Noname manuscript No.
(will be inserted by the editor)

Particle Swarm Optimisation Based AdaBoost for Object Detection

Ammar Mohemmed[1], Mark Johnston[2], Mengjie Zhang[1⋆]

[1] School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
[2] School of Mathematics, Statistics and Operations Research, Victoria University of Wellington, PO Box 600, Wellington, New Zealand

xxx

Abstract This paper proposes a new approach to using particle swarm optimisation (PSO) within an AdaBoost framework for object detection. Instead of using exhaustive search for finding good features to be used for constructing weak classifiers in AdaBoost, we propose two methods based on PSO. The first uses PSO to evolve and select good features only and the weak classifiers use a simple decision stump. The second uses PSO for both selecting good features and evolving weak classifiers in parallel. These two methods are examined and compared on two challenging object detection tasks in images: detection of individual pasta pieces and detection of a face. The experimental results suggest that both approaches perform quite well for these object detection problems, and that using PSO for selecting good individual features and evolving associated weak classifiers in AdaBoost is more effective than for selecting features only. We also show that PSO can evolve and select meaningful features in the face detection task.

Key words Particle swarm optimisation, AdaBoost, object classification, object recognition

1 Introduction

Object detection attempts to determine the existence of specific objects in a set of images and, if present, to determine the locations, sizes and shapes of these objects. It is a challenging problem because objects can occur under different orientations, lighting conditions, backgrounds and clutter. It often utilises a trained binary classifier that can distinguish the objects of interest from the background (including objects of other classes).

One of the methods that was intensively investigated to improve the performance of object classification is to use an ensemble of classifiers. Instead of attempting to build a single (strong) classifier, a bundle of classifiers that individually are not necessarily powerful, are grouped to share the burden of the classification task. Studies have shown that the performance of the ensemble is better than any of its components acting alone [1]. A large number of combination schemes and ensemble methods have been proposed in literature (for a survey see [2]), which can be categorised into two approaches. The first approach is the use of an ensemble of accurate, well trained classifier members. The effectiveness of this approach depends on the accuracy and diversity of the members [3,4]. To achieve good performance, the individual members in the ensemble should exhibit low error rates and produce uncorrelated errors.

The second approach to ensemble classification is to allow more tolerance to the accuracy of the individual classifiers, i.e., *weak* classifiers [5]. Two popular methods are Bagging and Boosting, which both rely on resampling the features to obtain different training sets for each of the classifiers. Bagging [6] combines classifiers each individually trained on a *bootstrap replica* of the original training set. Boosting refers to a general and provably effective method of producing an accurate ensemble by combining rough and moderately inaccurate rules of thumb. Kearns and Valiant [7] proved the fact that learners, each performing only slightly better than random, can be combined to form an arbitrarily good ensemble hypothesis [8].

One fast, robust detection system, based on learning, is AdaBoost ("adaptive boost") for object detection by Viola and Jones [9]. This system has three main characteristics: using the AdaBoost boosting algorithm to combine simple *weak* classifiers into a more effective *strong* classifier; use of an integral image to rapidly compute simple features; and using a cascade of AdaBoost classifiers to quickly eliminate most negative images from consideration. Due to its robustness, it has been used in different applications including face and pedestrian de-

⋆ *mark.johnston@msor.vuw.ac.nz, mengjie.zhang@ecs.vuw.ac.nz*

tection [10], gender classification [11], text detection [12] and others.

Basically, a weak classifier for image recognition and processing tasks consists of a Haar-like feature and a decision threshold. Haar-like features are rectangular regions developed by gray and white halves (see Figure 2), where the value of the feature is the difference between the pixel intensities in these halves. If the value of the feature is smaller (larger) than a decision threshold, the detection sub-window is classified to be positive or negative. The training algorithm consists of a number of rounds, where in each round a weak classifier is constructed by selecting the feature among the available ones that best minimizes the weighted classification error. Selecting the feature is done in an exhaustive search mechanism over every possible feature. It is this exhaustive search that prolongs the training time. That is because even in a small image of 24×24 pixels, there might be more than 180 000 possible features available to compute their responses on thousands of training examples to evaluate their classification error and then to select the best. In fact, the complete face detection cascade of Viola and Jones has over 6000 features, trained on 20 000 images, a total of $6000 \times 180000 \times 20000 = 2.16 \times 10^{13}$ feature evaluations.

Introduced by Kennedy and Eberhart in 1995 [13], particle swarm optimization (PSO) is a population-based evolutionary algorithm for problem solving, based on social-psychological principles and provides insights into social behaviour. At the beginning of evolution, an initial population of individual particles are defined as random guesses at the problem as candidate solutions. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions and remember the locations where they had their best success. The individual's best solution is called the local best. Each particle makes this information available to their neighbours. They are also able to see where their neighbours have had success. Movements through the search space are guided by these successes, with the population usually converging, by the end of a trial, on a global best solution. Compared with other main evolutionary paradigms such as genetic algorithms [14] and genetic programming [15], PSO is a relatively new paradigm, but is considered particularly suitable to optimising a large number of parameter values efficiently. PSO has been applied to a variety of optimisation and image recognition tasks and achieved a certain level of success [16,17]. As the above AdaBoost approach to object detection needs a large number of features/weak classifiers, PSO has the potential to automatically evolving a good set of features and weak classifiers for AdaBoost and improve the system performance.

1.1 Goals

To avoid these limitations of the above AdaBoost approach, the goal of this paper is to investigate a new approach using Particle Swarm Optimization (PSO) within an AdaBoost framework for object detection. Instead of using an exhaustive search, we aim to use PSO to automatically search for good features to reduce the computational cost. We will consider two PSO approaches for this purpose. The first one (PSOAdaBoost1) considers using PSO to evolve and select the good features only and the weak classifiers use a simple decision stump. The second (PSOAdaBoost2) considers using PSO for both selecting the good features and evolving weak classifiers. These approaches will be examined and compared on two challenging object detection tasks in images. The first is a pasta detection task derived from the competition at GECCO 2006 [18] which requires the detector to identify the location of all pieces of pasta (of various shapes, sizes and orientations) in an image. The second is a face detection task [19] where we are specifically interested in which features are selected by the method.

