Neurocomputing 113 (2013) 27–35

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Classification-based learning by particle swarm optimization for wall-following robot navigation

Yen-Lun Chen [a], Jun Cheng [a], Chuan Lin [a], Xinyu Wu [a,b,*], Yongsheng Ou [a,b], Yangsheng Xu [a,b]

[a] Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[b] Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 26 July 2012
Received in revised form
21 December 2012
Accepted 30 December 2012
Communicated by R. Tadeusiewicz
Available online 1 March 2013

Keywords:
Multi-category classification
Particle swarm optimization
Wall-following robot navigation

ABSTRACT

In this paper, we study the parameter setting for a set of intelligent multi-category classifiers in wall-following robot navigation. Based on the swarm optimization theory, a particle selecting approach is proposed to search for the optimal parameters, a key property of this set of multi-category classifiers. By utilizing the particle swarm search, it is able to obtain higher classification accuracy with significant savings on the training time compared to the conventional grid search. For wall-following robot navigation, the best accuracy (98.8%) is achieved by the particle swarm search with only 1/4 of the training time by the grid search. Through communicating the social information available in particle swarms in the training process, classification-based learning can achieve higher classification accuracy without prematurity. One of such learning classifiers has been implemented in SIAT mobile robot. Experimental results validate the proposed search scheme for optimal parameter settings.

1. Introduction

Classification-based learning represents a machine learning technique [1,2] with successful applications in several domains, such as biometrics, bioinformatics, and multimedia information management [3–6]. In recent years, mobile robots [7] have become widely applied to human daily life with the advancement of technology and the enhancement of digital information. The capability to interact with people is essential for robots that perform tasks in cooperation with humans, such as service and surveillance robots. In mapless navigation, the system uses no explicit representation about the space in which navigation is to take place [8], where the robot motions are determined by observing and extracting relevant information about the elements (such as walls, desks or doorways) in the environment, and navigation is carried out with respect to these elements. To navigate, the robot uses sensors to calculate the distance of the objects. Many sensors have been used to find the objects, including infrared sensor, laser range finder, visible-light camera, and ultrasonic sensors [9]. For example, SIAT mobile robot uses ultrasonic sensors to avoid collision as shown in Fig. 1.

Wall-following navigation is a kind of motion that the robot moves along the wall in a certain direction, or more generally, moves along the exterior of objects while keeping a safe distance away from objects [10]. When it is combined with other high-level intelligent behavior, the robot can accomplish complex tasks [11]. Recently, intelligent controllers developed using artificial neural network, fuzzy logic, genetic algorithms, or a combination thereof are appealing to deal with such complicated systems. Modeling a human expert control strategy (HCS) [12] with learning-based algorithms is a fine solution for the control of dynamic systems with unstructured uncertainties and fast-changing un-modeled dynamics. However, it is difficult to model the human expert control strategy. In such a case, collecting the needed training data to build a sufficiently accurate learning model for an intelligent classifier is one of the promising solutions. For the wall-following robot navigation, empirical evaluation of this task is a problem of pattern recognition and could be modeled as multi-category classifications. Extending a classifier from binary to multiple categories is still an ongoing research topic [13,14]. Generally, a single multi-class problem is considered as a collection of multiple binary problems. The problem is interpreted as a cascade of hierarchical binary-tree classifiers. Therefore, multi-category classification is considered in the framework of hierarchical-cascaded trees, for parallel and easy implementation with hardware circuits to satisfy the real-time requirement.

When training the classifiers for high accuracy, it is essential to select the parameter of each binary classifier at each node in the hierarchy. Conventionally, grid search is the approach taken. Between the minimum and maximum values of each parameter,

* Corresponding author at: Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China.
E-mail address: xy.wu@siat.ac.cn (X. Wu).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.037

Fig. 1. SIAT mobile robot navigates in an indoor environment.

the method divides the range into several grid-like corresponding dots to select the optimal point. However, doing a complete grid-search may be time-consuming, especially for a large amount of parameters. In training classifiers with low computational complexity, a simple grid-search method can meet the needs for practical applications; however, in training classifiers with high computational complexity, instead of performing exhaustive search for the best parameters, sample-based optimizations such as particle swarm methods [15,16] are proposed to efficiently select the parameters. Particle swarm optimization (PSO) is a parallel algorithm originally developed by Kennedy and Eberhart based on a metaphor of social interaction [17,18]. It is a powerful and easily implemented algorithm to solve optimization problems, especially in a multidimensional vector space. The PSO algorithm is initialed with a population of random candidate solutions, each of which is called a "particle", with a randomly assigned velocity and position. Then, each particle is attracted stochastically towards the location of its own previous best fitness and the best fitness of its neighbors.

When particles are unable to escape from a local optimum after hundreds or thousands of iterations, there exist invalid iterations in PSO [19,20] during the optimization of various continuous functions. The problem is mainly because that the algorithm lacks an effective scheme to escape from the local optimum. To overcome the premature property of PSO, a hybrid PSO (HPSO) algorithm is proposed by integrating the basic PSO with local optimization of pattern search. In each iteration, particles are selected with a probability, where pattern search is performed and the original particles are replaced to improve the precision of convergence if the new fitness value is better. HPSO enables the parameter optimization to possess parameter-fitness capability both globally and locally such that it can be more effective than the basic PSO to achieve an optimal performance. Using the HPSO algorithm, optimal parameters can be efficiently selected via reducing the complexity iteratively and avoiding irrelevant calculations.

The paper is organized as follows. Section 2 describes the multi-category classification and our problems in this paper. The particle-based search approach to evaluate parameters of classifiers are presented in Section 3. In Section 4, the effectiveness of the proposed method is illustrated via a simulation study based on the data from a mobile robot. Finally, we close the paper by stating the conclusions in Section 5.

2. Problem statement

In this section, we will first introduce the concept of binary classification and then multi-category classification.

