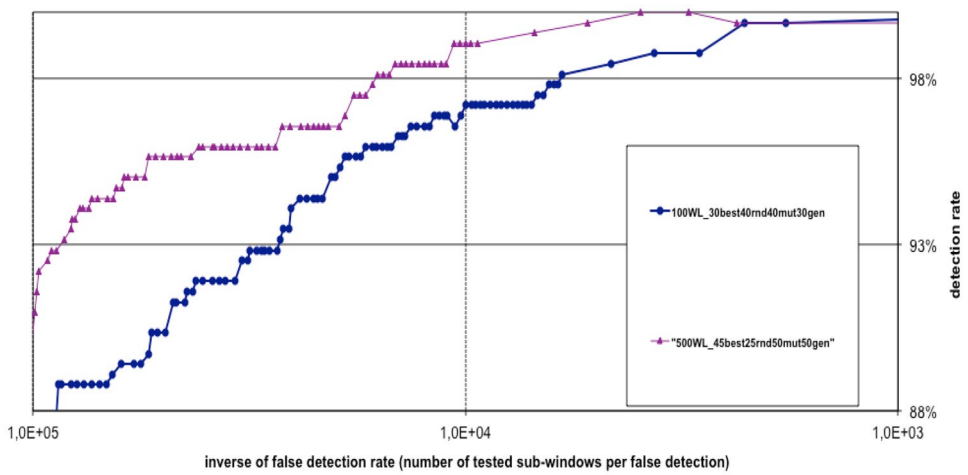

Fig. 17 Best combination of parameters during the training of EVO-HC-BOOST algorithm

Table 5 RW-BOOST parameters evaluation

Max trials	Max mutations	Training time
30,000	3000	2h24
30,000	10,000	3h16
30,000	30,000	5h49
30,000	100,000	34h15
100,000	3000	5h24
100,000	10,000	6h12
100,000	30,000	8h20
100,000	100,000	18h29
300,000	3000	13h17
300,000	10,000	13h53
300,000	30,000	16h41
300,000	100,000	25h46

the series of the couples of values which were associated to these parameters.

Analyzing the effect of MaxMut parameter As in the previous algorithm, it seems that augmenting the number of Max Mutations from 3000 to 100,000 tends to increase the time of training. The second observation, concerns the performance of the generated classifiers. Indeed, as it is illustrated in Fig. 18, it seems that augmenting MaxMut from 3000 to 10,000, automatically impacts the detection rate and it is reflected on the ROC curves, in which the classifier with MaxTrial = 300,000, and MaxMut = 100,000 has obtained the best detection rate.

Analyzing the effect of MaxTrials parameter In a reciprocal manner, to analyze the effect of *MaxTrials* parameter, we fixed *MaxMut* at each step, and varied *MaxTrials* from 30,000 to 300,000. Concerning the impact of this parameter on the learning time, it can be seen in Fig. 19 that it is quite considerable (exponential).

Concerning the impact of *MaxTrials* on the detection rate, it can be observed in Fig. 20, that ROC curves are overlapping, with a fixed value of *MaxMut* = 3000. Typically, this can be explained as an over-fitting during the learning process.

Discussions To finalize the study of the RandomSearch WeakLearner, a small comparison with the results obtained in the previous section with the EVO-HC-BOOST was

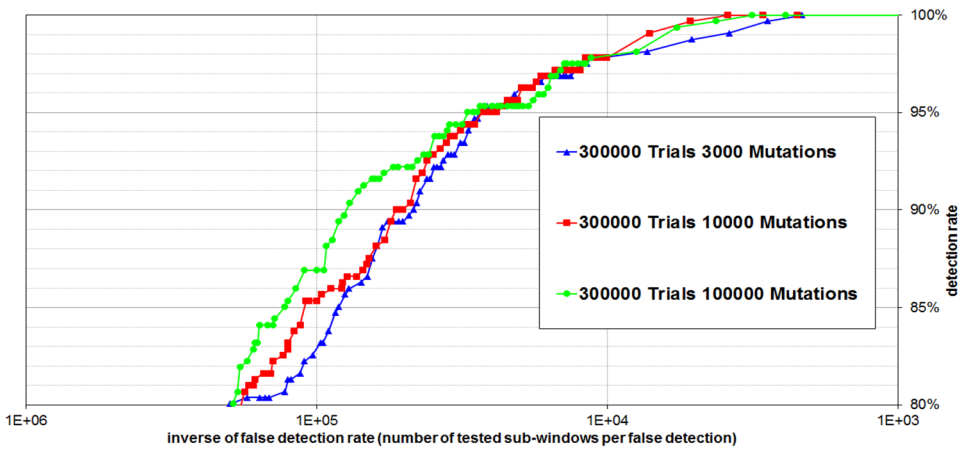

Fig. 18 Analyzing the impact of MaxMut parameter, through ROC curves during the training of RW-BOOST algorithm. MaxTrials parameter is fixed on 300000 whereas MaxMut varies from 3000 to 100,000