It is important to note that the goal of this paper is not to find an existing method (or develop a new method) that can achieve the best results for a specific task, or investigate general guidelines for which learning or evolutionary method is good for specific tasks [20]. Instead, the goal here is focused on investigating whether PSO can improve the performance of AdaBoost for object detection. Specifically, we investigate: (1) whether the two PSO approaches can achieve acceptable results on these tasks in comparison with the standard AdaBoost with exhaustive search (ExAdaBoost); (2) whether selecting good features only (as in PSOAdaBoost1) or good weak classifiers (as in PSOAdaBoost2) is most effective; and (3) whether PSO can evolve meaningful or interpretable features or combinations of features.

1.2 Organisation

In the remainder of the paper, Section 2 briefly describes the background about AdaBoost for object detection, PSO, face detection and related work. Section 3 describes the new PSO approaches for selecting good features and evolving weak classifiers. We then consider two experimental case studies: pasta detection in Section 4, and face detection in Section 5. Finally, Section 6 offers some conclusions and recommendations for future work.

2 Background

2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic optimization tool inspired by social behaviour

of flocks of birds (and schools of fish, etc.), as developed by Kennedy and Eberhart [13]. PSO starts with a population of particles whose positions represent the potential solutions for the studied problem and velocities are randomly initialized in the search space. The search for optimal position (solution) is performed by updating the velocity and position of particle i according to the following two equations for $i = 1, 2, \ldots, N_s$.

$$V_i(t+1) = V_i(t) + \phi_1 r_1(B_i(t) - X_i(t)) + \phi_2 r_2(B_i^g(t) - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where ϕ_1 and ϕ_2 are positive constants, called *acceleration coefficients*, N_s is the total number of particles in the swarm, r_1 and r_2 are two independently generated random numbers in the range $[0, 1]$ and g represents the index of the best particle in the neighborhood of a particle. The other vectors are defined as: $X_i = [x_1, x_2, \ldots, x_{iD}] \equiv$ position of ith particle; $V_i = [v_1, v_2, \ldots, v_D] \equiv$ velocity of the ith particle; $B_i \equiv$ position of the ith particle found so far, and $B_i^g \equiv$ best position found by the neighborhood of the ith particle. When the convergence criterion is satisfied, the best particle (with its position) found so far is taken as the solution to the problem. The pseudocode of PSO is shown in Algorithm 1.

Algorithm 1 Pseudocode for basic PSO [13]

```
initialize particles population
while maximum iteration or required fitness is not attained do
  calculate the fitness of each particle i
  update B_i if the current fitness is better than before
  determine B_i^g from the neighbours
  for each particle i do
    calculate V_i according to Eq (1)
    update X_i according to Eq (2)
    update the best global solution
  end for
end while;
```

However, in most cases, the velocities quickly attain very large values, especially for particles far from their global best. To control the increase in velocity, velocity damping is used in Eq.(1). Thus, if the right side of Eq.(1) exceeds a specified maximum value $\pm V^{max}$, then the velocity on that dimension is clamped to $\pm V^{max}$. In [21], Maurice proposed the use of a constriction factor to prevent velocity from growing out of bound:

$$\bar{V}_i(t) = \chi\Big(V_i(t) + \phi_1 r_1(B_i(t) - X_i) + \phi_2 r_2(B_i^g - X_i)\Big) \quad i = 1, 2, \ldots, N_s$$

$$\chi = 2\Big(\Big|2 - \phi - \sqrt{\phi^2 - 4\phi}\Big|\Big)^{-1} \quad (3)$$

where $\phi = \phi_1 + \phi_2$.

PSO has two typical topologies [22]: the Global Neighbourhood Topology and the Ring Topology (see Figure 1). The global topology supports fast information sharing between particles and would be used in cases where efficiency is valued over effectiveness. The ring topology slows down knowledge sharing so it can stop particles from converging to a local optimum solution too quickly.

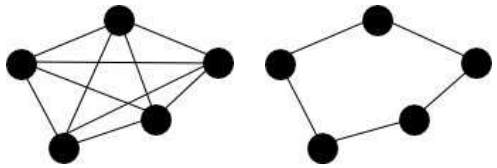

Fig. 1 PSO topologies: global (left) and ring (right)

2.2 AdaBoost for Object Detection

AdaBoost [23] is the most well known boosting procedure. It has been used in numerous empirical studies and has received considerable attention from the machine learning community in the last few years. Freund and Schapire [23] showed two interesting properties of AdaBoost. First, the training error exponentially goes down to zero as the number of classifiers grows. Second, AdaBoost still learns after the training error reaches zero.

Viola and Jones [9] developed an AdaBoost system for object detection. The system introduced a cascade of AdaBoost classifiers to quickly eliminate most non-faces from consideration speeding up the process of detection significantly. The system uses a set of 5 basic types of simple Haar-wavelet like features which are displayed in Figure 2. A feature is calculated by subtracting the sum of pixel grey scale values of the white area of the rectangle from the dark area. The complete set of features (180,000 according to [9]) is generated by varying the width, height and starting position of each of these features with respect to a 24×24 window within an image. The feature set can be parameterized by four parameters: width, height, and the feature's offset within the window as shown in Figure 2(b). The advantage of using these simple features is that they can be calculated very quickly with the use of an "integral image". An integral image II over an image I is defined as follows:

$$II(x, y) = \sum_{x' \le x, y' \le y} I(x', y')$$

Recalling that there are thousands of rectangle features associated with each image sub-window, the hypothesis is that a very small number of these features can be combined to form effective classifiers. The main

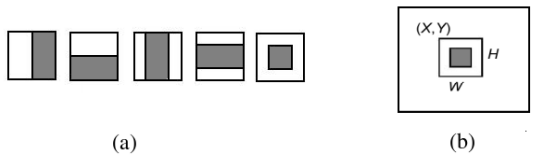

Fig. 2 (a) Standard Haar-like features. (b) The parameters of a feature. (X, Y): position of the detection window, W and H: width and height of the feature.

challenge is to find these features. The AdaBoost algorithm shown in Algorithm 2 attempts to find these features. The algorithm is supplied with a training set $(x_1, y_1), \ldots, (x_N, y_N)$ where $y_i \in \{-1, 1\}$, weighted by w_i uniformly. The algorithm iterates over a number of T rounds. In every round t, and for each feature f_j, there is a weak classifier $h_j(x)$ that consists of the feature, a decision threshold θ_j equaling $\frac{1}{2}(C_{-j} + C_{+j})$ where C_j is the mean of the feature responses on the examples.