2.1. Binary classification

We now introduce the binary classification concept. Suppose that we are given n training samples $(\mathbf{x}_l,y_l)$, where $\mathbf{x}_l \in R^d$ and $y_l \in \{+1,-1\}$. For example in the support vector machine (SVM) [21], a hyperplane $\mathbf{w}^T\mathbf{x}+b=0$ could be obtained to separate the samples of different classes on the two sides of hyperplane, where $\mathbf{w}$ is the norm vector and b is the bias of the hyperplane. When the training samples are linearly separable, support vector machine yields the optimal hyperplane that separates two classes without training error, where optimization is in the sense of maximizing the minimum distance from training samples to the hyperplane, or equivalently minimizing $\|\mathbf{w}\|^2$. For linearly non-separable cases, the concept of a separating hyperplane is generalized by employing the slack variable ξ_i with potential training errors. In other words, the parameter pair $(\mathbf{w},b)$ corresponding to the optimal hyperplane is a solution to the following optimization problem. Mathematically, it can be written as

$$\begin{aligned} \min: \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_l \xi_l, \\ \text{subject to}: \quad & \begin{cases} y_l(\mathbf{w}^T\phi(\mathbf{x}_l)+b) \geq 1-\xi_l, \\ \xi_l \geq 0, \end{cases} \end{aligned} \tag{1}$$

where C is a parameter to adjust the weight of training errors in the minimization. When this linear separation is not possible in the input space $\mathcal{X}$, a transformation $\phi: \mathcal{X} \to \mathcal{F}$ can be used to project the data to a high-dimensional feature space $\mathcal{F}$, where the prospect of finding a linear separating hyperplane is higher.

2.2. Multi-category classification

Based on the binary classifier, the concept is extended from binary to multiple classes. Mathematically, we want to find decision boundaries in vector space R^d based on the information from training samples $\mathbf{x}_l$ to separate k different classes.

Given training samples $(\mathbf{x}_l,y_l)$, where $\mathbf{x}_l \in R^d$ is a training sample and $y_l \in \{1,2,\ldots,k\}$ is the corresponding class label. One-versus-rest (OVR) approach constructs k binary classifiers, each of which separates one class from all the rest. The ith binary classifier is trained with all the training examples in the ith class with positive labels, and all the others with negative labels. For example in SVM, the ith binary classifier solves the following problem which yields the ith decision function $f_i(\mathbf{x}) = \mathbf{w}_i^T\phi(\mathbf{x})+b_i$:

$$\begin{aligned} \min: \quad & \frac{1}{2}\|\mathbf{w}_i\|^2 + C_i\sum_l \xi_l^i, \\ \text{subject to}: \quad & \begin{cases} \tilde{y}_l^i(\mathbf{w}_i^T\phi(\mathbf{x}_l)+b_i) \geq 1-\xi_l^i, \\ \xi_l^i \geq 0, \end{cases} \end{aligned} \tag{2}$$

where $\tilde{y}_l^i = 1$ if $y_l = i$ and $\tilde{y}_l^i = -1$ otherwise. The $k \times k$ matrix form of OVR could be interpreted as

$$\tilde{Y} = \begin{pmatrix} + & - & \cdots & - \\ - & + & \ddots & \vdots \\ \vdots & \ddots & \ddots & - \\ - & \cdots & - & + \end{pmatrix}. \tag{3}$$

In the classification phase, $\mathbf{x}$ is classified as the class i^* which has the largest value of the decision function

$$i^* = \arg\max_{i=1,\ldots,k} (\mathbf{w}_i^T\phi(\mathbf{x})+b_i). \tag{4}$$

Different from the one-versus-all method, one-versus-one (OVO) approach constructs $k(k-1)/2$ binary SVM classifiers, each of which separates two of k classes. Let $f_{i,j}(\mathbf{x}) = \mathbf{w}_{i,j}^T\phi(\mathbf{x})+b_{i,j}$ denote the decision function between class i and class j, which

mathematically is the solution to the following problem:

$$\min: \quad \frac{1}{2}\|\mathbf{w}_{i,j}\|^2 + C_{i,j}\sum_l \xi_l^{i,j},$$
$$\text{subject to}: \quad \begin{cases} \tilde{y}_l^{i,j}(\mathbf{w}_{i,j}^T\phi(\mathbf{x}_l)+b_{i,j}) \geq 1-\xi_l^{i,j}, \\ \xi_l^{i,j} \geq 0, \end{cases} \tag{5}$$

where $\tilde{y}_l^{i,j}=1$ if $y_l=i$ and $\tilde{y}_l^{i,j}=-1$ if $y_l=j$. The $k\times C_2^k$ matrix form of OVO can be interpreted as

$$\tilde{Y}=\begin{pmatrix} + & + & \cdots & & 0 \\ - & 0 & & 0 & \vdots \\ 0 & - & & + & 0 \\ \vdots & 0 & & 0 & + \\ 0 & & \cdots & - & - \end{pmatrix}, \tag{6}$$

where each column is a classifier with $+/-$ indicating the class label and 0 indicating no use of those samples in training the classifier. The voting scheme, or the "Max Wins", can be used in the classification phase. If the function $f_{i,j}(\mathbf{x})=\mathbf{w}_{i,j}^T\phi(\mathbf{x})+b_{i,j}$ decides $\mathbf{x}$ belongs to the ith class, then the vote for the ith class is increased by one. Otherwise, the jth class gets one vote. Finally, $\mathbf{x}$ is classified to the class with the most votes.

The training phase of the directed acyclic graph (DAG) [22] approach is the same as the OVO method by solving $k(k-1)/2$ binary problems. However, in the testing phase, DAG operates on a list initialized with all classes, where each node eliminates one class from the list. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. Each node is a binary classification problem of ith and jth classes. If the node prefers one of the two classes, the other class is eliminated from the list, and DAG proceeds to test the first and last elements of the new list. Starting at the root node, the DAG algorithm terminates when only one class remains in the list. Each test instance goes through a path before reaching a leaf node which indicates the predicted class.

3. Parameter selection via particle swarm optimization

In this section, parameter search by particle swarm optimization is presented to overcome the inefficient problem of grid search. As illustrated above, the number of parameters in multi-category classification, including weighting factor C_i and Gaussian kernel width σ_i, increases linearly in the method of OVR, and quadratically in OVO and DAG when the number of classes increases. If the grid search is used for parameter selection in training, the computational complexity increases exponentially as the number of parameters increases. It is not only inefficient but also inapplicable when either the number of classes or the number of samples is large.

To overcome the premature property of the conventional particle swarm optimization (PSO), a hybrid PSO (HPSO) algorithm is proposed by integrating the basic PSO with local optimization of pattern search. Convergence speed and convergence precision are the two important indicators of optimization performance, closely related to the setting of parameters in an algorithm. While considering the setting of parameters, the two indicators are conflicting; that is, in order to obtain a more precise convergence value, it will be at the cost of extending the time of convergence; on the other hand, to increase the optimization speed, it will usually be trapped in a local optimum solution. Since the pattern search method is very effective in local search, it is chosen to intensify the small-region search, which avoids early convergence at a local optimum value to achieve higher precision of convergence while increasing the speed of convergence.

