Learning Sampling Distributions for Efficient Object Detection

Yanwei Pang, *Senior Member, IEEE*, Jiale Cao, and Xuelong Li, *Fellow, IEEE*

***Abstract*—Object detection is an important task in computer vision and machine intelligence systems. Multistage particle windows (MPW), proposed by Gualdi *et al.*, is an algorithm of fast and accurate object detection. By sampling particle windows (PWs) from a proposal distribution (PD), MPW avoids exhaustively scanning the image. Despite its success, it is unknown how to determine the number of stages and the number of PWs in each stage. Moreover, it has to generate too many PWs in the initialization step and it unnecessarily regenerates too many PWs around object-like regions. In this paper, we attempt to solve the problems of MPW. An important fact we used is that there is a large probability for a randomly generated PW not to contain the object because the object is a sparse event relative to the huge number of candidate windows. Therefore, we design a PD so as to efficiently reject the huge number of nonobject windows. Specifically, we propose the concepts of rejection, acceptance, and ambiguity windows and regions. Then, the concepts are used to form and update a dented uniform distribution and a dented Gaussian distribution. This contrasts to MPW which utilizes only on region of support. The PD of MPW is acceptance-oriented whereas the PD of our method (called iPW) is rejection-oriented. Experimental results on human and face detection demonstrate the efficiency and the effectiveness of the iPW algorithm. The source code is publicly accessible.**

***Index Terms*—Feature extraction, object detection, particle windows (PWs), random sampling.**

I. Introduction

OBJECT detection is a key component of many computer vision systems [24], [41], [42], [44]. Generally, object detection consists of two steps: 1) feature extraction and 2) classification [43]. In this paper, we divide the task of object detection into three steps: 1) window generation; 2) feature extraction; and 3) classification. Window generation outputs windows determined by shape, location, and size. Features are extracted from the windows and then are classified by a classifier. Suppose that N windows are generated and the time spent in generating the windows is t_w. Let t_f be the time of extracting a feature vector from a window (i.e., subimage) and t_c be the time of classifying the feature vector as either positive or negative class. Then the computation time t of an object detection algorithm can be expressed as

$$t = t_w + N \times t_f + N \times t_c. \tag{1}$$

Usually, t_w is very small and can be neglected. Obviously, it is desirable if N is as small as possible on the condition that the detection rate and false positive rate are acceptable, which is the goal of our algorithm.

A dominant manner of window generation is sliding window (SW)-based scanning. Given a pixel stride and a scale factor, windows are determinately generated from top to bottom, left to right, and small to large. This deterministic manner requires a very large number of windows in order to detect objects with high detection rate and low false positive rate.

Windows can also be generated in a stochastic (random) manner. The stochastic manner can also be categorized into active sampling [1]. Recently, Gualdi *et al.* [13] proposed to generate the windows [called particle windows (PWs)] by sampling from a probability density function which is called proposal distribution (PD). This algorithm is called multistage PW (MPW). The initial PD is a uniform distribution, meaning that each candidate window has the same chance to contain the object. If some windows of the N_1 sampled PWs (called PWs in [13]) have large classifier responses, then the PD is updated by enhancing the positions nearby these windows. Consequently, when sampling from the updated PD, more windows will be generated near the previous PWs. Therefore, a smaller number $N_i (N_i < N_1)$ of PWs is needed to be drawn from the updated PD, which is why the MPW method can use a smaller number of windows to get the same detection rate and false positive rate as the SW method.

Despite the great success of MPW, there is still room to improve it. Due to the non-negativity of the weights and density, almost all PWs are sampled from the regions neighboring to the PWs obtained in the previous stages. Therefore, if the number of initial PWs is not large enough to contain true positives, then there is a very large probability that MPW will not sample the positive windows any more. In addition, MPW unnecessarily generates too many PWs around the object and object-like regions. Considering that classification of these regions is more time-consuming than obvious

Manuscript received July 22, 2015; revised October 23, 2015; accepted December 3, 2015. Date of publication January 6, 2016; date of current version December 14, 2016. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2014CB340400, and in part by the National Natural Science Foundation of China under Grant 61172121, Grant 61271412, Grant 61503274, and Grant 61222109. This paper was recommended by Associate Editor J. Basak.

Y. Pang and J. Cao are with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: pyw@tju.edu.cn; caojiale.tju@gmail.com).

X. Li is with the Center for Optical Imagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelong_li@opt.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2508603

nonobject regions in the algorithm of cascade AdaBoost, so generating too many PWs around the object and object-like regions greatly limits its efficiency. Importantly, it is unknown to use how many PWs in each stage.

In this paper, we propose to improve MPW with the aim of sampling a smaller number of PWs without any loss in detection rate and false positive rate. In addition, we solve the problem of determining the number of PWs in each stage. We call the proposed algorithm iPW (see Fig. 2). Compared to MPW, iPW has the following characteristics and advantages.

1) iPW does not need to generate a large number of PWs at the initial iteration (stage), which greatly reduces the detection time. Even if the initial PWs do not contain any object, iPW can detect the objects at next stages.
2) MPW unnecessarily draws too many PWs around the positive windows whereas iPW avoids generating the redundant PWs by using the information of both rejected and accepted PWs.
3) To use MPW, one has to empirically set the number of PWs in each stage whereas this is not a problem because iPW generates a single PW in each iteration (stage).
4) iPW fully makes use of the information of rejected negative PWs while MPW almost completely depends on the accepted positive PWs. Rejected negative PWs are used to directly suppress the PD around these windows and at the same time indirectly enhance the PD beyond the windows. In this sense, iPW is rejection-oriented while MPW is acceptance-oriented.
5) In MPW, the uniform distribution is used in the initialization stage and plays an unimportant role in the later stages so that it can be omitted. In iPW, a dented uniform distribution is used to play an important role for sampling useful PWs by rejecting background (i.e., nonobject) regions.
6) iPW utilizes dented Gaussian distribution while MPW utilizes full Gaussian distribution for sampling PWs. By using dented Gaussian distribution, iPW avoids drawing many unnecessary PWs around the object and object-like regions.
7) To obtain the same detection accuracy, the total number of PWs in iPW is much smaller than that in MPW.

The remainder of the paper is organized as follows. In Section II, related work is discussed. Section III reviews the MPW algorithm. The proposed iPW algorithm is described in Section IV. Experimental results are given in Section V before summarizing and concluding in Section VI.

II. Related Work

According to (1), the computation time of an object detection algorithm is determined by window generation, feature extraction, classification, and the number of windows. Accordingly, we can categorize existing efficient object detection algorithms into feature-reduced, classification-reduced, and window-reduced types. In addition, combination of the different types of algorithms should also be considered.

Cascade AdaBoost plus Haar-like features and integral channel features (ICFs) can be viewed as classical combination of feature-reduced and classification-reduced method [8], [38]. In contrast, neural network-based face detection [31], [35] is time-consuming in feature extraction and classification though it has comparable detection accuracy. Similarly, histogram of oriented gradients (HOGs) plus support vector machine (SVM) [11] can be improved in efficiency by using integral image and cascade structure. The trilinear interpolation of HOG can also be approximated by decomposing gradients into different angle planes followed by a simple smoothing step [29]. There are many efficient object detection algorithms using the technique of integral image for extracting simple but rich features. One important type of features for human detection is ICF [3], [8]. There are also many ICF variants.

Designing optimal cascade structure is also an important topic for increasing the speed of object detection. Recent methods include crosstalk cascade [10], CoBE [6], LACBoost [33], [40], sparse decision directed acyclic graphs [4], etc.

In addition to the integral-image-based algorithms, coarse-to-fine feature hierarchy [9], [46], and template matching with binary representation [14] are also feature-reduced methods. The coarse-to-fine feature hierarchy is able to reject the majority of negative windows by the lower resolution features and process a small number of windows with higher resolution features [46]. By deep analysis of statistic of multiscale features, Dollár *et al.* [9] developed an efficient scheme for computing feature pyramids. Liu *et al.* [22] developed a probability-based pedestrian mask which can be used as prefilter to filter out many nonpedestrian regions. Template matching with binary representation for gradient information is a promising method for detecting textureless objects in real time [14]. Owing to its elegant feature representation and the architecture of modern computers, template matching with binary representation for gradient information can use thousands of arbitrarily sized and shaped templates for object detection in very fast speed [14]. By setting proper sliding stride, features can be reused to avoid computing the features in a window overlapping with its neighbors [28]. The information of spatial overlap can also be used for image matching and recognition [2]. Deep learning with rich features hierarchy is also a promising direction [12].

Classifier-reduced method arrives at high efficiency by designing an efficient classifier. In addition to cascade AdaBoost which uses a few of classifier to reject a lot of windows, one can design efficient linear or nonlinear SVM for classification. Vedaldi and Zisserman [37] proposed to use explicit, instead of implicit, feature maps to approximate nonlinear SVM. Kung [17] and Mak and Kung [23] developed a low-power SVM classifier where the scoring function of polynomial SVMs can be written in a matrix-vector-multiplication form so that the resulting complexity becomes independent of the number of support vectors. A linear SVM classifies a feature vector by computing the inner product between the sum of weighted support vectors and the feature vector. To reduce the computation complexity of the inner-product-based classification, Pan *et al.* [30] proposed a sparse inner-product algorithm. The idea is that neighboring subimages are also neighboring to each other in the feature space and have

similar classifier responses. Pang *et al.* [24] also developed a distributed strategy for computing the classifier response.