Exhaustive search is done to select the classifier $h_t(x)$, among the available ones, with the minimum classification error ϵ_t, defined as the total weights of the misclassified examples. At the end of each round, the weights of the training examples misclassified by h_t are increased, so that the focus in the next round will be on these examples. The classifier with the minimum error $h_j()$ at round t is taken to be the winner among all available classifiers. The final *strong* classifier $H(x)$ at the end of T rounds is a linear combination of the weighted weak classifiers, where each classifier is weighted by a parameter α_t proportional to its error rate ϵ_t.

The pseudocode of the discrete AdaBoost algorithm is listed in Algorithm 2. It is called discrete because the output of the classifier is either $+1$ for a positive example or -1 for a negative example.

Algorithm 2 Pseudocode for AdaBoost [9]

Given N examples $(x_1, y_1), \ldots, (x_N, y_N)$ where $y_i \in \{-1, 1\}$
initialize $w_{1,i} = 1/2m, 1/2l$ for $y_i = -1, 1$ respectively, where m and l are the number of negatives and positives respectively.
for t = 1,...,T **do**
(1) for each feature j, train a classifier $h_j()$
(2) evaluate the error of the classifier $\epsilon_j = \sum_{i=1}^{N} \omega_{t,i}.b_i$
(3) choose a classifier $h_t()$ with lowest error ϵ_t
update weights: $\omega_{t+1,i} = w_{t,i}\beta_t^{1-b_i}$
where $b_i = 0$ if $h_t(x_i) = y_i, b_i = 1$ otherwise
with $\beta_t = \epsilon_t/(1 - \epsilon_t)$
end for
output strong classifier:

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t > 0 \\ -1 & \text{otherwise} \end{cases}$$

with $\alpha_t = log(1/\beta_t)$

2.3 Face Detection

One of the experimental case studies considered in this paper is detection of a face in an image (see Section 5). Face detection and localization is an important image analysis step in many computer vision applications like human computer interaction, biometrics, computer surveillance cameras and other security and authentication applications. This is a challenging problem due to variability in pose, occlusion, orientation, lighting, etc., and many different techniques have been proposed to solve it. These techniques differ in the concept of operation and can be classified into four categories [24]. Firstly, top-down knowledge based methods which are based on including human knowledge of what constitutes a typical face in the detection process, for example, a face often appears in an image with two eyes that are symmetric to each other, a nose, and a mouth. Secondly, bottom-up feature based methods, based on using features invariant to the pose, viewpoint or lighting conditions like skin colour and facial features such as eyebrows, eyes, nose, mouth, and hair line which are commonly extracted using edge detectors. Thirdly, template matching methods based on computing the correlations between an input image and stored patterns. Lastly, appearance based methods where, in contrast to template matching, the models (or templates) are learned from a set of training images. In general, appearance based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non face images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection. These learning methods have attracted much attention and have demonstrated a certain level of success [9, 23, 25–29].

Apart from the underlying technique, how to represent the image in terms of *features* affects the performance in terms of accuracy and computation efficiency. In general, image features are distinguished into global and local features. Global features quantify characteristics of the whole image to be measured. An example would be a colour histogram of an image which gives important information about the whole image. Local features quantify characteristics of a particular region of the object to be measured.

2.4 Related Work

To reduce the training time of AdaBoost, heuristic and evolutionary algorithms have been incorporated within an AdaBoost framework to replace the exhaustive search. McCane and Novins [19] considered that selecting features is an optimization problem with a well behaved fitness function represented by the error rate. They proposed a simple local search based on examining neighbourhood points and moving to the smallest error point.

A significant speed improvement over the training method presented by Viola and Jones [9] was achieved with only a modest increase in execution time. The feature selection method of Bartlett et al. [30] used a heuristic to find promising features; they selected 5% of all possible features randomly, and choose the best in terms of error rate on the training examples. This feature was then refined by shifting, scaling and reflecting to produce new sets of features and the best was selected from this new set to be the winner for that round. This feature selection process was then repeated with the updated weighted examples until the strong classifier achieves a minimum desired performance rate. Treptow and Zell [31] proposed to extend the feature family proposed by Viola and Jones to a more generalized set of similar features, and to use a genetic algorithm search as a weak learner, where a chromosome encodes the parameters of the feature (position and dimensions). They found that AdaBoost training with their evolutionary search over their larger feature set produced better detectors than exhaustive search applied to the initial limited feature set. Abramson et al. [32] used a new type of feature called *control points* with AdaBoost for pedestrian detection tasks. However, training is impossible using exhaustive searching due to their huge number of features. They proposed an evolutionary hill climbing algorithm for training their detector.

Li et al. [33] reported the use of support vector machines as weak classifiers for AdaBoost for classification tasks. Two algorithms, called *AdaBoostSVM* and *Diverse AdaBoostSVM* were developed and compared with a neural network based AdaBoost algorithm on 13 benchmark datasets chosen from the UCI machine learning repository. The results suggest that the proposed AdaBoostSVM algorithm performed better than the neural network based AdaBoost algorithm, and that the proposed diverse AdaBoostSVM performed slightly better than the SVM algorithm.

Hidaka and Kurita [34] reported the use of a PSO algorithm to evolve weaker classifiers in AdaBoost for face detection. The MIT CBCL face database was used in the experiments. They claimed that the proposed algorithm was 50 times faster than the usual AdaBoost while keeping comparable classification accuracy.

Another issue, which has got less attention, is how to determine the decision threshold of the weak classifier. Because the weak classifier is not required to be very accurate, a single threshold based on averaging the means of feature responses on the negative and positive training examples is widely used. However, Rasolzadeh et al. [35] reported a better performance using multithresholding. The multithresholds are found by assuming that the feature responses each follow a normal distribution. For the general case of multimodal distributions the feature responses are placed in histogram-like model.