3.1. Basic particle swarm optimization

The particle-based model consists of a swarm of particles, each of which is considered as a social individual. At the beginning, each particle is assigned a random position $\mathbf{z}$ and a random velocity $\mathbf{v}$ in the n-dimensional search space. Then at each time step t, a particle evaluates its fitness at the current position with a fitness function. Moreover, the particle records its own best position so far as $\mathbf{p}_t$. After that it adjusts velocity and position by $\mathbf{p}_t$ and the global best position $\mathbf{g}_t$ obtained through communication with its immediate neighbors. This information flow is obtained by defining a neighbor topology on the swarm. In every iteration t, a particle is updated through the following formula:

$$\mathbf{v}_{t+1}=w\mathbf{v}_t+c_1r_1(\mathbf{p}_t-\mathbf{z}_t)+c_2r_2(\mathbf{g}_t-\mathbf{z}_t), \tag{7}$$

$$\mathbf{z}_{t+1}=\mathbf{z}_t+\mathbf{v}_{t+1}, \tag{8}$$

where r_1 and r_2 are the random numbers evenly distributed between (0,1); c_1 and c_2 are the learning factors, and mostly $c_1=c_2=2$; w is the inertia weight.

In an improved particle swarm algorithm [23], the inertia weight decreases linearly in every iteration; therefore, at the beginning, particles tend to keep their previous speeds for exploration, and follow the trend of optimization later. The formula for linear decrement is as below:

$$w=w_{start}-\frac{w_{start}-w_{end}}{t_{max}}*t, \tag{9}$$

where t_{max} is the maximum number of iteration; w_{start} and w_{end} are the weights at the beginning and at the end, respectively.

The fitness function is defined as the classification accuracy in the parameter selection of multi-category classifications, and the particle position is defined as $\mathbf{z}=[C_1,\ldots,C_k,\sigma_1,\ldots,\sigma_k]$. For example in Fig. 2, the contour lines show the classification accuracy of *heartscale* [24] when $k=1$. The horizontal axis represents parameter C and the vertical axis represents parameter σ, respectively. Position vector $\mathbf{z}=[C,\sigma]$ is in the two-dimensional parameter space. The particle swarm algorithm only needs two iterations to achieve the optimal value with 20 particles. On the contrary, the grid method tries all possible combinations in the range between the minimum and maximum values of the parameter space. Exhaustive search must be done before the optimal value

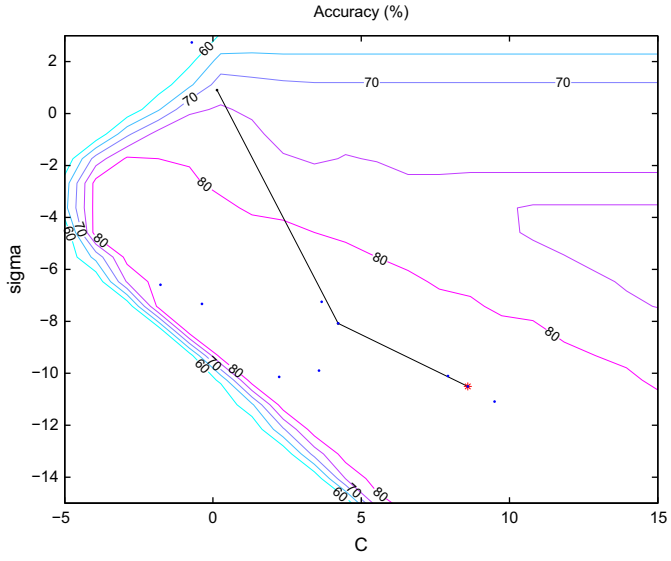

Fig. 2. The contour lines show the fivefold cross-validation accuracy of *heartscale* in two parameters C and σ (log scale, base 2). With 20 particles, the particle swarm algorithm only needs two iterations to achieve the optimal value.

could be determined, which is especially inefficient in training a time-consuming model. Therefore, the particle-based algorithm significantly reduces the computational complexity of parameter search to achieve the goal of efficient parameter optimization.

3.2. Principle of the hybrid particle swarm algorithm

The precision of particle-based algorithm depends on the initial position of particles to some extent. Firstly, this algorithm initializes the position of every particle according to the uniform distribution to enable particles evenly distributed in the entire solution space and to ensure that there exist particles in the nearby area of the optimum solution. Secondly, perform local optimization for part of initial particles with a given probability by using the pattern search method, so that the obtained particles will replace the original ones to change the positions of the initial particles by relocating the initial particles in the nearby area of the optimum value.

In every iteration, the particles update themselves through tracing individual and global best values, so the individual and global best values have direct impacts on the speed and precision. In this paper, the algorithm performs local optimization by using the pattern search method while selecting the individual and global best values with a probability P as initial values; the obtained particles with preferable fitness will replace the initial particles as new individual and global best values and can be used for updating the positions of particles in the next iteration, which can save numerous invalid iterations. Herein, the probability P of the pattern search method is selected based on the strategy of linear decrement; so that, in the beginning, a more precise function value may be obtained faster. Then, in the later phase, because the algorithm has reached a better precision, it is able to cut down unnecessary adoption of the pattern search to improve the execution efficiency. The decrement formula of probability P is as below [25]:

$$P = P_{start} - \frac{P_{start} - P_{end}}{t_{max}} * t, \tag{10}$$

where t_{max} is the maximum number of iteration; t is the current number of iteration; P_{start} and P_{end} are probabilities based on the pattern search method at the beginning and at the end, respectively.

3.3. Parameter analysis of the hybrid particle swarm algorithm

In the hybrid particle-based algorithm, the parameters are swarm scale, learning factors c_1 and c_2, inertia weight w, maximum velocity v_{max}, minimum velocity v_{min} and the pattern search probability P, which are analyzed as follows.

(1) *Swarm scale*: Usually, 20–200 particles are selected. Expanding the scale will improve the probability of successful global optimization, but also increase the execution time of the algorithm.

(2) *Learning factors c_1 and c_2*: Importance degree on the attraction of individual and global best values to every particle. If $c_1 = c_2 = 0$, particles will fly at the current speed until they arrive on the border. If so, limited area is searched and it is difficult to find the optimal solution. If $c_1 = 0$, particles lack the ability of self-cognition; despite the fast convergence speed, the algorithm will easily be trapped into the local best value. If $c_2 = 0$, where social information sharing is unavailable among particles, it is very unlikely to obtain the optimal solution.