Window-reduced method is a promising direction developed in recent several years. This kind of methods aim at reducing the number of windows where feature extraction and classification have to be conducted. When an image is represented by a small number of keypoints and their descriptors (i.e., visual words), branch&bound (also known as efficient subwindow search) is very efficient because it hierarchically splits the parameter spaces into disjoint spaces and uses quality functions to reject large parts of the parameter space [18]. Branch&bound can also be used in implicit shape model which adopts hough-style voting [20], [21]. Branch&rank generalizes the idea of branch&bound by learning a ranking function that prioritizes hypothesis sets that do contain an object over those that do not [19]. Acting testing is also a promising method for rapid object detection [1], [34]. But these visual-word-based methods are not suitable for detecting textureless objects because it is not reliable to detect keypoints from the objects.

As a signal can be reconstructed from irregularly sampled points [25], an object can be detected by randomly sampling a fraction of all the possible locations and scales. MPW is a window-reduced object detection method [13] using the random sampling technique. Branch&rank and branch&bound are based on keypoint detection whereas MPW extracts Haar-like features, HOG, or other features as the same manner of SW based object detection method. Therefore, MPW is expected to be suitable for detecting both texture-rich and texture-less objects. SW-based method investigates all the windows overlappingly and uniformly spaced in spatial and scale domains. In contrast, MPW only checks the windows generated from an updated PD. As iteration proceeds, the main peaks of the distribution evolve toward the objects. The open problem in MPW is how many windows should be generated in each iteration (stage). Existing MPW uses empirical numbers which is hard to result in an optimal solution. In this paper, we propose a novel PW-based object detection method that is more efficient than MPW and is able to avoids choosing particle numbers in multiple stages.

It is noted that the technique of detection proposals (DPs) [47] (sometimes called objectness [7] or selective search [36]) also generates a number of windows by sampling from all the possible windows. But the number of generated windows has to be large (e.g., 10^3 or 10^4) enough if acceptable detection quality is required. Moreover, the time spent on generating DP is not satisfying (see [15, Table 2]). Nevertheless, the DP methods have been successfully employed in deep leaning-based detection algorithms [35].

III. Multistage Particle Windows

The algorithm of MPW [13] is the basis of our method.

A. Algorithm

MPW investigates a fraction of all candidate windows in an image by sampling from a PD. Each PW represents a window $\mathbf{w} = (x, y, s)^T$, where x, y, and s are the horizontal position, the vertical position, and the size of the window, respectively.

(a) (b)

Fig. 1. Process of MPW. (a) Stage 1 samples initial PWs by uniform distribution. (b) Stage 2 generates PWs around pedestrians.

The window can also be expressed as $\mathbf{w}(x, y, s)$. Once a window $\mathbf{w}$ is generated, the feature vector is then extracted and classified. Let $f(\mathbf{w})$ be the classifier response. The main issue of MPW is how to design the PD.

At the beginning of the MPW algorithm, no prior knowledge is known about the preference on the candidate windows. So the PD $q_0(\mathbf{w}) = q_0(x, y, s)$ is modeled as a uniform distribution $u(\mathbf{w}) = u(x, y, s) = 1/N$, where N is number of all possible windows in the image.

In the first iteration, N_1 PWs are sampled from the uniform PD $q_0(\mathbf{w}) = u(\mathbf{w})$ [see Fig. 1(a)]. The classifier response $f(\mathbf{w}_i)$ is normalized by

$$f(\mathbf{w}_i) \leftarrow \frac{f(\mathbf{w}_i)}{\sum_{j=1}^{N_1} f(\mathbf{w}_j)} \tag{2}$$

so that $\sum_{i=1}^{N_1} f(\mathbf{w}_i) = 1$, and $0 \le f(\mathbf{w}_i) \le 1$. Then the PD is updated according to the classifier responses $f(\mathbf{w})$ of the N_1 PWs

$$q_1(\mathbf{w}) = (1 - \alpha_1) q_0(\mathbf{w}) + \alpha_1 \sum_{j=1}^{N_1} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma). \tag{3}$$

In (3), $G(\mathbf{w}_j, \Sigma)$ is a Gaussian distribution where the mean is centered at $\mathbf{w}_j$ and Σ is the standard deviation of the Gaussian distribution. The weight α_1 balances the previous PD q_0 and the mixture of Gaussian distributions. Gualdi *et al.* [13] found that $\alpha_1 = 1$ is almost the best choice, meaning the unimportance of previous PD. In Section III-B, we will explain why $\alpha_1 = 1$ is a reasonable choice.

The sum term in (3) is called measurement density function $p_1(\mathbf{w})$ [13]

$$p_1(\mathbf{w}) = \sum_{j=1}^{N_1} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma). \tag{4}$$

If $\alpha_1 = 1$, then the PD $q_1(\mathbf{w})$ is identical to the measurement density function $p_1(\mathbf{w})$.

In the second iteration, N_2 PWs are drawn from $q_1(\mathbf{w})$. Usually, N_2 is smaller than N_1. Because the classifier response $f(\mathbf{w})$ is large in the regions nearby the positives, most of the N_2 PWs lie around the positives [see Fig. 1(b)].

Algorithm 1 Algorithm of MPW

Input:

Stage number S;

The number N_i of particle windows in stage i, $i = 1, \ldots, S$;

The number N of all candidate windows.

Output:

The set $\mathbf{W}_P$ of positive particle windows.

1: **Initialization**
2: Empty the set of positive particle windows: $\mathbf{W}_P \leftarrow \Phi$.
3: Initialize the PD $g(\mathbf{w})$ by the uniform distribution $u(\mathbf{w}) = 1/N$, i.e., $g(\mathbf{w}) \leftarrow 1/N$.
4: **for** $s = 1$ **to** S **do**
5: Sample N_i particle windows from $g(\mathbf{w})$ and put them into $\mathbf{W}$, i.e., $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{N_i}\}$.
6: If $f(\mathbf{w}_j) = 1$, j=1, …, N_i, then $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}_j$.
7: Update $g(\mathbf{w})$ using the $\mathbf{W}$, and empty $\mathbf{W}$.
8: **end for**
9: **return** $\mathbf{W}_P$.

The iteration continues with the new PD in stage i

$$q_i(\mathbf{w}) = (1-\alpha_i)q_{i-1}(\mathbf{w}) + \alpha_i \sum_{j=1}^{N_i} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma_i). \quad (5)$$

If $\alpha_i = 1$, the PD becomes

$$q_i(\mathbf{w}) = p_i(\mathbf{w}) = g_i(\mathbf{w}) = \sum_{j=1}^{N_i} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma_i) \quad (6)$$

which is in fact the employed PD in the experiments in [13].

Algorithm 1 shows the procedure of MPW.

B. Why $\alpha_i = 1$

In this section, we explain why $\alpha_i = 1$ is a reasonable choice. To the best of our knowledge, we are the first to explain why $\alpha_i = 1$.

For the sake of simplicity, we assume that the scale is fixed and there is only one object in the image. For an $h \times w$ image, the number of candidate windows is $M = h \times w$. Let the size of support region be $m \ll M$. In the initialization step, N_1 PWs are drawn from the uniform distribution $q_0(\mathbf{w}) = u(\mathbf{w})$. Then the probability p that the N_1 PWs contain the object is $p = 1 - (1 - m/M)^{N_1}$. Suppose that $M = 640 \times 480$, $m = 50$, and $N_1 = 1000$, then $p = 0.15$. Obviously, if N_1 is small, it is a small probability for the PWs to contain the object.

In the later stage, MPW generates a smaller number of PWs, where $(1 - \alpha_1)$ fraction is from the uniform distribution. The probability that the $(1-\alpha_1)$ fraction of PWs contains the object is much smaller. So α_i is usually set 1.

C. Merits and Limitations

Compared to SW, MPW is able to obtain similar accuracy at lower computational load [13]. However, it is not clear that how to optimally select the number m of stages and the number N_i of PWs in each stage. Gualdi *et al.* [13] give an empirical rule for parameter selection

$$N_i = N_1 \times e^{-\gamma(i-1)}, i = 1, \ldots, m \quad (7)$$

where N_1 is the initial number of PWs. The empirical values of m and γ are 5 and 0.44, respectively. The exponential rule of (7) makes the number N_i decrease from stage to stage. Representative values of N_i are given in Table I.

TABLE I
REPRESENTATIVE VALUES OF N_i

Stage number i	1	2	3	4	5
N_i	2000	1288	829	534	349

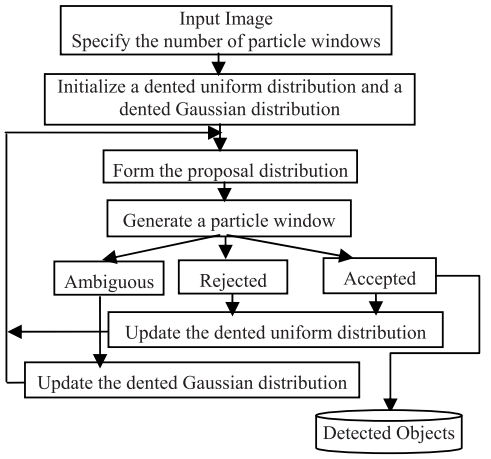


Fig. 2. Basic flowchart of the proposed iPW algorithm.