3 New PSO Based Approaches to Object Detection

We have developed two PSO methods to object detection. Our approach treats the tasks of finding the discriminative features and their thresholds as an optimization problem to be solved by PSO.

3.1 PSOAdaBoost1 with Simple Weak Classifiers

PSOAdaBoost1 uses PSO to select good features only and the weak classifiers use a simple decision stump.

Particle Encoding. A particle encodes the Haar-like feature parameters. The Haar-like features can be parameterized by the type (one of the types shown in Figure 2 (a)), upper-left position (X, Y), with respect to the detection window, and the size (W, H) in Figure 2(b). Figure 3 shows the encoding of the particle.

Type	X	Y	W	H

Fig. 3 Particle Encoding in PSOAdaBoost1.

Weak Classifiers. Because the weak classifiers are only required to be better than random guessing, a decision stump (a depth-one decision tree which compares a single input feature to a threshold) has been widely used [12,36]. The examples are assigned to the positive class if their single feature values are greater than the threshold and to the negative class otherwise. During learning in the standard AdaBoost with exhaustive search (ExAdaBoost), each feature in the input feature vector is examined in turn to find the best feature. The threshold T is computed by taking the average of the means of the feature values from negative and positive examples as shown in Figure 4(a).

Fitness Function. We seek to use PSOAdaBoost1 to find the best features to be used by the weak classifiers. The fitness function is to minimize the weighted error ϵ_j in AdaBoost (see Algorithm 1) for object detection, i.e.,

$$\epsilon_j = \sum_{i=1}^{N} \omega_{t,i}.b_i$$

$$\text{where} \qquad b_i = \begin{cases} 0 & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

3.2 PSOAdaBoost2 for Evolving Features and Weak Classifiers

PSOAdaBoost2 uses PSO for selecting good features and evolving corresponding thresholds in parallel. The particle encoding is shown in Figure 5. In addition to the five

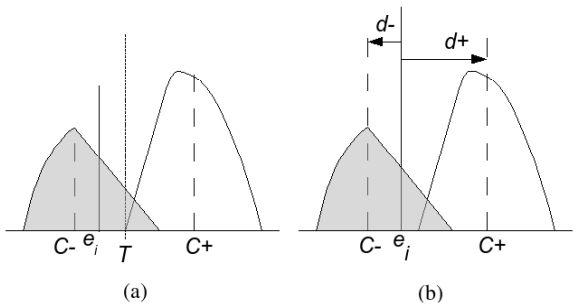

Fig. 4 (a) PSOAdaBoost1: using decision stump. C_- and C_+ are computed by taking the means of negative and positive examples respectively. Example e_i is labelled with class C_- as it is less than T. (b) PSOAdaBoost2: C_- and C_+ are found using PSO, and example e_i is labelled with class C_- as $d- < d+$.

parameters in PSOAdaBoost1, the particles in this PSO also include the 'centroids' of the positive and negative examples. The C_- and C_+ are automatically evolved by PSO in this method rather than being simply calculated by taking the mean of the examples. Instead of selecting a feature and then training a classifier to evaluate the goodness of the feature, the parameters of the feature and the associated weak classifier are considered as a whole to be optimized using PSO.

Type	X	Y	W	H	C_-	C_+

Fig. 5 Particle Encoding in PSOAdaBoost2.

Figure 4(b) shows how the weak classifier operates. For a given example e_i, the distance to the evolved centroids are computed and the example is labelled with the class that has the shortest distance. Because these two centroids are found in such a way to reduce the training error, the constructed weak classifiers are expected to be more accurate. In this case, the optimum centroids are the two points that minimize the inner class distance and maximize the across class distance on the respective feature axis. In other words, two centroids (C_- and C_+) are found to maximize the following class separation criteria:

$$J(C_-, C_+) = \frac{|C_- - C_+|}{\frac{1}{m}\sum_{i=1}^{m}|C_- - e_i^-| + \frac{1}{l}\sum_{i=1}^{l}|C_+ - e_i^+|}$$

where m is the number of negative examples and l is the number of positive examples. Instead of maximizing this criteria based on the distances, it is maximized using PSO indirectly in such a way as to reduce the error rate. This is similar (in concept) to [37], where they maximize the class separation represented by Fisher discriminant formula through evolutionary minimizing of the classification error. However, that approach does not work for a single feature.

PSOAdaBoost2 uses the same fitness function as PSOAdaBoost1. In PSOAdaBoost2, PSO is incorporated into AdaBoost to replace the exhaustive feature search and the weak classifier training. Thus, the three steps in the inner loop of Algorithm 2 are replaced by PSOAdaBoost2. In each round PSO is called to construct a new weak classifier.

4 Experimental Case Study: Pasta Detection

4.1 Task and Image Dataset

The task here is to locate all pieces of pasta in a given image. We use the pasta image dataset presented in the competition at the GECCO conference in 2006 [18] with the goal of evaluating and comparing the effectiveness of the proposed PSOAdaBoost1 and PSOAdaBoost2. Although the original problem in the competition was for image segmentation [38], and the images were coloured, we used these images for object (pasta) detection in this paper and converted the colour images into gray-scale images. The images are 1280×960 pixels and contain pasta pieces of different size, position and rotation with varied lighting conditions. The background of some images are noisy, including some pieces of alphabet soup, whose intensities are very similar to the pasta objects, making the detection problem even more challenging. Figure 6 shows some examples of these images used in the training process.

4.2 Object Detection Process

The object detection process using both PSO approaches consists of a training step and a testing step. In the training process, both AdaBoostPSO algorithms are applied to the training dataset of object "cutout" examples. As the pasta pieces are of different sizes and rotations and the images are of large size, it is not so straightforward to exactly cut out the pasta from the large images. In this paper, we implement the following procedure to create training examples. On the training images, the pasta pieces are firstly marked differently from the background. A scanning window, of size 61×61 pixels, moves across the large training images in discrete jumps considering the current position as a pasta (positive) example if it stands over a marked "white" spot, and a position is considered a negative (non-pasta) example if the scanning window stands on a "black" marked spot. An example marked image is shown in the first column in Figure 6. In this way, a window could be still considered a positive example if it includes some part of a pasta and some pieces of background. Thus, the positive examples contain different parts of the pasta pieces rather than one complete piece. In addition, these parts could be quite different as the pasta pieces have different rotations, shapes and brightness. Following this procedure,

Fig. 6 Pasta training images used to extract training examples. The first image shows the marking to create the examples (white areas are positive examples and black areas are negative examples).