(3) *Inertia weight w*: Inertia that keeps particles moving. A higher weight may accelerate global search, yet the algorithm will easily be trapped into a local optimal value. On the contrary, a lower weight may accelerate local search, yet it will increase the number of iterations when finding the global optimal value.

(4) *Maximum velocity v_{max} and minimum velocity v_{min}*: The range of particle displacements in a cycle are determined by the maximum and minimum velocities, which are usually set as the scale width. If the range is too small, particles move very slowly, which will cause longer execution time of the algorithm; if it is too large, particles move very quickly, which will be unfavorable for the convergence.

(5) *Pattern search probability P*: It decides the overall performance of particle swarm and the convergence precision. If the value is too small or equal to 0, it returns to the standard algorithm. If the value is relatively large, high-precision convergence may be obtained. However, it will lead to unnecessary adoption of pattern search method in the later phase, so that the execution efficiency will be reduced.

3.4. Realization of the hybrid particle swarm algorithm

*Step*1: Initialization, including learning factors c_1 and c_2, inertia weight w, maximum number of iteration t_{max}, pattern search probabilities at the beginning P_{start} and at the end P_{end}. Then, in the defined n-dimensional space, randomly generate m particles according to the uniform distribution and form the initial particle swarm; create initial velocity v of individual particles.

*Step*2: Randomly select particles with the probability P while performing local optimization for them by the pattern search method; then replace the initial particles with the newly obtained particles.

*Step*3: Particle swarm evaluation, i.e. calculate the fitness of every particle and re-initialize the worst particles.

*Step*4: Update the flying speed of the particle swarm according to the formula of particle speed; then, update the position of the particle swarm according to the formula of particle position; finally, produce new particle swarm.

*Step*5: Select the individual and global best particles of the particle swarm with the probability P in every iteration, while performing local optimization using the pattern search method to produce new particles; if the new particles have better fitness than the initial individual and global best value particles, replace them with the new particles to serve as a basis for updating the flying speeds and positions of the particles in the next iteration.

*Step*6: Check the stop condition. If it is met, end search; otherwise, $t = t+1$, return to step 3.

The above procedure is summarized in the flow chart of Fig. 3.

4. Experimental study

In this section, experimental results are provided to illustrate the analysis and estimation of particle-based parameter selection.

4.1. Experimental description

The performance of multi-category classifications was investigated on the *wall-following robot navigation* data set from the UCI machine learning repository [26]. The data were collected as the mobile robot navigates through the room following the wall in a clockwise direction, for four rounds. By commanding the robot to perform 4 full rounds around the room, the sensory data collection was performed with a reading rate of 9 samples per second for each sensor, resulting in 5455 samples as shown in Table 1. To navigate, the robot uses 24 ultrasound sensors arranged circularly around its waist, where the numbering of the ultrasound sensors starts at the front of the robot and increases in clockwise direction. There are three different data sets in the

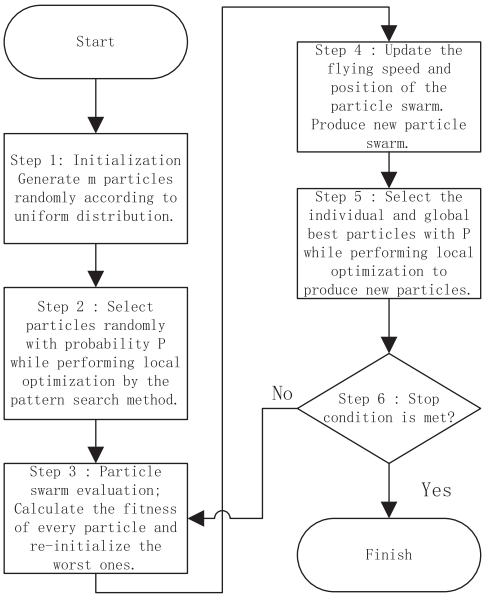

Fig. 3. Flow chart of the hybrid particle swarm algorithm.

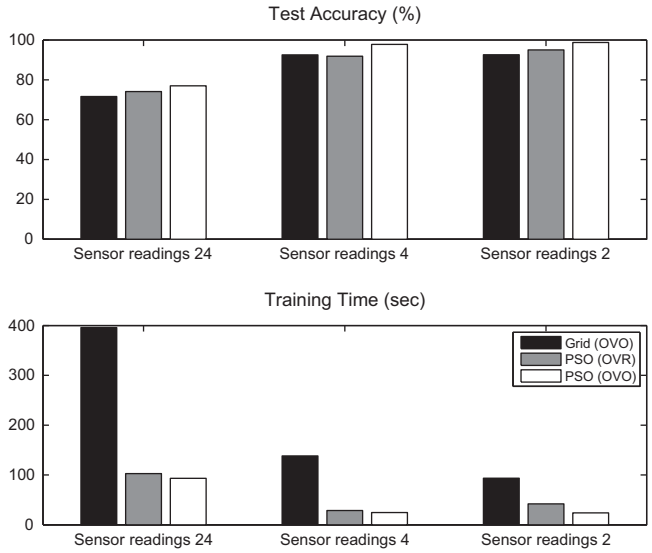

Fig. 4. Training time and test accuracy of the wall-following robot navigation.

Table 1
Class label distribution of UCI wall-following robot navigation.

One-versus-all classifier				Action	Samples	Percentage
f_1	–	–	–	Move-forward	2205	40.41
–	f_2	–	–	Slight-left-turn	328	6.01
–	–	f_3	–	Slight-right-turn	826	15.13
–	–	–	f_4	Sharp-right-turn	2097	38.43

provided repository. The first data set contains the raw values of the measurements of all 24 ultrasound sensors and the corresponding class label. The second one contains four simplified-distance sensor readings and the corresponding class label; these simplified distances are referred to as the front distance, left distance, right distance and back distance, which consist of the minimum sensor readings among those within 60° arcs located at the front, left, right and back parts of the robot, respectively. The third data set contains only the front and left simplified distances and the corresponding class label.

The first quarter of the samples, which correspond to the first full round of the navigation, is used for training and the other samples are used for testing. In the training phase, fivefold cross-validation is used to prevent the overfitting problem [24], where the training set is first divided into five subsets of equal size, and sequentially one subset is tested using the classifier trained on the remaining four subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. After training on cross-validation, the optimal parameters of the classifiers in the training phase and the decision function are then applied on the test samples to obtain the classification accuracy.