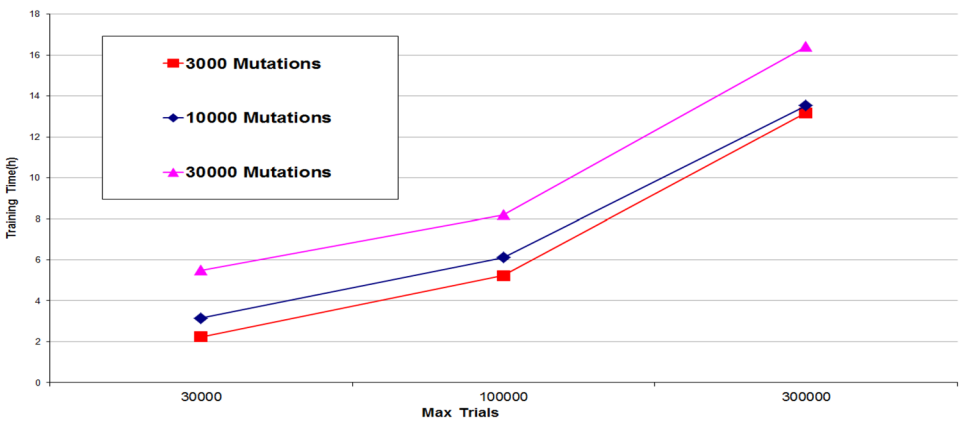

Fig. 19 Exponential training time variation was observed when varying the number of Maxtrials (from 30,000 to 300,000) of RW-BOOST algorithm

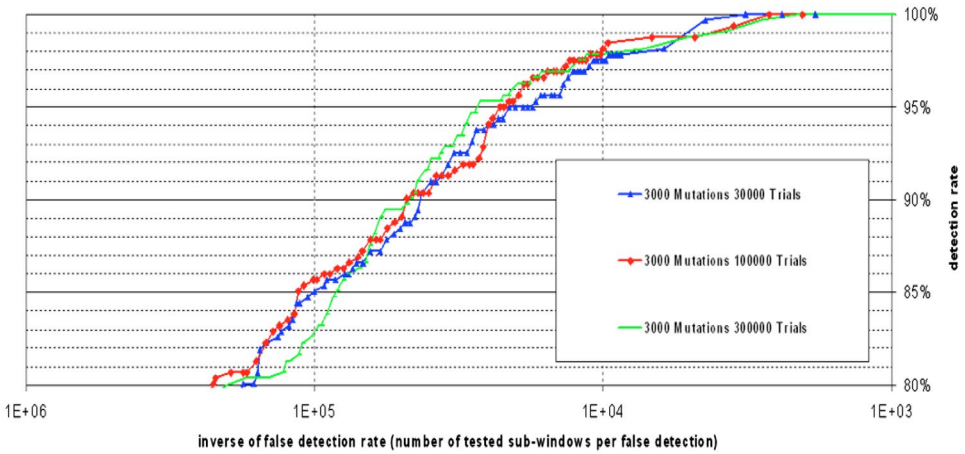

Fig. 20 Analyzing the impact of MaxTrial parameter, through ROC curves during the training of RW-BOOST algorithm. MaxMut parameter is fixed on 3000 whereas MaxTrials varies from 30,000 to 300,000