There are three problems with the MPW algorithm.

1) N_1 has to be large enough so that the N_1 PWs to some extent overlap the objects in the image. Otherwise, $N_2, \ldots, N_m$ PWs in later stages are hard to detect the objects. Extremely, if none of the N_1 PWs are positives, then the subsequent N_2 new PWs sampled from $q_1(\mathbf{w}) = \sum_{i=1}^{N_1} f(\mathbf{w}_i) G(\mathbf{w}_i, \Sigma)$ will not contain any clue of the location about the objects because $G(\mathbf{w}_i, \Sigma)$ is meaningless in this case.
2) MPW generates too many unnecessary PWs around the object and object-like regions. Considering that classification of these regions is more time-consuming than obvious nonobject regions in the algorithm of cascade AdaBoost, generating too many PWs around the object and object-like regions greatly limits its efficiency.
3) The rule of parameter selection is not guaranteed to be optimal, because there is no reason to support that the values in Table I are the best.

IV. Improved Particle Windows

In this section, we propose to improve MPW in order to detect the objects in an image with a smaller number of PWs.

Fig. 2 shows the basic flowchart of the proposed algorithm. The main difference between the proposed iPW and MPW is that iPW employs both accepted and rejected particles to update the dented uniform distribution and employs ambiguous particles to update the dented uniform distribution. The dented

distributions are more powerful than the common nondented distributions for excluding nonobjects and attracting attentions on objects. To clearly describe the iPW algorithm, we define several concepts: rejection PW, acceptance PW, and ambiguity PW. Then, the concepts of regions of rejection and acceptance are introduced. Finally, we will describe iPW algorithm based on these concepts and explain why iPW is superior to MPW.

A. Rejection, Acceptance, and Ambiguity Windows

As stated in Section III, each PW represents a window $\mathbf{w} = (x, y, s)^T$, where x, y, and s are the horizontal position, the vertical position, and the size of the window, respectively. The PW can also be expressed as $\mathbf{w}(x, y, s)$. In the following part, we use $\mathbf{w}(x, y)$ to represent $\mathbf{w}(x, y, s)$ when s is fixed.

We employ the classifier response $f(\mathbf{w})$ and its low and high thresholds (i.e., t_l and t_h) to define rejection, acceptance, and ambiguity PWs, respectively.

1) A window $\mathbf{w}$ is called rejection PW (RPW) if $f(w) < t_l$. Rejection PW is the PW which can be definitely classified as negative class due to its low classifier response.
2) A window $\mathbf{w}$ is called acceptance PW (APW) if $f(\mathbf{w}) \geq t_h$. Acceptance PW is the PW which can be safely classified as positive class (object) because of its high value of classifier response.
3) A window $\mathbf{w}$ is called ambiguity PW (ABPW) if $t_l \leq f(\mathbf{w}) < t_h$. One cannot classify ambiguity PW as positive class because its classifier response is not large enough. Meanwhile, one cannot classify it as negative class because its classifier response is not low enough. We call the set of the ambiguity PWs $\mathbf{W}_{AB}$.

The threshold t_h is just the threshold corresponding to the employed classifier. For a linear SVM classifier $f(\mathbf{x}) = \mathbf{w}'\mathbf{x}+\mathbf{b}$, the decision (classification) for a sample $\mathbf{x}$ is made according to whether or not $f(\mathbf{x})$ is larger than zero. Therefore, we employ zero as the threshold t_h (i.e., $t_h = 0$). Now, we explain how to select t_h for a cascade classifier. The classifier response $f(\mathbf{x})$ of a cascade AdaBoost is defined by $f(\mathbf{x}) = j_{\mathbf{x}}/L$, where $j_{\mathbf{x}}$ is the index j of the last stage which makes a positive classification for $\mathbf{x}$, and $L = 10$ is the total number of the stages of the cascade structure. Obviously, a sample $\mathbf{x}$ can be accepted as positive class only if it passes through all the L stages. In this case, $j_{\mathbf{x}} = L$ holds and the classifier response is $f_{\mathbf{x}} = j_{\mathbf{x}}/L = L/L = 1$. Consequently, t_h is set to 1. The threshold t_l is experimentally chosen which is to be described in Section V-A.

B. Regions of Rejection and Acceptance

The description of regions of rejection and acceptance is graphically supported by Fig. 3. Let the center of a window $\mathbf{w}$ coincide with the center of an object and the size (scale) of the windows match that of the object very well [Fig. 3(a)]. Then we compute the classifier responses of the neighboring windows $\mathbf{w}_i$. As shown in Fig. 3(b), it is usual that $f(\mathbf{w})$ is the largest and $f(\mathbf{w}_i)$ decreases with the distance $||\mathbf{w}_i - \mathbf{w}||$.

1) Region of Rejection: If a PW $\mathbf{w}$ is considered as a rejection PW, it can be used to securely reject a set of nearby windows. The centers of the nearby windows $\mathbf{w}_i$ and $\mathbf{w}$ itself are called region of rejection (RoR) of $\mathbf{w}$. We denote the RoR by $\mathbf{R}_R$

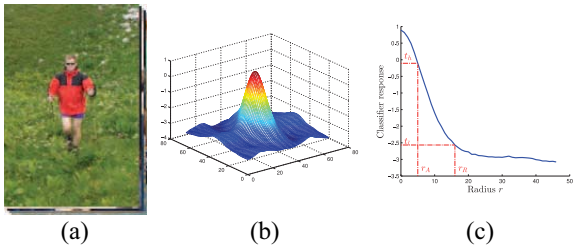


Fig. 3. (a) Original images. (b) Average classifier response $f(\mathbf{w}_i)$ of the windows. (c) Profile of (b).

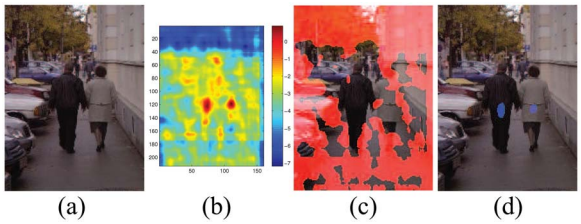


Fig. 4. Illustration of RoR and RoA. (a) Original image. (b) Classifier response. (c) Regions of rejection. (d) Regions of acceptance.

$$\mathbf{R}_R(\mathbf{w}) = \{x, y|\ ||\mathbf{w}_i - \mathbf{w}|| < r_R\} \tag{8}$$

where the radius r_R is the maximum radius

$$r_R = \max_{\mathbf{w}_i} ||\mathbf{w}_i - \mathbf{w}||, \quad \text{s.t. } f(\mathbf{w}_i) < t_l \tag{9}$$

r_R is a function of the classifier response $f(\mathbf{w})$.

All the windows belonging to $\mathbf{R}_R(\mathbf{w})$ are represented as $\mathbf{W}_R(\mathbf{w})$.

Assumption 1 (Rejection Assumption): If a window $\mathbf{w}$ is classified as a rejection window due to $f(\mathbf{w}) < t_l$ then all the windows $\mathbf{W}_R(\mathbf{w})$ can be directly rejected (i.e., labeled as negatives) without the necessity of computing the classifier response $f(\mathbf{w}_i)$, $\mathbf{w}_i \in \mathbf{W}_R(\mathbf{w})$.

This assumption is illustrated in Fig. 4(c), where the red part in the image consists of regions of rejection.

2) Region of Acceptance: If a PW $\mathbf{w}$ is considered as an acceptance PW, it can be used to securely accept a set of nearby windows. The centers of the nearby windows $\mathbf{w}_i$ and $\mathbf{w}$ itself form region of accept (RoA) of $\mathbf{w}$. Mathematically, RoA can be denoted by $\mathbf{R}_A$

$$\mathbf{R}_A(\mathbf{w}) = \{x, y|\ ||\mathbf{w}_i - \mathbf{w}|| < r_A\} \tag{10}$$

where the radius r_A is the maximum radius

$$r_A = \max_{\mathbf{w}_i} ||\mathbf{w}_i - \mathbf{w}||, \quad \text{s.t. } f(\mathbf{w}_i) \geq t_l. \tag{11}$$

Note that constraint is $f(\mathbf{w}_i) \geq t_l$ instead of $f(\mathbf{w}_i) \geq t_h$. A window $\mathbf{w}$ merely satisfying $t_h > f(\mathbf{w}) \geq t_l$ is called an ambiguity PW. But if it is close to an acceptance PW, then the existence of the acceptance PW is able to eliminate the ambiguity. Therefore, $f(\mathbf{w}_i) \geq t_l$ is used for defining RoA. Obviously, r_A with $f(\mathbf{w}_i) \geq t_l$ is larger than r_A with $f(\mathbf{w}_i) \geq t_h$.

