# Robust Tracking Based on PSO and On-line AdaBoost

Huchuan Lu, Wenling Zhang, Fan Yang
*Department of Electronic Engineering*
*Dalian University of Technology*
*Dalian, 116023, China*
*lhchuan@dlut.edu.cn*

Xiaojing Wang
*School of E-Business*
*Dongbei University of Finance & Economics*
*Dalian, 116023, China*

***Abstract*—Robust tracking is a challenging problem, due to intrinsic appearance variability of objects caused by in-plane or out-plane rotation and extrinsic factors change such as illumination, occlusion, background clutter and local blur. In this paper, we present a novel framework which uses an on-line AdaBoost algorithm as PSO's fitness function to search object and attain the efficiency and robustness of the tracking systems. An on-line AdaBoost classifiers based on RGB features is trained and employed. This new tracking framework, which is initialized with the region of the object at the first few frames, can automatically track the object at the remaining frames. We demonstrate our results under some challenging videos, and the results prove our framework is more robust.**

***Keywords*-Robust tracking; on-line AdaBoost; PSO; RGB features;**

## I. Introduction

The efficient and robust tracking of objects in complicated environments is important for many kinds of applications including video surveillance [1] or human-computer interaction [2]. Thus it is a great challenge to make robust visual tracking methods which can do well with the inevitable variations that can occur in natural scenes such as changes of the shape, changes in the illumination, or partial occlusion of the target. Moreover tracking success or failure may depend on how distinguishable an object is from its background. However if the object is very similar to the background, it requires more distinguished features.

Generally, several approaches have been proposed to fulfill the problem. The idea considering the tracking problem as a classification problem between object and background has lead to further works [3], [4], [5], applying well known classifiers to the tracking problem. In [3] a tracker is realized by using a Support Vector Machine which learns off-line to distinguish between the object and the background. What the work is similar to ours is that of [4]. And we adopt a object tracking method which is based on an on-line AdaBoost algorithm. Some algorithm performs on-line updating of the ensemble features for the target object during tracking and thus is able to cope with appearance changes of the object. Furthermore, the on-line trained classifier uses the surrounding background as negative examples in the update and therefore becomes very robust against the drifting problem [6]. In addition, this negative update allows the algorithm to choose the most discriminative features between the object and the background. Therefore the method can deal with both appearance variations of the object and different backgrounds. Usually the algorithm, which uses RGB color histogram information, is able to run in real-time because training is simply done by updating the model with the positive and negative examples of the current frame.

The particle swarm optimization (PSO) was originally designed by Kennedy and Eberhart [7], [8] and has been compared to genetic algorithms [9], [10] for efficiently finding optimal or near-optimal solutions in large search spaces. The basic idea of the proposed PSO-based algorithm is as follows: particles fly around randomly over the image searching for the best-fit tracking window parameters based on a predefined fitness function of object features. When some particles successfully detect the objects, they will broadcast this object information to their neighbors. Each particle makes its own decision not only based on its neighbors, but also on its own cognition, which provides the flexibility and ability of exploring new areas. This decision-making procedure can efficiently prevent the local optimum effect. The major advantage of this algorithm lies in the fact that the tracking algorithm can be applied to various objects under different environment, such as people in office environments or vehicles in outdoor fields. Even if some objects may get occluded or disappear from the image, the tracking algorithm can still catch up the objects whenever the objects reappear in the image with the real-time performance [11].

In this paper, firstly we select feature information using the RGB value, and train the positive and negative samples with the on-line AdaBoost, secondly to adopt the samples' results via on-line AdaBoost as the particle swarm optimization (PSO)'s fitness function, subsequently object's position is found in the next frame by the two steps. The tracking completes.

The reminder of the paper is organized as follows. In Section 2 we simply review the tracking algorithm and the on-line AdaBoost for feature selection which forms the bases of the approach. Particularly we introduce accurateness of the PSO algorithm. In Section 3 we show experimental results. Finally, we present some conclusion and further

work.