1297 positive examples and 2434 negative examples were generated to form the training set.

The trained classifiers were then used as a template detector, in a moving window fashion, to detect the pasta objects over the *test set* of full images. The detection window size is the same as the size of the training object cutout examples (61×61 pixels) and moves pixel by pixel over the large testing images. If a window is detected to be of pasta type, the centered pixel under the detection window is marked as an object (pasta) pixel. In this way, the pasta pixels are detected and an indication of the whole pasta (size and rotation) is given.

4.3 Experiment Configuration

In all the experiments, both PSO approaches used the ring topology. The ring is fixed in our experiments and a particle is connected to only two other particles (neighbours). The neighbours do not change during an experiment run. Considering the computational effort in the training phase, the two PSO approaches used the same population size of 20 and a maximum 100 iterations for comparison purposes, so there are 2000 different features generated during the run to evolve the best feature and construct a weak classifier. These parameter values were set based on common settings and some preliminary empirical search. Note that this number is much smaller than that needed to be evaluated using the exhaustive search in standard AdaBoost (roughly searching for 2×10^{11} features to find a single weak classifier), where a large percentage of them are redundant and do not have any discriminative power.

The particles are initialized within a range that does not exceed the size of the detection window and the two centroids parameters are initialised from $[-20000, 20000]$. In fact, this is not too critical as only the relative distance between the centroids is of concern.

As it is too expensive to use ExAdaBoost algorithm for the pasta detection process, we did not include it in our comparison. The next section will describe the experiment results of the two proposed PSOAdaBoost approaches.

4.4 Experimental Results and Discussion

To measure the performance of the two proposed PSOAdaBoost approaches, we used the Receiver Operating Characteristic (ROC) curve [39] for the test set. To compute points on the ROC curves, we used the large images in the test set to mark and generate 2941 positive examples and 3411 negative examples. The two marked test images are shown inset in Figure 7. The points on the ROC curves were obtained by evaluating the strong classifier against these examples while sliding the confidence threshold over the range $(-\infty, +\infty)$, and taking the average results of 100 independent runs. The ROC curves for the two approaches are shown in Figure 7. In this figure, the dashed curve represents the performance of the PSOAdaBoost1 approach and the solid curve corresponds to the PSOAdaBoost2 method.

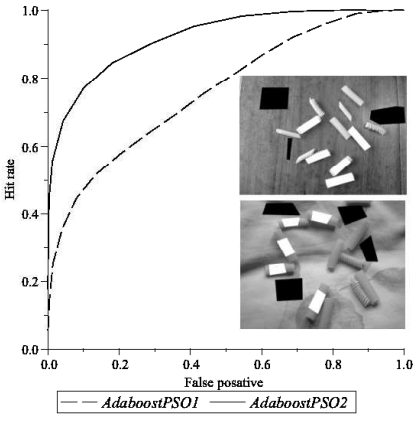

Fig. 7 ROC curves of PSOAdaBoost1 (solid curve) and PSOAdaBoost2 (dashed curve).

As can be seen from Figure 7, both curves are much higher than the diagonal from $(0, 0)$ to $(1, 1)$, showing that both new PSO based approaches performed the pasta detection quite well. The solid curve is higher than the dashed curve at almost all false positive rates, showing that PSOAdaBoost2 achieved better performance than PSOAdaBoost1.

To give an intuitive view of the detection results, Figure 8 shows the object sweep maps of pasta images in the test set. The images in the first row are the original images that were not used in the training process. The sec-

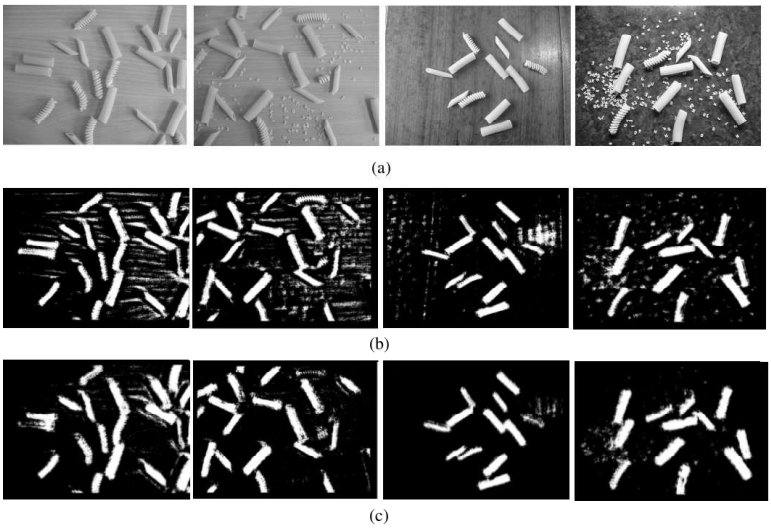

Fig. 8 Past detection: (a) unseen test images; (b) sweep maps produced by PSOAdaBoost1; (c) sweep maps produced by PSOAdaBoost2.

ond row shows the corresponding sweep maps achieved by PSOAdaBoost1 and the third row show the sweep maps of the original images achieved by PSOAdaBoost2. In these sweep maps, white pixels represent the "pasta pixels" detected by the two approaches. Clearly, these sweeping maps contain a number of false positives, that is, some non-pasta pixels (either noisy background or some alphabet soup) were incorrectly detected as pasta pixels. The sweeping maps confirm that both approaches successfully detected all pasta pieces in the images, but PSOAdaBoost2 produced a smaller number of false positives for all the examples images. In particular, PSOAdaBoost2 is much more tolerant to the noisy background than PSOAdaBoost1, and is also much better in discriminating pasta pixels from the alphabet soup. These results suggest that using PSO to select good features and evolve corresponding weak classifiers in AdaBoost is better than to select features only for the pasta detection problem examined in this paper.