The software adopted for simulation in the laboratory is Matlab 7.10, and the computer configuration is AMD Phenom(tm) Q9600B 2.30 GHz Windows PC with 4 GB RAM. In Fig. 4, the training time and test accuracy of wall-following robot navigation are compared between the particle-based search and the grid search. In the one-versus-rest (OVR) method, k classifiers are constructed to separate from the rest of the classes, and in the one-versus-one (OVO) method, $k(k-1)/2$ pairs of classifiers are constructed to separate each class from another one. The decision function is determined by combining the results. In the grid search, $C_i \in [2^{-5},2^{-4},\ldots,2^{15}]$ and $\sigma_i^2 \in [2^{-4},\ldots,2^{14}]$. In the PSO search, learning factors $c_1 = c_2 = 2$; particle swarm number 20; maximum flying speed $v_{max} = 1$; minimum flying speed $v_{min} = -1$; r_1 and r_2 are the random numbers between [0, 1]; the inertia weights $w_{start} = 0.9$ at the beginning and $w_{end} = 0.4$ at the end, respectively. By using the PSO search, it is able to obtain higher test accuracy with significant savings on the training time compared to the grid search. In the case of the sensor readings 2, the best accuracy 98.8% is achieved by the PSO search with only 1/4 of the training time by the grid search.

A receiver operating characteristic (ROC) [27] graph depicts relative tradeoffs between benefits (true positives) and costs (false positives), in which true positive rate is plotted on the vertical axis and false positive rate is plotted on the horizontal axis. Considering that ROC curves are mainly used in detection or verification applications, the ROC curve is defined in a rejection-classification problem as the curve depicting the correct classification rate P_C vs. false alarm rate P_{FA}, where P_C is defined as the percentage of target samples correctly classified in the target class. In Fig. 5, the ROC curves of classifiers f_1, f_2, f_3 and f_4 are plotted accordingly. It could be observed that given a false positive rate, the true positive rates are higher based on the sensor readings 2 than those based on the sensor readings 4 and 24.

The area under the ROC curve (AUROC) indicates the accuracy of a classifier in the way that larger AUROC indicates better accuracy. In Fig. 5, the AUROC of sensor readings 24 are $A_{f_1} = 0.868$, $A_{f_2} = 0.965$, $A_{f_3} = 0.940$, $A_{f_4} = 0.950$, the AUROC of sensor readings 4 are $A_{f_1} = 0.953$, $A_{f_2} = 0.997$, $A_{f_3} = 0.965$, $A_{f_4} = 0.999$, and the AUROC of sensor readings 2 are $A_{f_1} = 0.993$, $A_{f_2} = 0.998$, $A_{f_3} = 0.990$, $A_{f_4} = 0.999$. From these results, the classifiers based on the sensor readings 2 have better accuracy, which makes sense in the case of clockwise trajectories. However, the advantage of using only the front and left simplified distances instead of using all the other sensors may depend on the direction of navigation. In order to make the system be more general and practical, both clockwise and counterclockwise directions are performed in the next experiment.

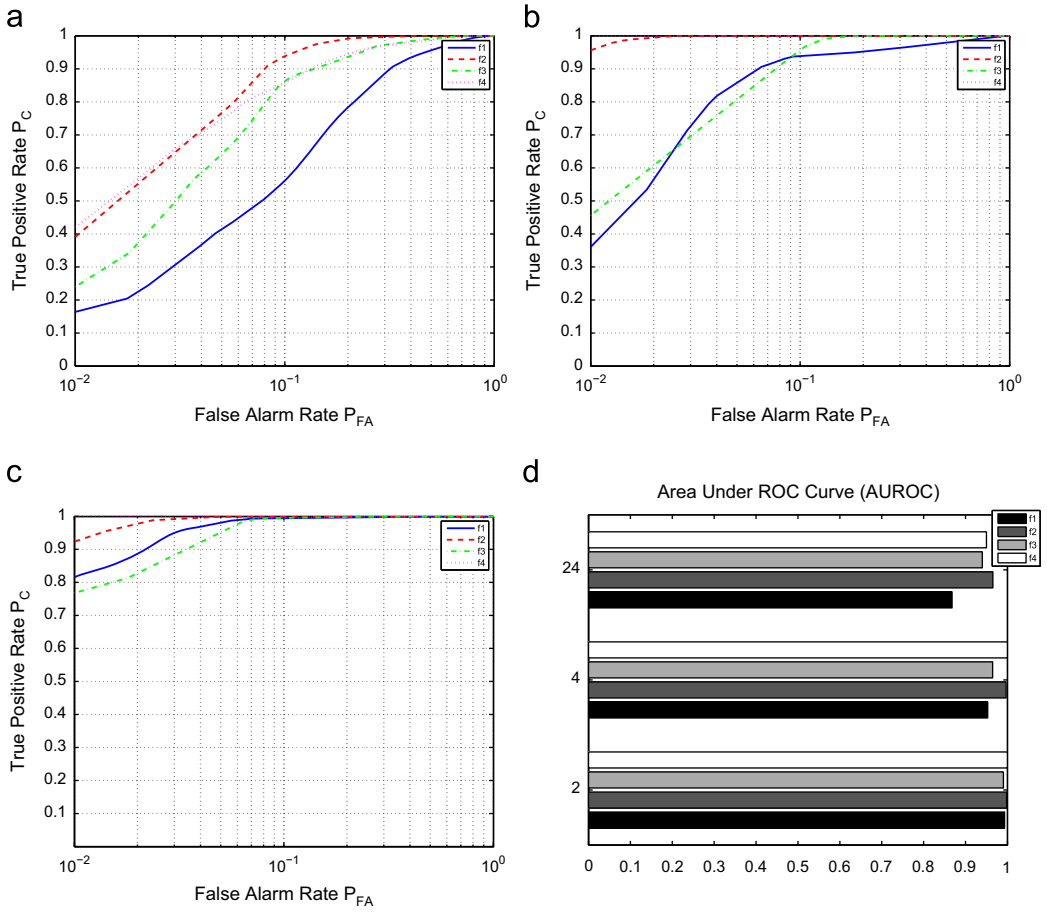

Fig. 5. ROC curves of the UCI wall-following robot navigation: (a) sensor readings 24, (b) sensor readings 4, (c) sensor readings 2, and (d) the area under ROC curves (AUROC).