## II. Tracking

### A. On-line Adaboost

The basic idea of on-line Adaboost proposed by [12] is that the importance of a sample can be estimated by propagating it through a series of selectors consisting of a set of weak learners. A selector selects exactly one of a set of *M* weak learners with hypothesis $H^{weak} = \left\{h_1^{weak}, ..., h_M^{weak}\right\}$

$$h^{sel}(x) = h_m^{weak}(x) \tag{1}$$

where $m$ is chosen according to an optimization criterion. In fact, we use the estimated error $e_i$ of each weak classifier $h_i^{weak} \in H^{weak}$ such that $m = \arg \quad \min_i e_i$

Weak learners $H$ correspond to features hence selectors can select from a subset of *M* features $F_{sub} = \{f_1, ..., f_M | f_i \in F\}$ of the feature pool. A fixed set of *N* selectors $h_1^{sel}, ..., h_N^{sel}$ is initialized randomly, each with its own feature pool. When a new training sample $< x, y >$ with an importance weight $\lambda$ arrives, the selectors are updated. The selector selects the weak learner with the smallest error

$$\arg \quad \min_m \left(e_{n,m}\right), \qquad e_{n,m} \ = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}} \tag{2}$$

Finally, the corresponding voting weight $\alpha_n$ and the importance weight $\lambda$ of the sample are updated and passed to the next selector $h_{n+1}^{sel}$.

The procedure is repeated for all selectors. By continuously entering labeled samples, selectors are trained. Therefore, a strong classifier is obtained by linear combination of selectors and used for classification later.

$$h^{strong}(x) = sign\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(x)\right) \tag{3}$$

Details of the algorithm can be found in [12].

### B. PSO algorithm

The particle swarm optimization's basic idea is introduced by [13] in detail. In the PSO algorithm each individual is called a "particle", and is subject to a movement in a multidimensional space that represents the belief space. Particles have memory, thus retaining part of their previous states. There is no restriction for particles to share the same point in belief space, but in any case their individuality is preserved. Each particle's movement is the composition of an initial random velocity and two randomly weighted influences: individuality, the tendency to return to the particle's best previous position, and sociality, the tendency to move towards the neighborhood's best previous position. The velocity and position of the particle at any iteration is updated based on the following equations:

$$v_{id}^{t+1} = v_{id}^t + c_1 * \Phi_1 * (p_{id}^t - x_{id}^t) + c_2 * \Phi_2 * (p_{gd}^t - x_{id}^t) \tag{4}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \tag{5}$$

Where $v_{id}^t$ is the component in dimension d of the i th particle velocity in iteration t, $x_{id}^t$is the component in dimension d of the i th particle position in iteration t, $c_1$,$c_2$ are constant weight factors, In our paper, they have the value 2. $p_{id}^t$ is the best position achieved by particle i, $p_{gd}^t$ is the best position found by the neighbors of particle i, $\Phi_1$ $\Phi_2$ are random factors in the [0,1] interval. According to (4), each particle adjusts its velocity by combing three forces: keeping the velocity of last moment, moving to the best position from its own memory, moving to the best position found by its neighbors. Different parameters in (4) provide varied balance among those three factors. Then a particle moves in the search space according to the combined velocity calculated by (4) to achieve a new position, which can present a new value of object features.

### C. Our approach

In the traditional method, the location of the salient object is confirmed through sliding window search, whereby we obtain a confidence value after evaluating the output of the classifier at each pixel. So that a confidence map of all pixels is generated. We analyze the confidence map and shift the searching window to the new location where the maximal confidence value lies. To find out the maximum, a mean shift procedure[14] can be used. However, the procedure is implemented pixel-by-pixel, which consumes much time. Additionally, the system is very complex.

In our algorithm, traditional object detection method is not necessary because parameters describing the object of interest are assigned manually in the first frame. Firstly the classifiers are trained; we label positive samples from the first few frames, which are still necessary by manual method. We label the positive target window with the red window, and the window's size depends on the size of target which is in the video. At the same time, negative samples are extracted by taking regions of the same size as the target window from the surrounding background by the blue window. These samples are made several iterations through the on-line AdaBoost algorithm. However, in order to obtain a first model. Then we can complete the initialization of the tracker via the training procedure which is on-line AdaBoost procedure. After that the tracking can be completely automated.

We initialize a rectangle window enclosing the region of interest in the first frame which includes the parameters of object's position and scaling. Four parameters are identified to describe the rectangle windows, including 2D location of the central point, width and height of the rectangle. These parameters can build up a four-dimensional search space.

So in such a space, each particle presents a search window with specific values of parameters, which can be defined as: $P = \{P_i | p_i \ (x_i, y_i, l_i, w_i), i = 1, 2, \ldots N\}$

Where $x_i$ and $y_i$ represent the central point of the rectangle related to particle i, $l_i$ and $w_i$ represents the length and width related to particle i, and N is the population of swarm particles. Each individual particle has different values of these parameters. The four parameters will change according to the object's variation at every iteration. So the tracking window will change its size and direction on the process of tracking. Every rectangle window represent a particle, all the particles will be put into the trained on-line AdaBoost classifiers to attain the confidence values.