5 Experimental Case Study: Face Detection

5.1 Task and Image Dataset

In this second case study we compare the standard AdaBoost with exhaustive search (ExAdaBoost) against our best PSO based AdaBoost (PSOAdaBoost2). Here ExAdaBoost solves the subproblem to determine a weak classifier exactly (using exhaustive search) whereas PSOAdaBoost2 solve this subproblem approximately (using PSO). The question is which gives better effectiveness on the full problem to design a strong classifier. The task here is to distinguish between face and non-face images. We are also specifically interested in what features are selected by PSO within the AdaBoost framework and whether these can be meaningfully interpreted.

For training and testing we use an image dataset consisting of 4999 images showing different faces and 6960 images which do not show face images and extracted randomly from images of different background [19]. Samples of these images are displayed in Figure 9. The gray value images have size of 25×25 pixels and are variance normalized. The face/non face sets are split randomly into a training set which consists of 2575 positives and 3480 negatives and a test set consists of 2424 positives and 3480 negatives. Caution has been taken during splitting to avoid similar face images being shared between the training set and test set.

Fig. 9 Sample images from face/non-face dataset.

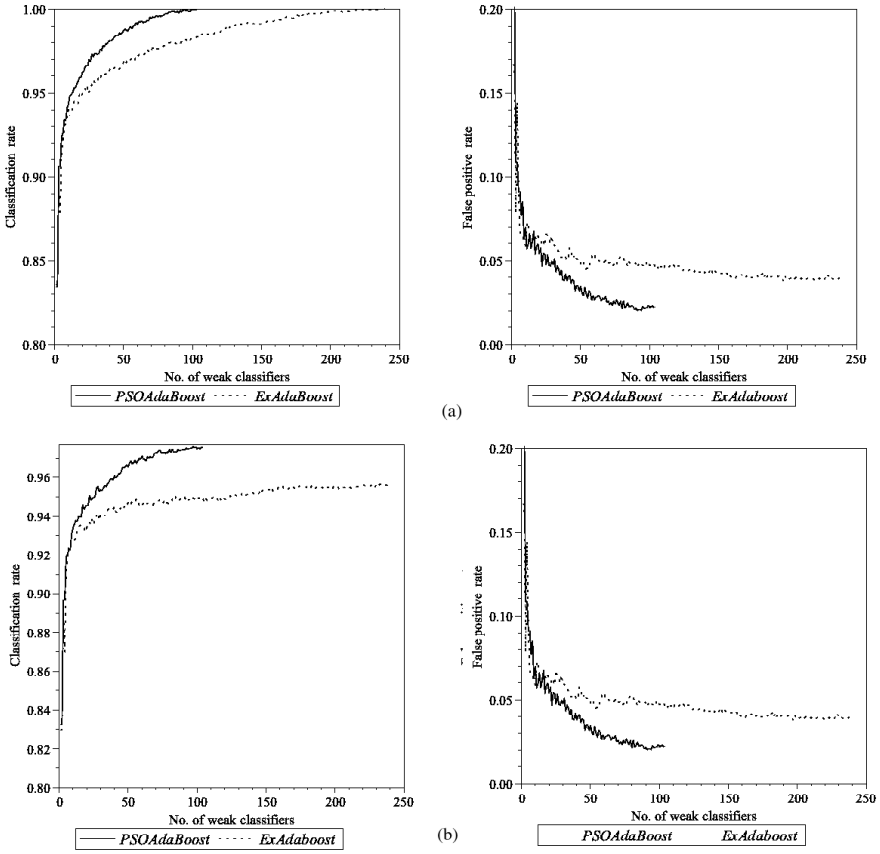

Fig. 10 Face detection performance with different numbers of weak classifiers: (a) results for the training set; (b) results for the test set.

5.2 Experiment Configuration

Similarly to the pasta experiment, the ring topology was used here again. The ring is fixed in our experiments and the neighbours do not change during an experiment run. Since we are comparing PSOAdaBoost2 against ExAdaBoost, we allow the PSO more resources than the pasta experiments. Here, the PSO consists of a population of 200 particles and runs to a maximum of 1000 iterations. However, it is terminated if there was no improvement in the global solution in a period of 50 iterations. The population is initialised randomly such that the feature parameters part of the particle (x, y, w, h) are initialised within the range $[0, 25]$, type of the feature in the range $[0, 4]$ and the centroids part (C_- and C_+) in the range $[-50, 50]$. Here r_1 and r_2 are random numbers independently taken in the range of $[0, 1]$, and ϕ_1 and ϕ_2 are both set to 2.05. These parameter values are set based on the common settings and some preliminary experimentation, and we found that the results are not so sensitive to different parameter values. In fact, the last range is not critical because as explained in section 2.2, it is the relative distance of the example with respect to the centroids that is of concern. The experiments for each algorithm are repeated for ten runs and the best, average and the worst results are reported. Note that each run is initialised with a different random seed number. The experiments were run on a Pentium V 3 GHz machine.

5.3 Experimental Results and Discussion

Training Efficiency Performance. Compared with the commonly used exhaustive AdaBoost method (ExAdaBoost), the new method PSOAdaBoost2 is much more efficient for finding a good (strong) classifier. Using the large training set of the face images, ExAdaBoost spent an average time of 347,788 seconds to find a good classifier for achieving the ideal performance, that is, all the examples in the training set were correctly classified. The PSOAdaBoost2 method, however, only required an average time of 18,645 seconds to evolve a good classifier to achieve the same performance, which is 19 times more efficient.

Table 1 Number of features required for face detection.

	Best	Average	Worst
ExAdaBoost	240	240	240
PSOAdaBoost2	95	104	112

A closer inspection reveals that the new PSOAdaBoost2 method used a much smaller number of features (weak classifiers) than the exhaustive AdaBoost method. Table 1 shows the number of features generated to form a strong classifier for the ideal training performance. While PSOAdaBoost2 only required 95 features in the best case and 112 features in the worst case, while ExAdaBoost needs 240 features to reach that point. On average, the PSOAdaBoost2 method can evolve a good strong classifier with 104 features. This number is much smaller than the number needed by the ExAdaBoost method (240). To show a clear pattern, Figure 10(a) shows the relationship between the classification accuracy rate and false positive rate against the number of features/weak classifiers required by both methods on both the training test and the test set. These results suggest PSO is effective for evolving good weak classifiers for this problem and that it is much more efficient than the exhaustive method.