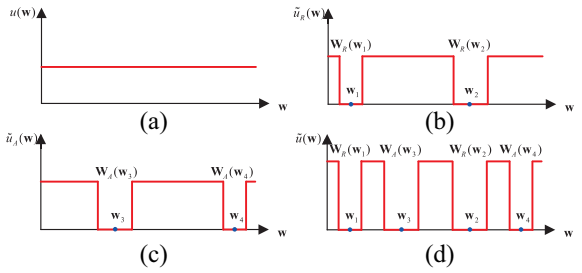


Fig. 5. Uniform distribution and dented uniform distribution. (a) Uniform distribution. (b) Dented uniform distribution $\tilde{u}_R(\mathbf{w})$ formed by two rejection PWs $\mathbf{w}_1$ and $\mathbf{w}_2$. (c) Dented uniform distribution $\tilde{u}_A(\mathbf{w})$ formed by two acceptance PWs $\mathbf{w}_3$ and $\mathbf{w}_4$. (d) Mixture dented uniform distribution by combing the rejection and acceptance PWs.

All the windows belonging to $\mathbf{R}_A(\mathbf{w})$ are represented as $\mathbf{W}_A(\mathbf{w})$.

Assumption 2 (Acceptance Assumption): If a window $\mathbf{w}$ is classified as acceptance window due to $f(\mathbf{w}) \geq t_h$ then all the windows $\mathbf{W}_A(\mathbf{w})$ should be directly accepted (i.e., labeled as positives) without the necessity of computing the classifier response $f(\mathbf{w}_i)$, $\mathbf{w}_i \in \mathbf{W}_A(\mathbf{w})$.

In Fig. 4(d), the blue part in the image consists of the regions of acceptance.

C. iPW: Rejection-Based Random Sampling

In this section, we describe iPW based on the concepts of RPW, APW, and ABPW, and their corresponding RoR and RoA.

1) Proposal Distribution of iPW: We introduce the motivation of iPW by first analyzing the properties and limitations of MPW.

a) Rejection particle windows and dented uniform distribution $\tilde{u}_R(\mathbf{w})$: As discussed in Section III-A, the PWs of MPW are sampled from $\sum f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. The contribution of a PWs $\mathbf{w}_i$ is determined by its weight $f(\mathbf{w}_i)$. If $f(\mathbf{w}_i)$ is very small, then the $\mathbf{w}_i$ contributes little to object detection because subsequent stage will not sample windows from the corresponding distribution $f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. However, if only a small number of PWs is generated, most of them will have small weights and the regions of objects will be not sampled. In this paper, we propose how to make use of these PWs with small weights (i.e., RPWs) for efficient object detection. One of the main contributions of the paper is to use rejection PWs $\mathbf{w}_i$ and the corresponding $\mathbf{W}_R(\mathbf{w}_i)$ [i.e., the set of windows insides the region $\mathbf{R}_R(\mathbf{w}_i)$] to form a dented uniform distribution $\tilde{u}_R(\mathbf{w})$. Let the number of RPWs be N_R, then $\tilde{u}_R(\mathbf{w})$ can be expressed as

$$\tilde{u}_R(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_R(\mathbf{w}_1) \cup \cdots \cup \mathbf{W}_R(\mathbf{w}_{N_R})\} \\ \dfrac{1}{a_R}, & \text{otherwise} \end{cases} \tag{12}$$

where a_R satisfies $\iiint (1/a_R)dxdyds = 1$. Fig. 5(b) illustrates an 1-D dented uniform distribution. Obviously, sampling from the dented uniform distribution is capable of avoiding drawing windows from the existing regions of rejection.

b) Acceptance particle window and dented uniform distribution $\tilde{u}_A(\mathbf{w})$: In MPW, when a PW $\mathbf{w}_i$ (i.e., acceptance PW) has large weight $f(\mathbf{w}_i)$, it has large contribution to the distribution $\sum f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. Sampling from this updated distribution will generate PWs overlapping or even coinciding with $\mathbf{w}_i$. We think that it is redundant to resample the acceptance window $\mathbf{w}_i$ and its close neighbors. To avoid the redundancy, we propose to employ the acceptance PWs to maintain and update a dented uniform distribution $\tilde{u}_A(\mathbf{w})$

$$\tilde{u}_A(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_A(\mathbf{w}_1) \cup \cdots \cup \mathbf{W}_A(\mathbf{w}_{N_A})\} \\ \dfrac{1}{a_A}, & \text{otherwise} \end{cases} \tag{13}$$

where a_A satisfies $\iiint (1/a_A)dxdyds = 1$, and N_A is the number of acceptance PWs. Fig. 5(c) illustrates a $\tilde{u}_A(\mathbf{w})$.

Obviously, both the rejection and acceptance PWs play role in excluding regions in uniform distribution. Therefore, as illustrated in Fig. 5(d), we propose to combine $\tilde{u}_R(\mathbf{w})$ and $\tilde{u}_A(\mathbf{w})$ into a unified dented uniform distribution $\tilde{u}(\mathbf{w}) = \tilde{u}_R(\mathbf{w}) \times \tilde{u}_A(\mathbf{w})$

$$\tilde{u}(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_R \cup \mathbf{W}_A\} \\ \dfrac{1}{a}, & \text{otherwise} \end{cases} \tag{14}$$

where a satisfies $\iiint (1/a)dxdyds = 1$.

c) Ambiguity particle windows and dented gaussian distribution: Suppose that there are N_{AB} ambiguity PWs in $\mathbf{W}_{AB}$. With the class label being either positive class or negative class, the region nearby the ambiguity PW exhibits the largest uncertainty relative to the rejection PWs and the acceptance PWs, so it contains potential clue for detecting objects. One way to exploit the N_{AB} ambiguity PWs is directly using them for modeling a mixture of Gaussian distribution $g(\mathbf{w})$

$$g(\mathbf{w}) = \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma). \tag{15}$$

However, our experimental results show that sampling from $g(\mathbf{w})$ may result in PWs overlapping with the existing rejection or acceptance PWs. Clearly, it is useless to sample such PWs. Therefore, to remove the redundancy, we propose to model a mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$ by using the N_{AB} ambiguity PWs with the help of the existing rejection and acceptance PWs or equivalently the current dented uniform distribution $\tilde{u}(\mathbf{w})$

$$\tilde{g}(\mathbf{w}) = \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i)\big[G(\mathbf{w}_i, \Sigma) \times (a \times \tilde{u}(\mathbf{w}))\big]. \tag{16}$$

Because $a \times \tilde{u}(\mathbf{w}) = 0$ in the regions of rejection and acceptance and $a \times \tilde{u}(\mathbf{w}) = 1$ elsewhere, so multiplying $G(\mathbf{w}_i, \Sigma)$ with $a \times \tilde{u}(\mathbf{w})$ results in a dented Gaussian distribution as is illustrated in Fig. 6. In Fig. 6(a) two ambiguity PWs $\mathbf{w}_1$ and $\mathbf{w}_2$ in $\mathbf{W}_{AB}$ form a mixture of Gaussian distribution $g(\mathbf{w})$. In Fig. 6(b), a rejection PW $\mathbf{w}_R$ and an acceptance PW $\mathbf{w}_A$ form a dented uniform distribution $\tilde{u}(\mathbf{w})$. Fig. 6(c) shows the final dented Gaussian distribution $\tilde{g}(\mathbf{w})$.

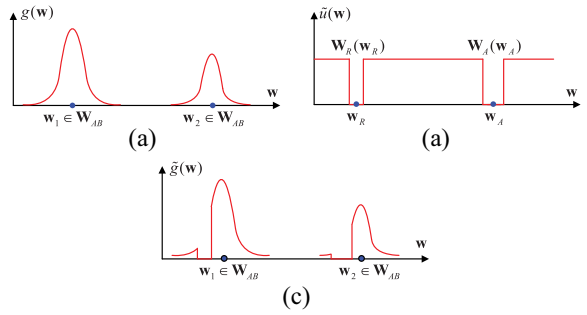


Fig. 6. Dented Gaussian distribution. (a) Gaussian distribution. (b) Dented uniform distribution. (c) Mixture of dented Gaussian distribution.

d) Proposal distribution: The PD $\tilde{q}(\mathbf{w})$ of iPW is a weighted average of the dented uniform distribution $\tilde{u}(\mathbf{w})$ and mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$

$$\begin{aligned}\tilde{q}(\mathbf{w}) &= P_u \times \tilde{u}(\mathbf{w}) + P_g \times \tilde{g}(\mathbf{w}) \\ &= P_u \times \tilde{u}(\mathbf{w}) \\ &\quad + P_g \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i)\big[G(\mathbf{w}_i, \Sigma) \times (a \times \tilde{u}(\mathbf{w}))\big]. \quad (17)\end{aligned}$$

More generally, the PD $\tilde{q}_i(\mathbf{w})$ in stage i is expressed as

$$\begin{aligned}\tilde{q}_i(\mathbf{w}) &= P_u(i) \times \tilde{u}_i(\mathbf{w}) \\ &\quad + P_g(i) \sum_{j=1}^{N_{AB}} f(\mathbf{w}_j)\big[G(\mathbf{w}_j, \Sigma) \times (a \times \tilde{u}_i(\mathbf{w}))\big]. \quad (18)\end{aligned}$$

The weights P_u and P_g can be regarded as the posterior probabilities for $\mathbf{w}_i$ to be generated from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$, respectively. That is $p(\tilde{u}(\mathbf{w})|\mathbf{w}_i) = P_u$ and $p(\tilde{g}(\mathbf{w})|\mathbf{w}_i) = P_g$ which can be respectively defined by

$$P_u = \alpha \times \left(1 - \frac{N_R + N_A}{N}\right), \text{ and } P_g = 1 - P_u. \quad (19)$$

In (19), $\alpha \in [0, 1]$ is used for performance adjusting, $N_R = |\mathbf{W}_R|$ and $N_A = |\mathbf{W}_A|$ are the numbers of rejection and acceptance PWs in $\mathbf{W}_R$ and $\mathbf{W}_A$, respectively.

e) Hypothesis 1: Suppose there are two stages and the number of PWs in stage 1 and stage 2 are N_1 and N_2, respectively. In both MPW and iPW, the N_1 PWs are generated from the same uniform distribution. But the N_2 PWs in stage 2 of MPW are sampled from $q(\mathbf{w})$ whereas they are sampled from $\tilde{q}(\mathbf{w})$ in iPW. Then the probability for N_2 PWs in iPW to contain the object is larger than that in MPW.