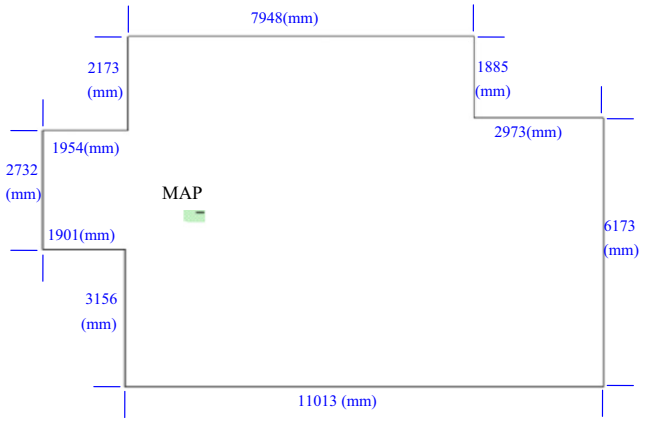

Fig. 6. The map used for Pioneer 3 DX wall-following navigation.

4.2. Experiments on the wall-following data from pioneer 3 DX

The data were collected as the Pioneer 3 DX mobile robot navigates through the room following the wall in both clockwise and counterclockwise directions. The sonar sensory data collection was performed by commanding the robot to perform four full rounds around the room as shown in Fig. 6, and the clockwise/counterclockwise trajectories are shown in Fig. 7a and b, respectively. Five controlling commands, including sharp left-turn, slight left-turn, forward, slight right-turn, and sharp right-turn, are used in the navigation.

In Tables 2 and 3, experimental results of PSO and HPSO on OVR, OVO and DAG classifications are listed for clockwise and counterclockwise trajectories, respectively. The table shows the prediction result in a specific layout that allows visualization of the performance of a supervised learning algorithm. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. All correct predictions are located in the diagonal of the table to easily identify classification errors, as the errors will be represented by any non-zero values outside the diagonal.

As can be seen, HPSO has more percentage of samples located in the main diagonal with a few percentage of samples misclassified to other classes, which indicates correct classifications in most of the time. Furthermore, the distance of a misclassification sample to the main diagonal is proportional to the level of errors; the closer to the main diagonal indicates lower level of errors. If more levels instead of the current 2 levels of slight and sharp turns, an error-tolerance region could be set as a strip of elements near the main diagonal of the table to be the correct controlling margin.

4.3. Experimental system and the results

Experiments are further investigated on the Pioneer 3 DX platform which is installed as the base of SIAT mobile robot for

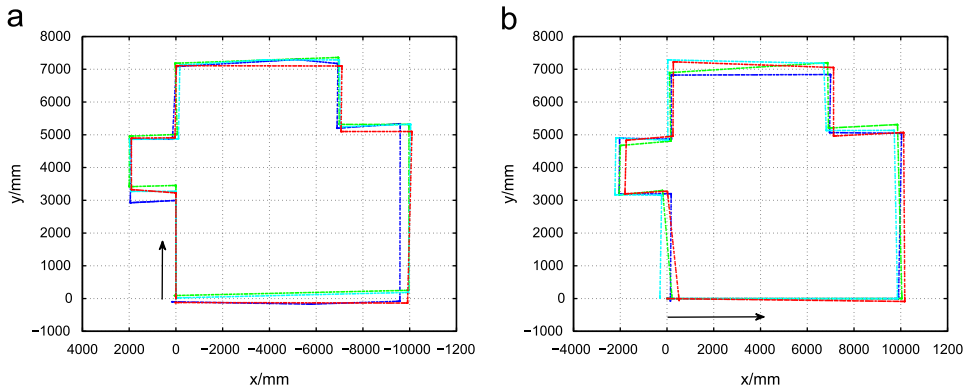

Fig. 7. Pioneer 3 DX mobile robot trajectories. (a) Clockwise and (b) counterclockwise.

Table 2
Prediction percentage (%) of *Clockwise trajectories*, where each row represents a known label and each column represents a predicted label. *L*: sharp left-turn, *l*: slight left-turn, *F*: forward, *r*: slight right-turn, *R*: sharp right-turn.

	PSO						HPSO					
OVR		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**88.0**		8.3	3.7		*L*	**100**				
	l			75	25		*l*		**50**	25	25	
	F	8.6		**61.7**	29.7		*F*	4.7		**92.2**	3.1	
	r	1.3		1.8	**96.9**		*r*			1.1	**98.9**	
	R			100			*R*	80				**20**
OVO		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**93.5**		6.5			*L*	**99.5**				0.5
	l			100			*l*		**75**	25		
	F	8.6		**57.1**	32.0	2.3	*F*	3.1		**95.3**	1.6	
	r			4.0	**94.7**	1.3	*r*		0.1	1.3	**98.6**	
	R	60		40			*R*	20				**80**
DAG		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**93.5**		6.5			*L*	**99.5**				0.5
	l	25		75			*l*		**75**		25	
	F	7.0		**56.3**	34.4	2.3	*F*	3.1		**94.6**	2.3	
	r	0.1		3.4	**95.2**	1.3	*r*		0.1	1.0	**98.9**	
	R	60		40			*R*	20				**80**

Table 3
Prediction percentage (%) of *Counterclockwise trajectories*, where each row represents a known label and each column represents a predicted label. *L*: sharp left-turn, *l*: slight left-turn, *F*: forward, *r*: slight right-turn, *R*: sharp right-turn.

	PSO						HPSO					
OVR		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**45.7**	43.2	6.2		4.9	*L*	**76.1**	18.3			5.6
	l	4.7	**90.4**	4.9			*l*		**98.3**	0.9	0.8	
	F	7.1	13.3	**67.2**		12.4	*F*		3.4	**83.9**	5.1	7.6
	r			100			*r*		27.3		**72.7**	
	R	4.2		0.5		**95.3**	*R*					**100**
OVO		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**79.0**	11.1	9.9			*L*	**95.8**				4.2
	l	11.8	**83.2**	5.0			*l*		**99.5**		0.5	
	F		10.6	**86.7**		2.7	*F*		2.5	**96.6**		0.9
	r			100			*r*		54.5		**45.5**	
	R	1.9	2.3	8.4		**87.4**	*R*					**100**
DAG		*L*	*l*	*F*	*r*	*R*		*L*	*l*	*F*	*r*	*R*
	L	**77.8**	21.0			1.2	*L*	**95.8**				4.2
	l	11.7	**83.4**	4.7	0.2		*l*		**99.5**		0.5	
	F		18.6	**77.0**		4.4	*F*		1.7	**97.5**		0.8
	r		71.4	28.6			*r*		54.5		**45.5**	
	R	1.9	3.7	5.6		**88.8**	*R*					**100**

the wall-following robot navigation. The platform can rotate with zero radius, climb a 25° slope and cross a ditch of 2.5 cm. The speed can reach 1.6 m/s. The mobile base has a motor with 500-tick decoders and can carry out different kinds of robot motion [28]. As the primary means of robot navigation, the frontal sonar ring is composed of eight transducers arranged at angles 90°, 50°, 30°, 10°, −10°, −30°, −50°, and −90°, where the position of each sonar is fixed in the array as shown in Fig. 8. Each sonar comes with its own electronics for independent operation. During normal operations, the sonars fire in sequence from 0 to 7 along the array in Fig. 8.