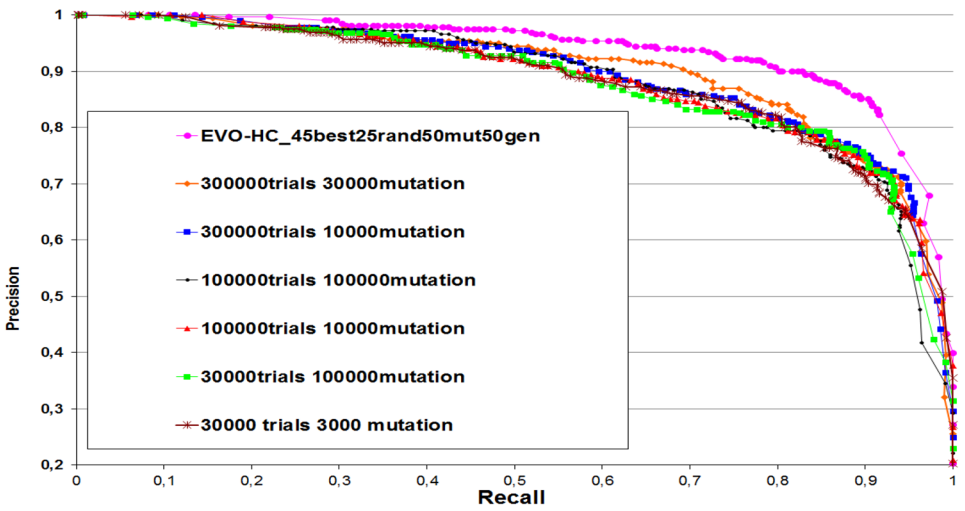

Fig. 21 Comparison between EVO-HC-BOOST and RW-BOOST

made. Results are presented in precision recall curves of Fig. 21. They show that even for high values of *MaxTrials* (300,000) and *MaxMut* (100,000) (With 25h46 of learning time) the detection performance remains very low compared with those obtained with the EVO-HC. We can see that the considerable discrepancy between the best classifier obtained with the EVO-HC (45*best*25*rand*50*mut*50*gen*) and those of the Random Search.

5.3 PSO-BOOST evaluation

Analyzing the effect of the number of the weak learners Evaluating this algorithm involves firstly checking the impact of the number of utilized weak learners on the global detection rate. Figure 22 illustrates different detectors with 100, 200, and 500 weak-learners. We can observe that this later is the most efficient classifier in term of true positives in the ROC curve. For information, the average convergence time of all PSO-BOOST classifiers tested here was up to 1h30h which is faster compared to the previous methods.

Analyzing the effect of the swarm population size For evaluating the swarm population, we changed *Popsize* regarding these values: 40, 45, 60. Results in ROC curves of Fig. 23 shows that for Popsize=60, we obtained the best classification rate. Consequently, with large swarm population the search space is explored more efficiently.

Comparison between Evo-HC and PSO-BOOST To enhance our analysis, we decided to compare the three precedent algorithms: EVO-HC, RW and PSO-BOOST. For each algorithm we kept the best combination of parameters that we discovered and discussed previously. The resulting ROC curves of this comparison are given in Fig. 24. We can see that Swarm-BOOST outperform the rest of classifiers.

Discussions In the three algorithms discussed and evaluated above, one of the key difference resides is in the mechanism of generating and producing a new population of solutions via the modification of the obtained solutions from the past populations. The three described methods generate

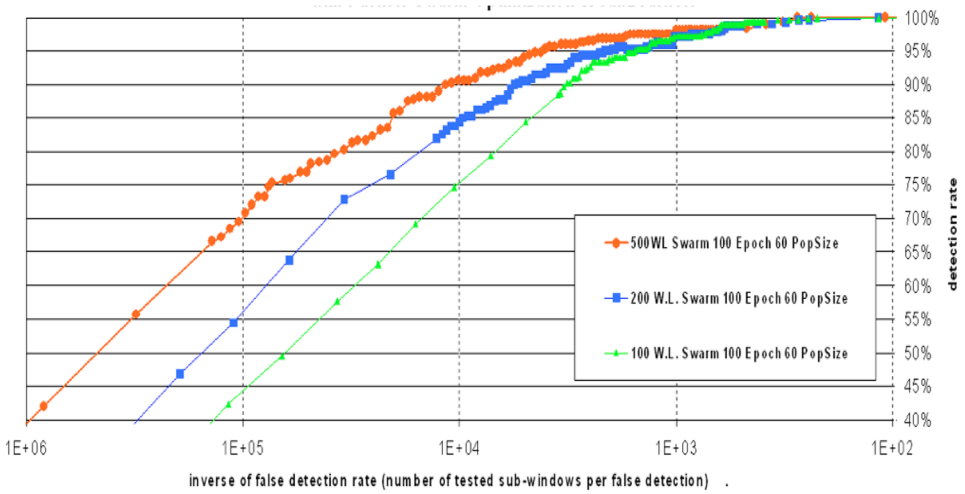

Fig. 22 ROC curves with multiple weak learners of PSO-BOOST algorithm. Using high number of weak learners the system generates good classifier