In order to track, feature extraction is necessary. We use RGB color histogram to describe the color cue, due to its simplicity and swift. In the interests of simplicity, advanced to the next frame, we adopt the particle swarm optimization (PSO) for tracking. We specify the number of particles, which are randomly generated around the object position in the Previous Frame. What is more important, the method reduces search times greatly, compared with the traditional method. For each particle, we extract its corresponding window from the current frame, and calculate its confidence value, which is produced by PSO's fitness function [13].

The choice of PSO's fitness function is very important; also the method is especially novel in this paper. We send the RGB feature of the image block corresponding to the particle into the trained on-line AdaBoost classifier, so that we obtain a confidence value. The confidence values of all particles are regarded as the input of the fitness function.

At every time every particle can shift to find the optimum via some velocity, after that the global optimum will be found. There are some rules can terminate the whole procedure such as setting a threshold value, formulating the times of iteration and the number of particle. Subsequently, the optimal position is found. We regard the sample with the highest confidence as the tracking result and label the image block which is the region of interest.

Our proposed algorithm is easy to implement and runs very fast.

## III. Experimental Results and Discussion

We run our algorithm on a Pentium 2GHz Dual Core PC with 1GB memory. We implement all codes by Matlab 7.0. All testing sequences are 320*240 color images. The RGB color feature for characterize a sample is a 64 (4*4*4) dimensional histogram equally discredited in RGB color space. For on-line Adaboost, the number of selectors and weak learners are both 64, equal to the dimension of RGB feature vector. The number of particles randomly sampled is 20 and the times of iteration are 20. By dispersing particles in a relatively smaller region instead of the whole space, searching procedure can be definitely accelerated. The size of particle window varies following the object's change. And the particles' search radius is 20 pixels in this paper. The training procedure lasts for five or ten frames. Image blocks warped are resized to 32*32, and then used to extract features.

Figure 1. The boy stands up and wags his head greatly.

We apply our approach to several videos, and compare the tracking results with [11] in Figure 1. The comparison we make is of a sequence which contains a boy stands up and wags his head continuously, even rolls his head 90 degrees. Despite large variations, our approach can lock the boy's head all the time. The red boxes are detected object location. Compare with the [11], his tracking window drifts away, it can't find the object well. Moreover, the scale of detection windows alters automatically according to the states of the boy's head.

In the second video, one girl walks in the corridor which has some shadow at the end. When the girl disappears in the shadow and reappears from the shadow, our tracking window can still keep up with the object is shown in Figure 2. The proposed algorithm shows the robustness to keeping tracking the object all over the cases.

## IV. Conclusions and Future work

In this paper, we propose a novel framework which uses an on-line AdaBoost algorithm as PSO's fitness function to searching object and to achieve the efficiency and robustness of the tracking systems. Adopt the PSO algorithm with simple particles search methods instead of sliding window technique, which reduce much time Complexity intensely. Our approach has less strict requirements on the quantity

Figure 2. One girl walks into the shadow and walks out of the shadow.

of labeled samples at the beginning. Especially, it exhibits a satisfying result when significant scale and the position changes of objects occur.

There are still several remained and need to be improved and extended in the future work. It needs to investigate new feature selection approaches that can select the plentiful feature information to train classifier in order to find the optimal position.

## REFERENCES


[1] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *ICCV*, vol. 2, 2003, pp. 734–741.

[2] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, 1998.

[3] S. Avidan, "Support vector tracking," *PAMI*, vol. 2, pp. 1064–1072, 2004.

[4] S.Avidan, "Ensemble tracking," in *CVPR*, vol. 2, 2005, pp. 494–501.

[5] O. Williams, A. Blake, and R. Cipolla, "Sparse bayesian learning for efficient visual tracking," *PAMI*, vol. 27, pp. 1292–1304, 2005.

[6] I. Matthews, T. Ishikawa, and S. Baker., "The template update problem," *PAMI*, vol. 26, pp. 810–815, 2004.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[8] Y. H. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *the 7th Annual Conference on Evolutionary Programming*, San Diego,USA, 1998.

[9] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *the 7th Annual Conference on Evolutionary Programming*, 1998.

[10] P. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance difference," in *the 7th Annual Conference on Evolutionary Programming*, San Diego, USA, 1998.

[11] Y. Zheng and Y. Meng, "Adaptive object tracking using particle swarm optimization," in *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, FL, USA, 2007.

[12] H. Grabner and H. Bischof, "On-line boosting and vision," in *CVPR*, 2006, pp. 260–267.

[13] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resource," vol. 1, 2001, pp. 81–86.

[14] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *ICCV*, vol. 2, 1999, pp. 1197–1203.