It is trivial to prove Hypothesis 1. If there are a nonzero number of rejection and/or acceptance PWs, then the search region is reduced by these PWs [equivalently, the dented uniform distribution $\tilde{u}_1(\mathbf{w})$]. Consequently, sampling the same number of PWs from reduced search domain is better than from the original large domain in the sense of detecting the objects. Generally, if both iPW and MPW use the same number of PWs in each stage, then the probability for iPW to detect the objects is larger than that of MPW.

f) Each stage consists of one particle window: So far, we have designed the new PD of iPW. The question is that how many PWs are to be generated in each stage. In MPW, the exponential rule of (7) is used for setting the window number. But this is far from optimal. Intuitively, we think that it is optimal if each stage contains one PW. However, it fails completely for MPW. Throughout the paper, iPW means the one where each stage has a single new PW. The number of generated PWs incrementally increases one by one. The first letter "i" of "iPW" is named after "incremental."

Algorithm 2 Basic Algorithm of iPW

Input:
The number N of all candidate windows;
The number N_{iPW} of total particle windows;
High and low classifier thresholds t_h and t_l, respectively;
Output:
The set $\mathbf{W}_P$ of positive particle windows.
Initialization
Empty the sets of rejection, acceptance, and ambiguity particle windows: $\mathbf{W}_R \leftarrow \Phi$, $\mathbf{W}_A \leftarrow \Phi$, and $\mathbf{W}_{AB} \leftarrow \Phi$, respectively. Let the numbers of rejection, acceptance, and ambiguity particle windows be $N_R = 0$, $N_A = 0$, and $N_{AB} = 0$.
Initialize the dented uniform and dented Gaussian distributions by $\tilde{u}(\mathbf{w}) \leftarrow 1/N$ and $\tilde{g}(\mathbf{w}) \leftarrow 0$.
Iteration:
for $i = 1$ **to** N_{iPW} **do**
$P_u = \alpha \times (1 - \frac{N_A + N_R}{N})$, $P_g = 1 - P_u$.
Sample a particle window $\mathbf{w}$ from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$. The probabilities for $\mathbf{w}$ to be generated from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ are P_u and P_g, respectively.
Put $\mathbf{w}$ into $\mathbf{W}_R$, $\mathbf{W}_A$, or $\mathbf{W}_{AB}$ according to the classifier response:
If $f(\mathbf{w}) < t_l$, then $\mathbf{W}_R = \mathbf{W}_R \cup \mathbf{W}_R(\mathbf{w})$, $N_R \leftarrow |\mathbf{W}_R|$;
If $f(\mathbf{w}) \geq t_h$, then $\mathbf{W}_A = \mathbf{W}_A \cup \mathbf{W}_A(\mathbf{w})$, $N_A \leftarrow |\mathbf{W}_A|$, and $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}$;
If $t_l \leq f(\mathbf{w}) < t_h$, then $\mathbf{W}_{AB} = \mathbf{W}_{AB} \cup \mathbf{w}$, $N_{AB} \leftarrow |\mathbf{W}_{AB}|$.
Update $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ using the updated $\mathbf{W}_R$, $\mathbf{W}_A$, and $\mathbf{W}_{AB}$.
end for
return $\mathbf{W}_P$.

2) Basic Algorithm of iPW: The core of iPW is iteratively sampling PWs from the PD $\tilde{q}(\mathbf{w})$ and updating the sets of rejection, acceptance, and ambiguity PWs. Because the PD $\tilde{q}(\mathbf{w})$ is a weighted average of the dented distributions $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$, drawing a PW from $\tilde{q}(\mathbf{w})$ is equivalent to drawing from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ with the probabilities P_u and P_g, respectively.

The basic algorithm of iPW is given in Algorithm 2. The output is $\mathbf{W}_P$ (the final set of positive PWs), where nonmaximum-suppression is applied for final object detection.

In the initialization step, the sets of rejection, acceptance, and ambiguity PWs are emptied (line 2). The dented uniform distribution $\tilde{u}(\mathbf{w})$ is initialized by the uniform distribution $u(\mathbf{w}) = 1/N$ because currently there are no rejection and acceptance PWs (line 3). The mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$ is initialized to be 0 because so far there are no ambiguity PWs to really construct it (line 3).

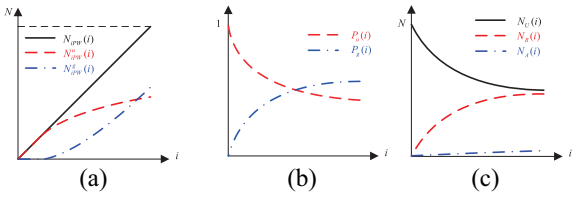


Fig. 7. Characteristics of iPW. Curves of (a) $N_{\text{iPW}}(i)$, $N^{g}_{\text{iPW}}(i)$, and $N^{u}_{\text{iPW}}(i)$, (b) $P_u(i)$ and $P_g(i)$, and (c) $N_U(i)$, $N_R(i)$, and $N_A(i)$.

In each iteration, a PW is generated from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ until the predefined number N_{iPW} of PWs are obtained. According to the value $f(\mathbf{w})$ of classifier response, the generated PW $\mathbf{w}$ is classified as rejection, acceptance, or ambiguity PW. If $\mathbf{w}$ is classified as rejection PW, then all the windows $\mathbf{W}_R(\mathbf{w})$ in the $\mathbf{R}_R(\mathbf{w})$ (RoR PW of $\mathbf{w}$) will be remerged into $\mathbf{W}_R$. If $\mathbf{w}$ is classified as acceptance PW, then all the windows $\mathbf{W}_A(\mathbf{w})$ in the $\mathbf{R}_A(\mathbf{w})$ (RoA PW of $\mathbf{w}$) will be remerged into $\mathbf{W}_A$. Otherwise it will be put into the window set $\mathbf{W}_{AB}$ (see line 9–11).

Finally, based on the updated $\mathbf{W}_R$, $\mathbf{W}_A$, and $\mathbf{W}_{AB}$, the dented distributions $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ are updated according to (14) and (16), respectively.

a) Characteristics of the iPW algorithm: Fig. 7 demonstrates the characteristics of the proposed iPW algorithm. Because each stage (iteration) generates a single PW, the cumulative number $N_{\text{iPW}}(i)$ of generated PWs at stage i is $N_{\text{iPW}}(i) = i$. Among the $N_{\text{iPW}}(i)$ PWs, $P_u(i)$ fraction is sampled from $\tilde{u}(\mathbf{w})$ whereas $P_g(i)$ fraction is sampled from $\tilde{g}(\mathbf{w})$. That is, the number of windows coming from $\tilde{u}(\mathbf{w})$ is $N^{u}_{\text{iPW}}(i) = P_u(i) \times N_{\text{iPW}}(i)$, and the number of windows coming from $\tilde{g}(\mathbf{w})$ is $N^{g}_{\text{iPW}}(i) \approx P_g(i) \times N_{\text{iPW}}(i)$. Fig. 7(a) shows that most of the generated PWs are from $\tilde{u}(\mathbf{w})$ in the first several stages. But its fraction [i.e., $P_u(i)$] decreases as iteration proceeds meanwhile the fraction $P_g(i)$ increases [see Fig. 7(b)].

The above phenomenon is explained as follows. Because the number of windows that contains objects is very small relative to the total number of windows in the image, sampling a PW from the initial distribution $u(\mathbf{w})$ and the distribution $\tilde{u}_i(\mathbf{w})$ in first few stages (e.g., $i = 10$) will result in rejection PWs in a very large probability. Consequently, the number $N_R(i)$ of rejection PWs increases very fast with i, but the number $N_A(i)$ of acceptance PWs increases very slowly [see Fig. 7(c)], which makes the number $N_u(i) = N - N_R(i) - N_A(i)$ of unvisited windows be very large for small i. According to (19), $P_u(i) \gg P_g(i)$. But as iteration proceeds, $N_R(i)$ and $N_A(i)$ increase monotonically making P_u decrease.

Because $N_R(i)$ and $N_A(i)$ increase monotonically with i, so it can pay more attention to the other unvisited potential areas. This characteristic makes iPW not to generate too many unnecessary PWs around the object and object-like regions. It is known that classification of these regions is very time-consuming if cascade AdaBoost is adopted. This is one of the advantages of iPW over MPW.