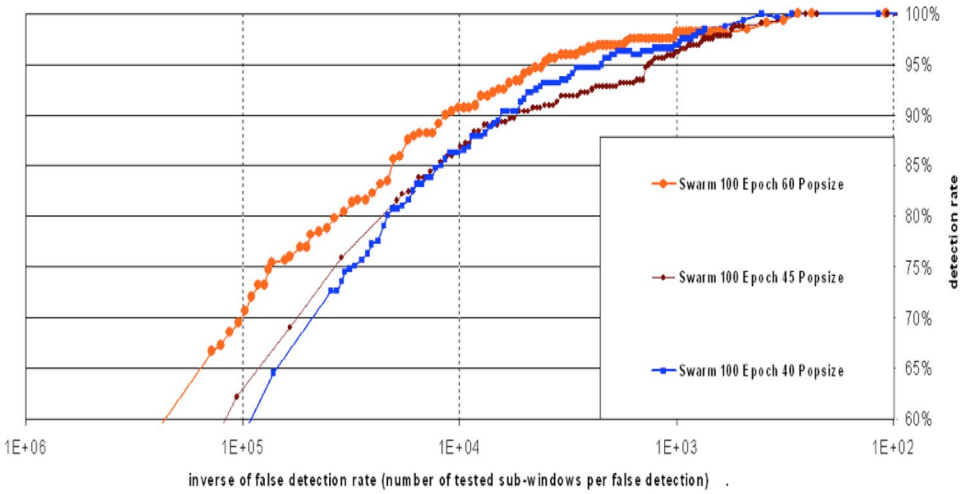

Fig. 23 ROC curves with different swarm population size of PSO-BOOST algorithm. Using great Popsize the system generates good classifier

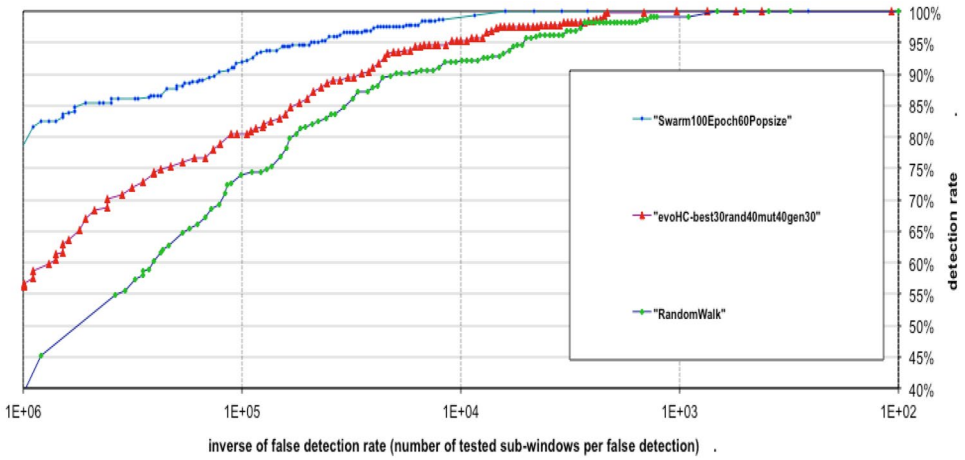

Fig. 24 ROC curves of the comparison between EVO-HC-, RW- and PSO-BOOST algorithms. Results show that swarm-based detector outperforms both genetic-based and random walk algorithms

a population of visual features with different equilibrium between intensification and diversification. This dynamic behavior of the population can be deducted from the basic perturbation method used in the creation of new solutions. These facts were observed via the precedent results that have shown good performances of PSO compared to RW