System Test Effective Performance. Table 2 summarises the overall comparison of the best results of PSOAdaBoost2 and ExAdaBoost on the test data. PSOAdaBoost2 has an average classification accuracy rate of 97.63% on the test set and a false positive rate of 2.25% for the face images. The ExAdaBoost method only achieved an average 95.55% of accuracy with 4.03% false positive for the faces, even using a large number of features (weak classifiers). Figure 10(b) shows the relationship between the classification accuracy and false positive rate against the number of weak classifiers (features) used to form a strong classifier. These results show that the proposed PSOAdaBoost2 method is not only more efficient than the existing ExAdaBoost method for learning/evolving a good classifier, but also more effective on the classification performance on the unseen test set. This also suggests that the new PSOAdaBoost2 method has a better generalisation ability for this dataset.

Table 2 Face detection performance on the test set.

	Average Classification rate	Average False Positive rate
ExAdaBoost	0.9555	0.0403
PSOAdaBoost2	0.9763	0.0225

5.4 Feature Analysis

To further understand why the PSOAdaBoost2 method does such a good job, we more closely inspected some features (weak classifiers) that PSOAdaBoost2 selected using PSO. Figure 11 shows six of these evolved features in the faces. We noticed that the evolved features were concentrated around the eyes region. As stated in the literature, the eye region features are important (even dominant) for face recognition. This suggests that PSO can efficiently find these useful features that can be combined to solve the face detection problem.

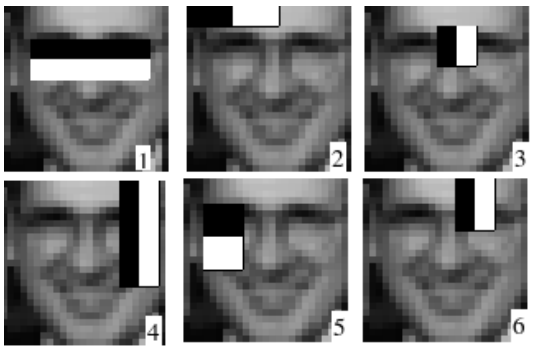

Fig. 11 The first six features selected by PSOAdaBoost2, overlayed on a sample image.

To inspect the roles these features played in the training and testing process, Table 3 shows the classification accuracy rate of the features acting individually or combined with the previously selected ones. The first row shows that feature 1 (in Figure 11) can result in 83.4% accuracy on the training set and 83.2% on the test set. Row 3 shows that using feature 3 alone can produce an accuracy rate of 79.2% and 78.3% on the training set and test set, respectively. But the combined feature sets (features 1, 2 and 3) can produce a classification accuracy of 90.6% and 89.7% on the training set and the test set, respectively.

Table 3 Classification performance of the first six features selected by PSOAdaBoost2.

	Training		Testing	
Feature No.	Indiv.	Combined	Indiv.	Combined
1	0.834	0.834	0.832	0.832
2	0.825	0.834	0.825	0.832
3	0.792	0.906	0.783	0.897
4	0.801	0.906	0.792	0.897
5	0.754	0.924	0.757	0.919
6	0.745	0.926	0.726	0.920

These results suggest that combining useful features together can generally improve the classification performance. However, the ways of combining the individually good features are not clear and adding a new selected feature that can individually perform well to a good combination does not always improve the classification performance. For example, adding feature 2 to feature 1 does not improve the performance, and adding feature 4 to the previous feature combination that consists of feature 1, 2 and 3 does not improve the performance either.

Therefore, an improvement to PSOAdaBoost2 that can be considered is to further investigate effective feature selection mechanisms, feature ranking algorithms and feature combination/construction methods using PSO in the future.

6 Conclusions

The goal of this paper was to investigate a new approach using PSO within AdaBoost for object detection. The goal was successfully achieved by using PSO for selecting good features and evolving weak classifiers in parallel within the AdaBoost algorithm for object detection. In this way, the original time consuming exhaustive search in AdaBoost was successfully avoided.

Two PSO based methods were developed in this paper. PSOAdaBoost1 considers using PSO to evolve and select the good features only and the weak classifiers use a kind of decision stump. PSOAdaBoost2 considers using PSO for both selecting the good features and evolving weak classifiers in parallel. The experimental results show that both approaches performed quite well for the pasta detection problem, and that using PSO for selecting good individual features and evolving associated weak classifiers in AdaBoost is more effective than for selecting features only for this problem. PSOAdaBoost2 was examined and compared with the exhaustive AdaBoost algorithm on a large face image dataset. The experiment shows that PSOAdaBoost2 method was able to be trained in much shorter time and achieved more accurate classification performance.

Inspection of the evolved individual features by PSO reveals that the new PSOAdaBoost algorithm can find meaningful features for face classification and the best evolved individual features were mostly concentrated on the eyes regions. Combining the good individual features sequentially into the top (group) features generally increased the face recognition performance, but not all the "good" individual features can make a clear improvement. Clearly, some of the evolved best/good individual features are redundant to a certain extent and integrating all of them sequentially based on their individual performance is not necessary.

As future work, the following directions will be considered for improving the system performance:

- This paper mainly considered selecting good individual features. The features may be selected based not only on individual fitness but also considering the context with the previously selected features. This will in turn reveal whether and how PSO can be effectively used for automatic feature construction, feature ranking and feature selection, which is a big topic in face and image detection and recognition as well as other classification tasks.
- Another interesting future direction is to investigate the effect of replacing the two prototype/centroid parameters used in this work with a single boundary parameter for binary classification.
- In this paper, we used three methods to determine the optimal thresholds: A simple decision tree (decision stump, in PSOAdaBoost1) and PSO (PSOAdaBoost2). From the system performance point of view, two other simple methods could be used. The first is to find the optimal thresholds by sorting the weighted Haar-like feature values of the training examples. The second is to keep the weighted feature values of the positive and the negative examples as histograms and to use the histograms for classification. One potential problem of these methods is that some of them can only be used for binary classification, but not for multi-class classification or detection problems.
- Although the new PSO approach was much more efficient than the exhaustive AdaBoost method for object detection and classification, the training time was still a bit too long (five hours). More efficient ways need to be investigated in the future.