3) Semi-Incremental Version of iPW: The purely incremental Algorithm 2 has some problems. First, in the first few iterations the number of PWs is very small, so $N_{AB}(i)$ in it is very small. In this case, $\tilde{g}(\mathbf{w})$ cannot reflect the probability distribution of the whole image. In order to have enough ambiguity particles to represent the probability distribution, the variables N^*_C and b are introduced in Algorithm 3. Through them, the PWs are forcibly sampled from $\tilde{u}(\mathbf{w})$ in first several stages until a certain number N^*_C of PWs, especially the ambiguity PWs, is available. By this way, it can reflect the probability of the whole image very well. Second, if a PW is sampled from $\tilde{g}(\mathbf{w})$ and then $\tilde{g}(\mathbf{w})$ is immediately updated, the latter sampled PWs will be heavily centered at regions having strongest classifier responses. Namely, the regions with strongest responses will be enhanced more and more, but the regions with the relative low responses where object exists will be ignored. So instead of updating $\tilde{g}(\mathbf{w})$ per PW, it is wise to update $\tilde{g}(\mathbf{w})$ until there is a certain number of PWs (line 14). To overcome the above problems, a semi-incremental version of iPW (i.e., Algorithm 3) is proposed. The main differences from Algorithm 2 are written in italic. We call

Algorithm 3 Semi-Incremental Version of iPW

Input:

The number N of all candidate windows;
The number N_{iPW} of total particle windows;
High and low classifier thresholds t_h and t_l, respectively;
The threshold number N^*_C for the number of ambiguity particle windows

Output:

The set $\mathbf{W}_P$ of positive particle windows.

Initialization

Empty the sets of rejection, acceptance, and ambiguity particle windows: $\mathbf{W}_R \leftarrow \Phi$, $\mathbf{W}_A \leftarrow \Phi$, and $\mathbf{W}_{AB} \leftarrow \Phi$, respectively. Let the numbers of rejection, acceptance, and ambiguity particle windows be $N_R = 0$, $N_A = 0$, and $N_{AB} = 0$.
Initialize the dented uniform and dented Gaussian distributions by $\tilde{u}(\mathbf{w}) \leftarrow 1/N$ and $\tilde{g}(\mathbf{w}) \leftarrow 0$.
Initialize the binary indicator $b = 0$ and the number of cumulative particle windows $N_C = 0$.

Iteration:

for $i = 1$ **to** N_{iPW} **do**
If $b = 0$, then $P_u \leftarrow 1$ and $P_g \leftarrow 0$, else $P_u = a \times (1 - \frac{N_A + N_R}{N})$ and $P_g = 1 - P_u$.
Sample a particle window $\mathbf{w}$ from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ with P_u and P_g, respectively. $N_C = N_C + 1$.
Put $\mathbf{w}$ into $\mathbf{W}_R$, $\mathbf{W}_A$, or $\mathbf{W}_{AB}$ according to the classifier response:
If $f(\mathbf{w}) < t_l$, then $\mathbf{W}_R = \mathbf{W}_R \cup \mathbf{W}_R(\mathbf{w})$, $N_R \leftarrow |\mathbf{W}_R|$;
If $f(\mathbf{w}) \geq t_h$, then $\mathbf{W}_A = \mathbf{W}_A \cup \mathbf{W}_A(\mathbf{w})$, $N_A \leftarrow |\mathbf{W}_A|$, and $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}$;
If $t_l \leq f(\mathbf{w}) < t_h$, then $\mathbf{W}_{AB} = \mathbf{W}_{AB} \cup \mathbf{w}$, $N_{AB} \leftarrow |\mathbf{W}_{AB}|$.
Update $\tilde{u}(\mathbf{w})$ using the updated $\mathbf{W}_R$ and $\mathbf{W}_A$.
If $N_C = N^*_C$, then $b = 1$, update $\tilde{g}(\mathbf{w})$, $\mathbf{W}_{AB} \leftarrow \Phi$, $N^*_C = N^*_C \times e^{-\gamma}$, and $N_C = 0$.
end for
return $\mathbf{W}_P$.

Algorithm 4 Draw a PW $\mathbf{w}$ From $\tilde{u}(\mathbf{w})$ [or $\tilde{g}(\mathbf{w})$]

Input:

The sets of rejection and acceptance particle windows: $\mathbf{W}_R$ and $\mathbf{W}_A$, respectively;

The maximum iteration number $N_{\max}$;

Output:

a particle window $\mathbf{w}$.

for $n = 1$ **to** $N_{\max}$ **do**

Draw a window $\mathbf{w}_n$ from the uniform distribution $u(\mathbf{w})$ (or $g(\mathbf{w})$).

If $\mathbf{w} \notin \mathbf{W}_R$ and $\mathbf{w} \notin \mathbf{W}_A$, then $\mathbf{w} = \mathbf{w}_n$, and break.

end for

return $\mathbf{w}$.

it semi-incremental algorithm because $\tilde{g}(\mathbf{w})$ is updated once there is a certain number N_C^* of PWs, though each stage has one PW and the rejection regions are updated in a purely incremental manner.

4) Efficiently Sampling From Dented Uniform and Gaussian Distributions: As can be seen from (1), the computation time of an object detection algorithm is composed of the time of window generation, feature extraction, and classification. So it is important for iPW to efficiently generating PWs from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$.

To efficiently draw a PW from $\tilde{u}(\mathbf{w})$, we, in Algorithm 4, propose to iteratively draw a window from standard uniform distribution $u(\mathbf{w})$ until it does not belong to $\mathbf{W}_R$ or $\mathbf{W}_A$. Similarly, to efficiently draw a PW from $\tilde{g}(\mathbf{w})$, in Algorithm 4 we propose to iteratively draw a window from standard mixture of Gaussian distribution $g(\mathbf{w})$ until it does not coincide with the elements of $\mathbf{W}_R$ and $\mathbf{W}_A$. As one can design algorithm for checking $\mathbf{w} \in \mathbf{W}_R$ and $\mathbf{w} \in \mathbf{W}_A$ in an extremely efficient manner, the computation time of window generation in iPW is negligible. The maximum iterations number $N_{\max}$ is used for avoiding infinite loops.

V. Experimental Results

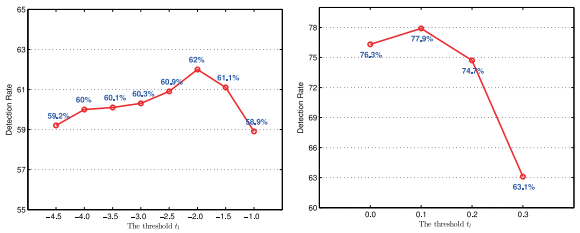
A. Experimental Setup

Experiments are carried out on the National de Recherche en Informatique et en Automatique (INRIA) pedestrian dataset and the MIT-CMU face dataset to compare the proposed iPW with MPW and SW. To detect pedestrians in INRIA dataset, HOG and SVM [11] are used for features and classifier, respectively. Haar-like features and cascade AdaBoost classifier are employed for detecting faces in the MIT-CMU dataset [38]. Note that the proposed algorithm is very general and hence can be used for other types of features and classifiers. The source code is publicly accessible at http://yanweipang.com/papers.

Intermediate results are also given to show the rationality of the assumptions mentioned above.

B. Results on the INRIA Pedestrian Database

In the INRIA dataset, the positive training set consists of 1208 normalized pedestrian windows, and the negative training set contains a mass of windows sampled from 1218 big and nonpedestrian images. The image size of the training window is 128×64 pixels, from which a 3780-dimensioned HOG feature vector is extracted. A linear SVM classifier $f(\mathbf{w})$ is obtained from the training sets.



(a) (b)

Fig. 8. Detection rate varies with t_l. (a) SVM classifier on the INRIA dataset. (b) AdaBoost classifier on the MIT-CMU dataset.

As can be seen from Algorithms 2 and 3, the explicit parameters of iPW are t_l, t_h, N_C^*, α, and γ. In our experiments, $t_l = -2.0$ and $t_h = 0$ are used. The optimal value of t_l is chosen according to Fig. 8(a) which shows how the detection rate varies with t_l when the FPPI $= 0.1$. The optimal t_l is the one with the largest detection rate. If t_l is too small, then the rejection ability of the algorithm becomes weak resulting in too large area of unvisited regions. It's difficult for the limited number of particles to efficiently detect the objects from the large unvisited regions. If t_l is too large, then the regions containing objects may be mistakenly rejected, because rejection and acceptance PWs are defined by not only t_l and t_h, but also r_R and r_A. So the parameters also include r_R and r_A. In (8) and (10), the regions of rejection and acceptance are circular and isotropic whose size are determined by r_R and r_A, respectively. However, because the height h of the pedestrian is larger than its width w, it is more reasonable that the RoR and acceptance is rectangle. The size of the rectangle is represented as $r_R^x \times r_R^y$. Likewise, the size of RoA is represented as $r_A^x \times r_A^y$. As stated in Section IV-B, r_R^x and r_R^y depend on the classifier response $f(\mathbf{w})$. In our experiments, r_R^x and r_R^y are quantized to nine intervals according to the value of $f(\mathbf{w})$. r_R^x and r_R^y also depend on the object width w and height h in question. Solid experiments are conducted to find rule for setting r_R^x and r_R^y according to $f(\mathbf{w})$, h, and w. Table II shows how to choose r_R^x and r_R^y. Note that $h = 128$ and $w = 64$. In Algorithm 3, we only use r_R^x and r_R^y when $f(\mathbf{w})$ belongs to the first four intervals. r_A^x/w and r_A^y/h are set to 0.16 and 0.16, respectively. $N_C^* = 0.5N_{\text{iPW}}$ is employed in the initialization step of Algorithm 3. The parameters α and γ are set 0.2 and 0.7, respectively.