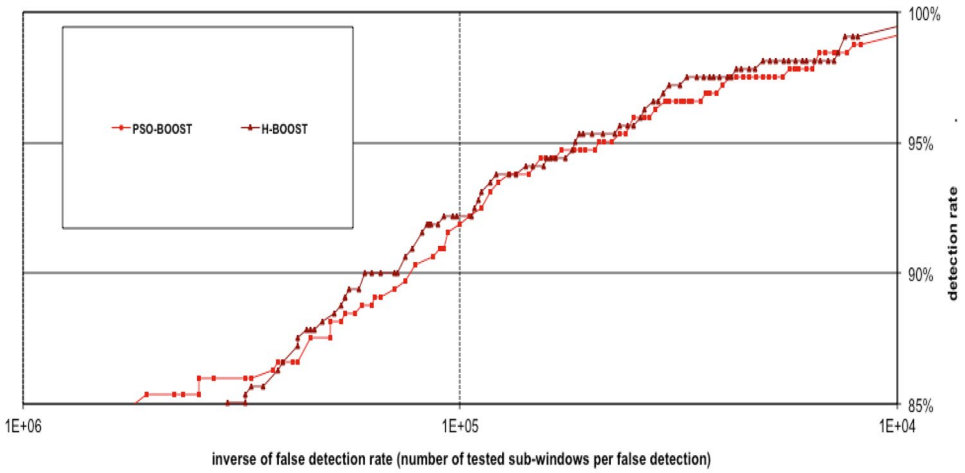

Fig. 25 Comparison of the ROC curves obtained with PSO-BOOST and H-BOOST (hybrid) algorithms

and EVO-HC. This can be interpreted by the intelligent way with which the swarm particles explore the search space instead of a "random walk". Moreover, the drawback of the EVO-HC and RW were in their expensive computational cost compared to PSO. This work has confirmed the previous claimed results in other optimization domains, in which PSO had the better effectiveness (finding the true global optimal solution) compared to GA, with significantly better computational efficiency (Sidhartha and Padhy 2008). Motivated by these results we have implemented an hybrid combination of PSO and EVO-HC by modifying the line number 17 of algorithm 5, in which we applied the previous mutation operator of EVO-HC, on each best obtained particle $\overrightarrow{x_i}$. Our intuitive hypothesis supposes that global best optimum could be in the neighborhood of each best particle $\overrightarrow{x_i}$. The result of H-BOOST (hybrid boosting) are presented in Fig. 25. They show thin improvement of the detection rate compared to the previously best PSO-BOOST classifier. Future exploration of H-BOOST could bring better detector compared to PSO and EVO-HC algorithms.

6 Discussions and conclusion

In this paper we have presented new boosting classifiers implemented with Adaboost algorithm, in which the weak learners are obtained using challenging meta-heuristics methods, for the aim of object detection and tracking. Firstly we have fine-tuned EVO-HC, an existing Adaboost classifier based on genetic algorithms by identifying its best parameters. Genetic algorithms method has been popular due to their intuitiveness, ease of implementation, and their ability to effectively solve highly non-linear, mixed integer optimization problems. Secondly, we have proposed a new classifier based on particle swarm optimization. Particles Swarms are inspired by the swarming or collaborative behavior of biological populations. The evaluation has shown good results compared to the one based on genetic algorithms. After that we have proposed a third classifier which combines both swarm intelligence and genetic algorithms, and it obtained the best classification rate compared to the other studied techniques. The approaches are supposed to find a solution to a given objective function but employ different strategies and computational effort. It was appropriate to compare their performance in term of detection rate using boosting classification. In fact, all techniques were rigorously evaluated, in term of processing time, detection rates (ROC and Precision/recall curves), parameters calibration,... .

The proposed framework detects objects in new images with an explicit more rapid way than the previous approaches.

As perspectives to this work, we would like to compare the boosting-based classifiers with other machine learning techniques such as deep learning (Goodfellow et al. 2016). Indeed, we can consider the use of the same visual features that can be considered as convolution filter, and implement multi resolution analysis in the neural networks. In a complementary way, we would like to evaluate the proposed algorithms in a full parallel way under big data platforms. Indeed, a parallel Adaboost implementation has been recently published (BoostingPL) in Weka (Indranil and Reddy 2012). Our aim is to check whether the learning process can be improved or not.

Compliance with Ethical Standards

Funding This work was funded by Ecoles des Mines.

Conflict of interest The author declares that there is no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by the author.