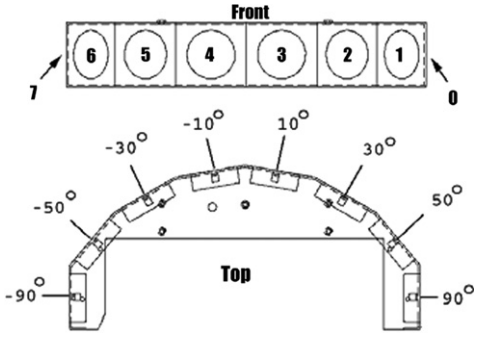

Fig. 8. Sonar configuration (courtesy of ActivMedia Robotics, LLC) of Pioneer 3 DX installed in SIAT mobile robot.

As the experimental scenes shown in Fig. 9, by using the PSO-based parameter selection for learning classifiers, the experiment of robot navigation is successful and SIAT mobile robot can complete wall-following tasks in an indoor environment. The wall-following trajectory is further illustrated in Fig. 10. As can be seen, the whole trajectory is matched with the environment well, except in the right-lower part of the trajectory, where the height of the nearby wall is just a little higher than the position of sonar sensors. For other parts with concrete walls, the robot trajectory is straight and stable from the accurate classification of unambiguous sonar inputs.

5. Conclusions

In this paper, parameter optimizations of intelligent classifiers for wall-following robot navigation are investigated. A hybrid particle scheme is proposed to select the optimal parameters. An experimental study is given to validate the proposed selection approach and discussions therein. Further exploration for methods to embed the search scheme when model re-training is a part of future work.

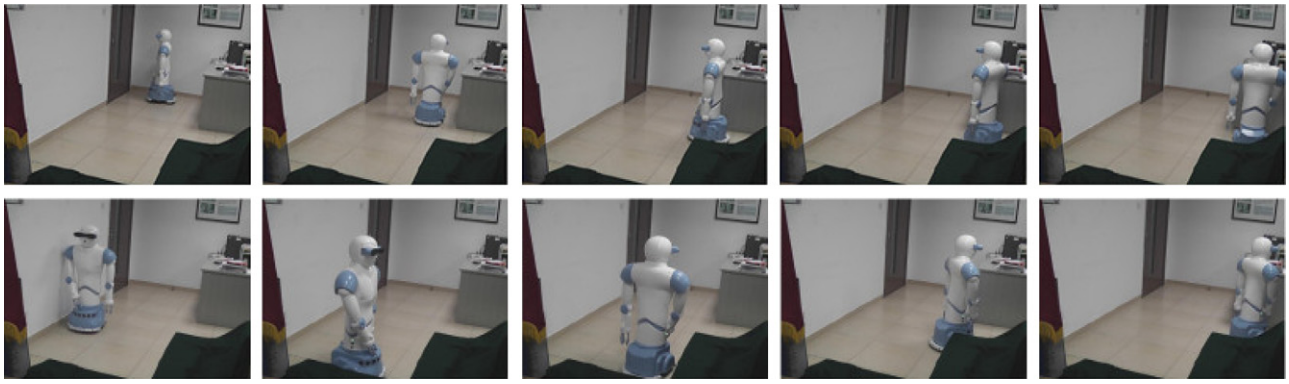

Fig. 9. Scenes of SIAT mobile robot performing wall-following navigation in an indoor experiment.

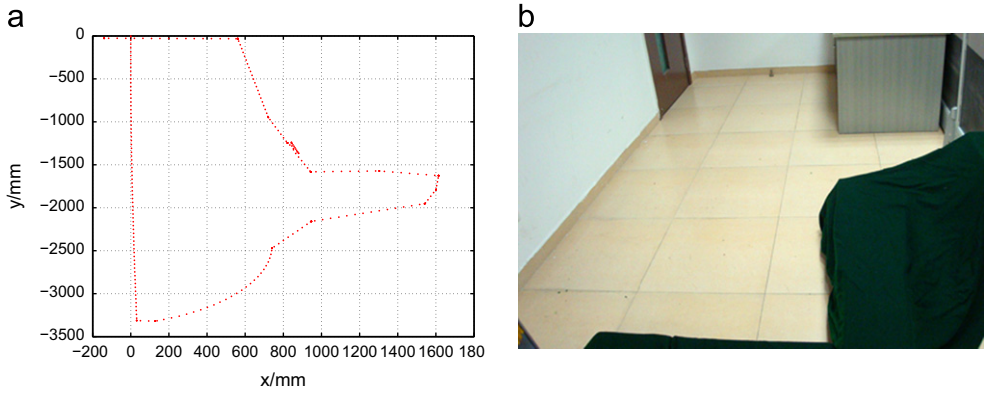

Fig. 10. The trajectory of SIAT mobile robot in the wall-following experiment.

Acknowledgments

This work described in this paper is partially supported by the National Natural Science Foundation of China (61005012), Shenzhen Fundamental Research Program (JC201105190948A and JC201005270365A), and Guangdong Innovative Research Team Program (201001D0104648280). The authors would like to thank Sheng Wang, Ying Liu, Dawei Dai, Xin Kong and anonymous reviewers for their highly constructive suggestions and substantial help.