It is noted that regions of rejection and acceptance are cubic when scale factor is considered. The testing image is zoomed out by a factor $1/1.05$. If a window is rejected at current scale s, which belongs to the interval N_{Interval}, then the windows in adjacent scales s' from $s \times 1.05^{3-N_{\text{Interval}}}$ to $s/1.05^{3-N_{\text{Interval}}}$ with the size $0.8^{\Delta} r_R^x \times 0.8^{\Delta} r_R^y$ ($\Delta = |\log_{1.05}^{s/s'}|$) are also rejected. If a window is accepted at current scale s, then the windows in adjacent scales s' from $s \times 1.05^3$ to $s/1.05^3$ with the size $0.8^{\Delta} r_A^x \times 0.8^{\Delta} r_A^y$ ($\Delta = |\log_{1.05}^{s/s'}|$) are also accepted.

TABLE II
SET r_R^x AND r_R^y ACCORDING TO $f(\mathbf{w})$, h, AND w

$N_{Interval}$	0	1	2	3	4	5	6	7	8
$f(\mathbf{w})$	[-inf, -4.0]	[-4.0, -3.5]	[-3.5, -3.0]	[-3.0, -2.5]	[-2.5, -2.0]	[-2.0, -1.5]	[-1.5, -1.0]	[-1.0, -0.5]	[-0.5, 0.0]
r_R^x/w	0.22	0.18	0.16	0.12	0.10	0.06	0.06	0.02	0.02
r_R^y/h	0.22	0.18	0.16	0.12	0.10	0.06	0.06	0.02	0.02

TABLE III
DETECTION RATES VARY WITH THE NUMBER N OF PWS WHEN FPPI $= 0.1$

N	2367	7100	11833	16567	21300	26033
MPW	0.469	0.594	0.614	0.623	0.625	0.627
iPW	0.561	0.620	0.625	0.627	0.628	0.630

Fig. 9. Detection results of iPW and MPW under 500 PWs. PWs of the last stage in (a) MPW and (c) iPW. Final detection result of (b) MPW and (d) iPW.

Table III compares iPW with MPW in terms of detection rate when they generate and examine the same number N of PWs. When $N = 2367$, the detection rate of iPW is 0.561 whereas the detection rate of MPW is 0.469, meaning that the detection rate of iPW is 9.2% higher than that of MPW. As N decreases, the advantage of iPW becomes more remarkable.

To further see the advantage of iPW, we show in Fig. 9 a specific detection result of iPW and MPW when N is as small as 500. The red dots in Fig. 9(a) indicate the centers of PWs in the last stage of MPW. Fig. 9(b) gives the final detection result by a nonmaximum suppression algorithm, where the man is detected whereas the woman is missing. The ambiguity PWs of the last stage of iPW are shown in Fig. 9(c) and the final detection result is shown in Fig. 9(d). On the one hand, Fig. 9 demonstrates that MPW fails to detect the woman when a small number of PWs is sampled whereas iPW is capable to localize both the woman and man. iPW generates the PWs one by one. If the generated PW has a lower classifier response, then the PW (i.e., rejection PW) will tell the next PW not to sample from it and its neighboring region (i.e., RoR). As a result, other regions including the object will have larger possibility to be investigated. By contrast, MPW does not have the rejection mechanism. When the current PWs do not contain clues of the objects, it is almost impossible for the latter PWs to capture the location information of the objects.

On the other hand, comparing Fig. 9(a) and (c), one can observe that MPW generates too many unnecessary PWs around the man, whereas iPW can properly assign the limited number of PWs to both the man region and the woman region. This phenomenon can be more clearly seen from

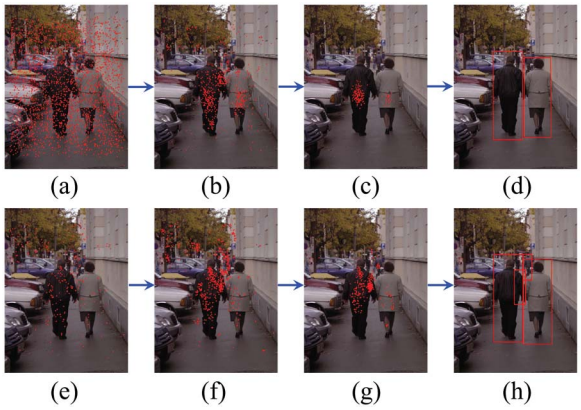


Fig. 10. MPW generates too many unnecessary PWs around the pedestrian regions. Updating process of PWs in (a)–(c) MPW and (e)–(g) iPW. Final detection result of (d) MPW and (h) iPW.

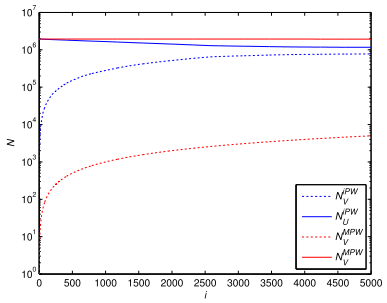


Fig. 11. Variations of N_v^{MPW}, N_u^{MPW}, N_v^{iPW}, and N_u^{iPW}.

Fig. 10. Fig. 10(a)–(c) gives the updating process of PWs in MPW, Fig. 10(d) shows that MPW is able to detect two persons if there is enough number of PWs in the initialization. Fig. 10(e)–(g) gives the updating process of PWs in iPW, Fig. 10(h) shows that iPW detect even four pedestrians, including a false positive.

Fig. 11 shows how $N_v^{\text{iPW}}(i) = |\mathbf{W}_R| + |\mathbf{W}_A|$ and $N_u^{\text{iPW}}(i) = N - |\mathbf{W}_R| - |\mathbf{W}_A|$ vary with the number i of generated PWs of iPW algorithm and how $N_v^{\text{MPW}}(i) = i$ and $N_u^{\text{MPW}}(i) = N - i$ vary with i of MPW algorithm. One can see that the number $N_v^{\text{iPW}}(i)$ of visited windows of iPW grows much faster than that of MPW. Equivalently, the number $N_u^{\text{iPW}}(i)$ of unvisited windows of iPW drops much faster than that of MPW. So generating the same number of PWs, iPW can classify (reject or accept) more windows (regions) than MPW. This explains why iPW obtains better detection accuracy than MPW when they use the same number of PWs.

If a smaller number of PWs is generated, can iPW achieve the same detection accuracy as MPW? If it is true, then one can conclude that iPW is more efficient than MPW. To answer

TABLE IV
EFFICIENCY OF SW, MPW, AND IPW ON INRIA

N_{SW}	N_{MPW}	N_{iPW}	N_{iPW}/N_{MPW}	T_{MPW}/T_{iPW}	T_{SW}/T_{MPW}	T_{SW}/T_{iPW}
47335	14200	7099	0.49	1.80	2.19	3.94

TABLE V
SET r_R ACCORDING TO $f(\mathbf{w})$ AND h

$f(\mathbf{w})$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_R/h	0.100	0.090	0.060	0.050	0.050	0.040	0.040	0.030	0.040	0.030

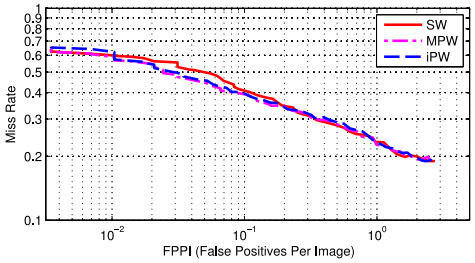


Fig. 12. DET curves comparing iPW with MPW and SW on INRIA.

this question, SW is used as a baseline. It scans the image with the pixel stride 8 and scaling factor 1.05, and the number of scanned windows is denoted by N_{SW}. N_{SW} varies with the size of testing image. For a 480×640 image, N_{SW} is 47 335. Let MPW generate $N_{\mathrm{MPW}} = 0.3 \times N_{\mathrm{SW}}$ PWs and iPW generate $N_{\mathrm{iPW}} = 0.15 \times N_{\mathrm{SW}}$ PWs. The resulting curves of miss rate versus false positive per image (FPPI) are plotted in Fig. 12. It is seen that these different window generation algorithms have very close operating points. For example, the miss rates of SW, MPW, and iPW are 23.4%, 22.8%, and 23.4%, respectively when FPPI $= 1$. At these operating points, the average values of N_{SW}, N_{MPW}, and N_{iPW} in INRIA are shown in Table IV. Table IV shows that to achieve the same operating point SW has to investigate 47 335 windows whereas it is enough for iPW to generate and check 7099 windows. The detection time T_{SW} of SW is 3.94 times of that (i.e., T_{iPW}) of iPW. Moreover, $N_{\mathrm{iPW}}/N_{\mathrm{MPW}} = 0.499$ means that using half of PWs iPW can obtain the same detection rate as MPW. The ratio of detection time T_{MPW} of MPW and detection time T_{iPW} of iPW is 1.8, implying much higher efficiency of iPW than MPW.