References

Abramson, Y., Freund, Y.: Semi-automatic visual learning (SEVILLE): tutorial on active learning for visual object recognition. In: Proc, CVPR (2005)

Abramson, Y., Moutarde, F., Stanciulescu, B., Steux, B.: Apprentissage de détecteurs visuels d'objets par dopage utilisant un algorithme hybride évolution-escalade. In: 8 conference francophone sur l'apprentissage automatique (CAp'2006), Tregastel, France (2006)

Abramson, Y., Steux, B., Ghorayeb, H.: Yet even faster (YEF) real-time object detection. IJISTA **2**(2/3), 102–112 (2007)

Abril, P.S., Plant, R.: The patent holder's dilemma: buy, sell, or troll? Commun. ACM **50**(1), 36–44 (2007)

Andreopoulos, A., Tsotsos, J.K.: 50 years of object recognition: directions forward. Comput. Vis. Image Underst. **117**(8), 827–891 (2013)

Antani, S., Kasturi, R., Jain, R.: A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. Pattern Recognit. **35**(4), 945–965 (2002)

Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer Inc, New York (2006)

Bottou, L.: Stochastic Gradient Descent Tricks, pp. 421–436. Springer, Berlin (2012)

Bradley, J.K., Schapire, R.E.: Filterboost: regression and classification on large datasets. In: Proceedings of the 20th international conference on neural information processing systems, NIPS'07, USA, pp. 185–192. Curran Associates Inc., New York (2007)

Brunelli, R., Poggio, T.: Face recognition: features versus templates. IEEE Trans. Pattern Anal. Mach. Intell. **15**(10), 1042–1052 (1993)

Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, Adaboost and Bregman distances. Mach. Learn. **48**(1–3), 253–285 (2002)

Conway, D., White, J.M.: Machine Learning for Hackers. O'Reilly Media, Sebastopol (2012)

Ehrenfeucht, A., Haussler, D.: A general lower bound on the number of examples needed for learning. Inf. Comput. **82**(3), 247–261 (1989)

Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res. **4**, 933–969 (2003)

Ghimire, D., Lee, J.: Geometric feature-based facial expression recognition in image sequences using multi-class Adaboost and support vector machines. Sensors **13**(6), 7714–7734 (2013). https://doi.org/10.3390/s130607714

Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn., pp. 1–443. Addison-Wesley Longman Publishing Co., Inc., Boston (1989). ISBN:0201157675

Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org

Hajjem, M., Bouziri, H., Talbi, E.-G., Mellouli, K.: Parallel ant colony optimization for evacuation planning. In: Bosman, P.A.N. (ed.) Genetic and evolutionary computation conference, Berlin, Germany, July 15–19, 2017, Companion Material Proceedings, pp. 51–52. ACM (2017)

Ho, J., Yang, M.-H., Lim, J., Lee, K.-C.,Kriegman, D.: Clustering appearances of objects under varying illumination conditions. In: 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. Proceedings, vol. 1, pp. I-11–I-18 (2003)

Kramer, O.: Genetic Algorithm Essentials, Volume 679 of Studies in Computational Intelligence. Springer, Berlin (2017)

Kennedy, J., Eberhart, R.C., Shi, Y. (eds.): Swarm Intelligence, The Morgan Kaufmann Series in Artificial Intelligence, pp. 475–495. Morgan Kaufmann, San Francisco (2001). ISBN:9781558605954

LeCun, Y., Huang, F.J., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, CVPR'04, pp. 97–104, Washington, DC, USA. IEEE Computer Society (2004)

Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: proceedings of IEEE ICIP, pp. 10–14 (2002). https://doi.org/10.1109/ICIP.2002.1038171

Lienhart, R., Kuranov, A., Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: DAGM-Symposium (2003)

Liu, H., Chen, S., Kubota, N.: Intelligent video systems and analytics: a survey. IEEE Trans. Ind. Inform. **9**(3), 1222–1233 (2013)

Marsland, S.: Machine Learning: An Algorithmic Perspective, 1st edn. Chapman & Hall/CRC, London (2009)

Mekami, H., Benabderrahmane, S., Bounoua, A., Taleb-Ahmed, A.: Local patterns and big time series data for facial poses classification. JCP **13**(1), 18–34 (2018)

Metidji, B., Taib, N., Baghli, L., Rekioua, T., Bacha, S.: Phase current reconstruction using a single current sensor of three-phase AC motors fed by SVM-controlled direct matrix converters. IEEE Trans. Ind. Electron. **60**(12), 5497–5505 (2013)

Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc, New York (1997)

Pai, G.A.V., Michel, T.: Metaheuristic optimization of constrained large portfolios using hybrid particle swarm optimization. Int. J. Appl. Metaheuristic Comput. **8**(1), 1–23 (2017)