References

[1] D. Tao, X. Li, X. Wu, S.J. Maybank, Geometric mean for subspace selection, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 260–274.
[2] A.C. Lorena, A.C. de Carvalho, Evolutionary tuning of SVM parameter values in multiclass problems, Neurocomputing 71 (16–18) (2008) 3326–3334.
[3] X. Wang, Z. Li, D. Tao, Subspaces indexing model on Grassmann manifold for image search, IEEE Trans. Image Process. 20 (9) (2011) 2627–2635.
[4] N. Guan, D. Tao, Z. Luo, B. Yuan, Online nonnegative matrix factorization with robust stochastic approximation, IEEE Trans. Neural Networks Learn. Syst. 23 (7) (2012) 1087–1099.
[5] Y. Wang, S. Chen, H. Xue, Can under-exploited structure of original-classes help ECOC-based multi-class classification? Neurocomputing 89 (2012) 158–167.
[6] J. Yu, W. Bian, M. Song, J. Cheng, D. Tao, Graph based transductive learning for cartoon correspondence construction, Neurocomputing 79 (2012) 105–114.
[7] C.-H. Chao, B.-Y. Hsueh, M.-Y. Hsiao, S.-H. Tsai, T.-H. Li, Real-time target tracking and obstacle avoidance for mobile robots using two cameras, in: ICCAS-SICE, 2009, pp. 4347–4352.
[8] G.N. DeSouza, A.C. Kak, Vision for mobile robot navigation: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (2002) 237–267.
[9] A.L. Freire, G.A. Barreto, M. Veloso, A.T. Varela, Short-term memory mechanisms in neural network learning of robot navigation tasks: a case study, in: Proceedings of the Sixth Latin American Robotics Symposium (LARS'2009), 2009, pp. 1–6.
[10] M. Katsev, A. Yershova, B. Tovar, R. Ghrist, S. LaValle, Mapping and pursuit-evasion strategies for a simple wall-following robot, IEEE Trans. Robotics 27 (2011) 113–128.
[11] Y. Liu, R. Fu, J. Wang, Y. Ou, X. Wu, A. Peng, A wall-following strategy for mobile robots based on self-convergence, in: Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 2011, pp. 824–828.
[12] Y. Ou, H. Qian, X. Wu, Y. Xu, On stability region analysis for a class of human learning controllers, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010), 2010, pp. 1303–1309.
[13] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[14] A. Joshi, F. Porikli, N. Papanikolopoulos, Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2995–3002.
[15] X. Zhang, Y. Guo, Optimization of SVM parameters based on PSO algorithm, in: Fifth International Conference on Natural Computation (ICNC), 2009, pp. 536–539.
[16] K. Boese, A. Kahng, Simulated annealing of neural networks: the 'cooling' strategy reconsidered, in: IEEE International Symposium on Circuits and Systems (ISCAS), vol. 4, 1993, pp. 2572–2575.
[17] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[18] S. Shi, H. Rabitz, Quantum mechanical optimal control of physical observables in micro systems, J. Chem. Phys. 92 (1) (1990) 364–376.
[19] K. Xu, L. Zhang, R. Fu, Y. Ou, Y. Xu, A stochastic scattering particle swarm optimizer, in: Proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics, 2010.
[20] L. Zhang, K. Xu, R. Fu, Y. Ou, X. Wu, A stochastic perturbing particle swarm optimization model, in: Proceedings of the 2010 Second WRI Global Congress on Intelligent Systems, GCIS '10, vol. 01, IEEE Computer Society, Washington, DC, USA, 2010, pp. 35–38.
[21] Y. Liu, Y.F. Zheng, One-against-all multi-class SVM classification using reliability measures, in: Proceedings of the International Joint Conference on Neural Networks, 2005, pp. 849–854.
[22] J.C. Platt, N. Cristianini, J. Shawe-Taylor, Large margin DAGs for multiclass classification, in: NIPS, 1999, pp. 547–553.
[23] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE Word Congress on Computational Intelligence, 1998, pp. 69–73.

[24] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 27:1–27:27. software available at 〈http://www.csie.ntu.edu.tw/~cjlin/libsvm〉.
[25] D. Dai, S. Wang, Y.-L. Chen, X. Wu, Y. Xu, A particle swarm optimization algorithm based on the pattern search method, in: International Conference on Computer Design and Applications (ICCDA), vol. 1, 2011, pp. 155–160.
[26] A. Frank, A. Asuncion, UCI Machine Learning Repository, 2010. URL: 〈http://archive.ics.uci.edu/ml〉.
[27] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Lett. 27 (2006) 861–874.
[28] C. Chen, X. Wu, L. Han, Y. Ou, Butler robot, in: Proceedings of the IEEE International Conference on Information and Automation, 2011, pp. 732–737.

Yen-Lun Chen is an Assistant Professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. Chen received her B.S. and M.S. degrees from the Department of Electrical Engineering at National Taiwan University, Taipei, Taiwan, and her Ph.D. degree from Department of Electrical and Computer Engineering at the Ohio State University, Columbus, Ohio, USA. Her research interests include machine learning, pattern recognition, computer vision, and their applications in the area of robotics and multimedia signal processing.

Jun Cheng received Bachelor of Engineering, Bachelor of Finance and Master of Engineering from the University of Science and Technology of China in 1999 and 2002 respectively. His Ph.D. degree was awarded at the Chinese University of Hong Kong in 2006. Currently he is with the Shenzhen Institutes of Advanced Integration Technology, Chinese Academy of Sciences, as a Professor and Director of the Laboratory for Human–Machine Control. His research interests include computer vision, robotics, machine intelligence, and control.

Chuan Lin received her master degree from College of Mechanical and Electrical Engineering, Guilin University of Electronic Technology in 2012. She currently works at the Chinese University of Hong Kong, Shenzhen (CUHKSZ) in China. Her research interests include computer vision and robotics.

Xinyu Wu is a Professor at Shenzhen Institutes of Advanced Technology, and an Associate Director of Center for Intelligent and Biomimetic Systems. He received his B.E. and M.E. degrees from the Department of Automation, University of Science and Technology of China in 2001 and 2004, respectively. His Ph.D. degree was awarded at the Chinese University of Hong Kong in 2008. He has published over 50 papers and a monograph. His research interests include computer vision, robotics, and intelligent system.

Yongsheng Ou is a Professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He holds a B.Sc. degree in Mechanical and Electrical Engineering (Beijing University of Aeronautics and Astronautics, 1995) and an M.Sc. degree in Electrical Engineering (Institute of Automation, Chinese Academy of Sciences, 1998). Ou received a Ph.D. degree in Automation and Computer-Aided Engineering from the Chinese University of Hong Kong in 2004. He is a coauthor of the monograph on Control of Single Wheel Robots (Springer, 2005). His research interests include learning control by demonstration, computer vision, control of biped robots and control of distributed parameter systems with applications to fusion reactors.

Yangsheng Xu is a Professor of Automation and Computer-Aided Engineering at the Chinese University of Hong Kong where he has worked since 1997. He received BSE and MSE from Zhejiang University in China, and Ph.D. from the University of Pennsylvania in US in the area of Robotics. His current research interests are in the areas of robotics, intelligent systems, and electric vehicles.