Acknowledgment

We would like to thank the anonymous referees for their time, comments, and suggestions that provided great help for improving the paper.

This work was supported in part by the University Research Fund under the number of URF09-2399/85808 at Victoria University of Wellington for 2008/2009, and the Marsden Fund council from the government funding (08-VUW-014), administrated by the Royal Society of New Zealand.

References

1. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: A survey and categorisation. Journal of Information Fusion (Special issue on Diversity in Multiple Classifer Systems) **6** (March 2005) 5–20
2. Valentini, G., Masulli, F.: Ensembles of learning machines. Proceedings of the 13th Italian Workshop on Neural Nets, Lecture Notes in Computer Science **2468** (2002) 3–19
3. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. Neural Information Processing Systems **7** (1995) 231–238
4. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research **11** (1999) 169–198
5. Ji, C., Ma, S.: Combinations of weak classifiers. IEEE Transactions on Neural Networks **8** (Jan 1997) 32–42
6. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
7. Kearns, M.J., Valiant, L.G.: Cryptographic limitations on learning boolean formulae and finite automata. Journal of ACM **1** (1994) 67–95

8. Kearns, M.J., Valiant, L.G.: The boosting approach to machine learning: An overview. Nonlinear Estimation and Classification (Springer 2003)
9. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (December 2001) 511–518
10. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using pattern of motion and appearance. In: ICCV. (2003) 734–741
11. Verschae, R., del Solar, J.R., Correa, M.: Gender classification of faces using adaboost. Lecture Notes in Computer Science **4225** (2006) 68–78
12. Bargeron, D., Viola, P., Simard, P.: Boosting-based transductive learning for text detection. In: Eighth International Conference on Document Analysis and Recognition. Volume 2. (2005) 1166–1171
13. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ. (1995) 1942–1948
14. Goldberg, D.: Genetic Algorithms in Search, Optimisation and Machine learning. Addison Wesley, Reading, Ma (1989)
15. Koza, J.R.: Genetic programming : on the programming of computers by means of natural selection. Cambridge, Mass. : MIT Press, London, England (1992)
16. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. App. **2008** (2008) 1–10
17. Omran, M.G., Engelbrecht, A.P., Salman, A.A.: Particle swarm optimization for pattern recognition and image processing. In Abraham, A., Grosan, C., Ramos, V., eds.: Swarm Intelligence in Data Mining. Volume 34 of Studies in Computational Intelligence. Springer (2006) 125–151
18. Poli, R.: GECCO 2006 pasta segmentation competition. http://cswww.essex.ac.uk/staff/rpoli/GECCO2006/ (2006)
19. McCane, B., Novins, K.: On training cascade face detectors. In: Image and Vision Computing. (2003) 239–244
20. Tanwani, A.K., Afridi, J., Shafiq, M.Z., Farooq, M.: Guidelines to select machine learning scheme for classification of biomedical datasets. In Pizzuti, C., Ritchie, M.D., Giacobini, M., eds.: EvoBIO. Volume 5483 of Lecture Notes in Computer Science., Springer (2009) 128–139
21. Maurice, C.: The swarm and queen: towards a deterministic and adaptive particle swarm optimization. In: IEEE Congress on Evolutionary Computation. Volume 2. (1999) 1951–1957
22. Jian, W., Xue, Y.C., Qian, J.X.: An improved particle swarm optimization algorithm with neighborhoods topologies. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics. Volume 4. (2004) 2332–2337
23. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of International Conference on Machine Learning. (1996) 148–156
24. Yang, M.H., Kriegman, D., Ahuja, N.: Detecting faces in images: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(1) (2002) 34–58
25. Li, S., Zhang, Z.: Floatboost learning and statistical face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, **26**(9) (2004) 1112–1123
26. Sung, K.K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(1) (1998) 39–51
27. Rowley, H., Baluja, S., Kanade, T.: Neural network-based face detection. Proceedings of 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96) (1996) 203–208
28. Garcia, C., Delakis, M.: Convolutional face finder: a neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(11) (2004) 1408–1423
29. Huang, L.L., Shimizu, A., Kobatake, H.: Robust face detection using gabor filter features. Pattern Recognition Letters **26** (August 2005) 1614–1649
30. Bartlett, M.S., Littlewort, G., Fasel, I., Movellan, J.R.: Real time face detection and facial expression recognition: Development and application to human computer interaction. In: CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction. (2003) 139–157
31. Treptow, A., Zell, A.: Combining adaboost learning and evolutionary search to select features for real-time object detection. In: Congress on Evolutionary Computation. Volume 2. (June 2004) 19–23
32. Abramson, Y., Moutarde, F., Steux, B., Stanciulescu, B.: Combining adaboost with a hill-climbing evolutionary feature-search for efficient training of performant visual object detectors. In: 7th International FLINS Conference on Applied Artificial Intelligence (FLINS'06), Genova, Italy. (August 2006) 737–744
33. Li, X., Wang, L., Sung, E.: A study of adaboost with svm based weak learners. Proceedings of 2005 IEEE International Joint Conference on Neural Networks **1** (2005) 196–201 vol. 1
34. Hidaka, A., Kurita, T.: Fast training algorithm by particle swarm optimization and random candidate selection for rectangular feature based boosted detector. Proceedings of 2008 IEEE International Joint Conference on Neural Networks (2008) 1163–1169
35. Rasolzadeh, B., Petersson, L., Pettersson, N.: Response binning: Improved weak classifiers for boosting. In: IEEE Intelligent Vehicles Symposium (IV2006). (June 2006) 344– 349
36. Bradski, G., Kaehler, A., Pisarevsky, V.: Learning-based computer vision with Intel's open source computer vision library. Intel Technology Journal **9**(1) (May 2005) 119–130
37. Sierra, A., Echeverria, A.: Evolutionary discriminant analysis. IEEE Transactions on Evolutionary Computation **10(1)** (Feb 2006) 81–92
38. Cagnoni, S., Mordonini, M., Sartori, J.: Particle swarm optimization for object detection and segmentation. In: EvoWorkshops. Volume 4448 of Lecture Notes in Computer Science. (2007) 241–250
39. Metz, C.E.: ROC methodology in radiologic imaging. Investigative Radiology **21**(9) (1986) 720–732