Panda, S., Padhy, N.P.: Comparison of particle swarm optimization and genetic algorithm for facts-based controller design. Appl. Soft Comput. **8**(4), 1418–1427 (2008). (Soft Computing for Dynamic Data Mining)

Papageorgiou C., Oren, M., Poggio, T.A.: A general framework for object detection. In: Proceedings of 6th International Conference on Computer Vision, Bombay, 4–7 January 1998, pp. 555–562 (1998). https://doi.org/10.1109/ICCV.1998.710772

Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**(1), 33–57 (2007)

Palit, I., Reddy, C.K.: Scalable and parallel boosting with mapreduce. IEEE Trans. Knowl. Data Eng. **24**(10), 1904–1916 (2012)

Rudin, C., Daubechies, I., Schapire, R.E.: The dynamics of adaboost: cyclic behavior and convergence of margins. J. Mach. Learn. Res. **5**, 1557–1595 (2004)

Schapire, R.E.: A brief introduction to boosting. In: Proceedings of the 16th international joint conference on artificial intelligence, vol 2, IJCAI'99, pp 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc, San Francisco (1999)

Schapire, R.E.: Theoretical views of boosting and applications. In: Proceedings of the 10th international conference on algorithmic learning theory, ALT '99, London, UK, pp. 13–25. Springer, Berlin (1999)

Schapire, R.E.: Advances in boosting. In: Proceedings of the eighteenth conference on uncertainty in artificial intelligence, UAI'02, San Francisco, CA, USA, pp. 446–452. Morgan Kaufmann Publishers Inc., San Francisco (2002)

Schapire, R.E., Singer, Y.: Boostexter: a boosting-based systemfor text categorization. Mach. Learn. **39**(2–3), 135–168 (2000)

Shantaiya, S., Verma, K., Mehta, K.: A survey on approaches of object detection. Int. J. Comput. Appl. **65**(18), 14–20 (2013). (Published by Foundation of Computer Science, New York, USA)

Talbi, E.-G.: Metaheuristics: from design to implementation. Wiley Publishing, Hoboken (2009)

Tavares, Y.M., Nedjah, N., de Macedo Mourelle, L.: Tracking patterns with particle swarm optimization and genetic algorithms. IJSIR **8**(2), 34–49 (2017)

Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)

Venter, G.: Review of Optimization Techniques. Wiley, New York (2010)

Viola, P.A., Jones, M.J.: Fast and robust classification using asymmetric Adaboost and a detector cascade. In: Proceedings of NIPS, pp. 1311–1318. MIT Press, Cambridge (2001a)

Viola, P.A., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 511–518 (2001b). https://doi.org/10.1109/CVPR.2001.990517

Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco (2011)

Wu, Y., Chen, H., Zhao, X., Zhai, Y.: A vision-based recognition method for transformer based on adaboost and multi-template matching. In: 2015 IEEE international conference on cyber technology in automation, control, and intelligent systems (CYBER), pp. 1429–1432 (2015)

Yang, M.-H., Kriegman, D.J., Ahuja, N.: Detecting faces in images: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**(1), 34–58 (2002)

Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. **38**(4), 1–45 (2006). https://doi.org/10.1145/1177352.1177355

Sidahmed Benabderrahmane The main research interest of Dr. Sid Ahmed Benabderrahmane concerns the overall KD (knowledge discovery) loop that leads to the use of machine learning (ML) and data mining (DM) techniques to resolve complex problems and extract knowledge from structured and unstructured data. He has been involved in many academic and industrial applications within many projects, using different kind of data, such as: spatio-temporal sequences, biomedical databases, social data, images or textual corpus. He earned his PhD from Nancy University (France) in December 2011, and an MSc in computer science from Paris Evry University in 2007. Actually, he is a research scientist at the University of Edinburgh (UK). Before that, he was a postdoctoral researcher and a teaching assistant at the LIASD artificial intelligence and robotics laboratory of Paris 8 University, where he has been working on big data mining, deep learning, time series forecasting, and recommendation problems. Before that, he was a research associate at the INRIA laboratory where he worked on time series data mining.

Sid Ahmed has also gained a valuable experience in image processing and artificial life domains. He is interested in the use of meta-heuristic methods with boosting classifiers for for real time object detection and tracking for autonomous cars and unmanned vehicles.