C. Results on the MIT-CMU Face Database

In Section V-B, the feature and classifier are HOG and SVM, respectively. In this section, we evaluate iPW by using Haar-like features and cascade AdaBoost classifier for detecting faces in the standard MIT-CMU face database. The testing set consists of 125 images with 483 frontal faces. A ten layers cascade model is learned from 20 000 normalized 20×20 small face images and 5000 nonface large negative images.

Because the range of response of cascade AdaBoost is quite different from SVM, the low and high classifier thresholds t_l and t_h are also different from those in Section V-B. The classifier response $f(\mathbf{w})$ of a cascade AdaBoost is defined by $f(\mathbf{w}) = j_{\mathbf{w}}/L$, where $j_{\mathbf{w}}$ is the index j of the last stage which provides a positive classification for $\mathbf{w}$, and $L = 10$ is total number of the stages of the cascade structure. The optimal value of t_l is chosen according to Fig. 8(b). Specifically, $t_l = 0.2$ and $t_h = 1.0$ are adopted. Consequently, the length r_R and r_A of regions of rejection and acceptance should be tuned. r_R and r_A are related to $f(\mathbf{w})$, h, and w. But the detection window is square, so $h = w$. The relationship between r_R, $f(\mathbf{w})$, and h is given in Table V, where r_R/h monotonously decreases with $f(\mathbf{w})$. In Algorithm 3, we only use the r_R when $f(\mathbf{w})$ belongs to the first two values. r_A^x/w and r_A^y/h are set to 0.1 and 0.1, respectively. $N_C^* = 0.5N_{\mathrm{iPW}}$ is employed in the initialization step of Algorithm 3. The parameters α and γ are set 0.2 and 0.7, respectively.

TABLE VI
DETECTION RATES VARY WITH THE NUMBER N OF PWS WHEN FPPI $= 0.1$

N	25066	75200	125333	175466	225600	275733
MPW	0.583	0.758	0.788	0.806	0.811	0.812
iPW	0.676	0.792	0.806	0.813	0.816	0.819

Similar to Section V-B, regions of rejection and acceptance are cubic. The testing image is zoomed out by a factor $1/1.15$. If a window is rejected at current scale s, the windows in neighboring scales s' from $s \times 1.15$ to $s/1.15$ with the size $0.5^{\Delta} r_R^x \times 0.5^{\Delta} r_R^y$ ($\Delta = |\log_{1.15}^{s/s'}|$) form the $\mathbf{W}_R$ of rejection PWs. If a window is accepted at current scale s, then the windows in adjacent scales s' from $s \times 1.15$ to $s/1.15$ with the size $0.5^{\Delta} r_A^x \times 0.5^{\Delta} r_A^y$ ($\Delta = |\log_{1.15}^{s/s'}|$) are also accepted.

With the above parameters, iPW is applied to 125 testing images and detection rates corresponding to different number N of PWs are shown in Table VI. Table VI also gives the detection rates of MPW. It is observed that iPW has almost higher detection rate in each case. When the number of PWs is small, the advantage of iPW is more remarkable. For example, when $N = 25\,066$, the detection rate of iPW is 9.3% higher than that of MPW.

Fig. 13(b) and (d) respectively shows the detection results of MPW and iPW when the number of PWs is limited to 5000. Clearly, iPW is capable of detecting the two faces in the testing image whereas MPW does not detect any face at all. Fig. 13(a) and (c) shows the centers of the PWs generated in the last stage of MPW and iPW, respectively.

Fig. 14(a)–(d) shows that when the number of PWs of MPW is upto 20 000, MPW can detect the two faces in the testing image. But MPW assigns too many PWs around object and object-like regions. The iterations of iPW are

TABLE VII
EFFICIENCY OF SW, MPW, AND IPW ON MIT-CMU

N_{SW}	N_{MPW}	N_{iPW}	N_{iPW}/N_{MPW}	T_{MPW}/T_{iPW}	T_{SW}/T_{MPW}	T_{SW}/T_{iPW}
501334	125333	65173	0.520	1.50	1.15	1.73

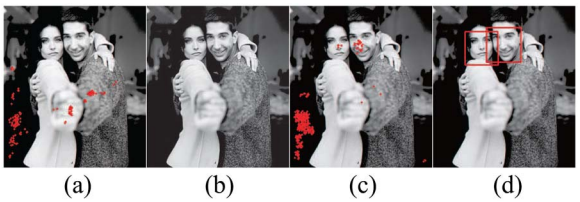


Fig. 13. Detection results of iPW and MPW under 5000 PWs. PWs of the last stage in (a) MPW and (c) iPW. Final detection result of (b) MPW and (d) iPW.

Fig. 14. MPW generates too many unnecessary PWs around the face regions. (a)–(c) Show the updating process of PWs. (e)–(g) Show the updating process of iPW. (d) and (h) are the final detection result of MPW and iPW, respectively.

shown in Fig. 14(e)–(h) which assign proper number of PWs around object regions. The intuition is that if there are a few acceptance PWs around the object, then it is no longer necessary to generate additional PWs around the object. Instead, the opportunity should be given to check other region.

Finally, experiments are conducted to see whether iPW can achieve comparable face detection results as MPW if a smaller number of PWs is used. As in pedestrian detection experiments, SW is also used as baseline. It slides the testing image with pixel stride 2 and scale factor 1.25. As a result, SW generates 621 816 and 3 941 097 windows for 454×628 and 1024×1280 images, respectively. Limit the numbers of PWs in MPW and iPW to $N_{\text{MPW}} = 0.25 \times N_{\text{SW}}$ and $N_{\text{iPW}} = 0.13 \times N_{\text{SW}}$, respectively. The resulting ROC curves of SW, MPW, and iPW are shown in Fig. 15. It is seen from Fig. 15 that SW is almost consistently inferior to both MPW and iPW. Even only 0.13 fraction of windows are used, iPW can obtain higher detection rates than SW. Moreover, one can see (Fig. 15) that the miss rates of iPW are almost identical to that of MPW.

Table VII shows the testing time T_{SW} of SW is 1.15 times and 1.73 times of MPW and iPW, respectively, when the number of generated windows of SW, MPW, and iPW are 501 334, 125 333, and 65 173. The miss rates of the three algorithms correspond to the point in Fig. 15 with FPPI = 1. The speedup effect in face detection is not significant as pedestrian detection. The reason is that the number of HOG features is fixed in pedestrian detection but the number of Haar-like features varies with the response of cascade classifier.

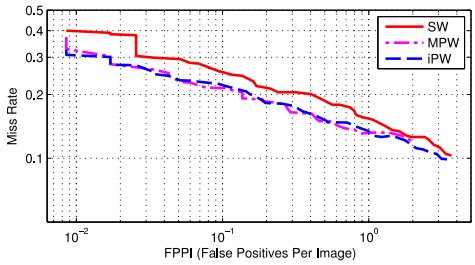


Fig. 15. DET curves comparing iPW with MPW and SW on MIT-CMU.

VI. CONCLUSION

In this paper, we have proposed a method to significantly improve MPW. The PD of MPW mainly relies on the regions of support. In contrast, the proposed iPW and semi-incremental version of iPW algorithms construct the PD based on the proposed concepts of rejection, acceptance, and ambiguity PWs which are defined by low and high thresholds of the classifier response. Both the rejection and acceptance PWs are used for reducing the search space. The existence of the objects is reflected by the acceptance PWs and the main clue of object locations is contained in ambiguity PWs. Specifically, the proposed PD is a weighted average of a dented uniform distribution and a dented Gaussian distribution which are dented by the rejection and acceptance PWs. An important characteristic of the proposed algorithms is that single PW is generated in each stage, which makes iPW to run in an incremental or semi-incremental manner. Experimental results have shown that iPW is about two times more efficient than MPW.

REFERENCES

[1] A. Andreopoulos and J. K. Tsotsos, "A computational learning theory of active object recognition under uncertainty," *Int. J. Comput. Vis.*, vol. 101, no. 1, pp. 95–142, 2013.
[2] B. Alexe, V. Petrescu, and V. Ferrari, "Exploiting spatial overlap to efficiently compute appearance distances between image windows," in *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, 2011, pp. 2735–2743.
[3] R. Benenson, M. Mathias, R. Timofte, and L.V. Gool, "Pedestrian detection at 100 frames per second," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, 2012, pp. 2903–2910.
[4] D. Benbouzid, R. Busa-Fekete, and B. Kégl, "Fast classification using sparse decision DAGs," in *Proc. Int. Conf. Mach. Learn.*, Edinburgh, U.K., 2012, pp. 2951–2955.
[5] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 613–627.
[6] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *Int. J. Comput. Vis.*, vol. 77, no. 1–3, pp. 65–86, 2008.

[7] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 3286–3293.
[8] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proc. Brit. Mach. Vis. Conf.*, London, U.K., 2009, pp. 91.1–91.11.
[9] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[10] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, 2012, pp. 645–659.
[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005.
[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 580–587.
[13] G. Gualdi, A. Prati, and R. Cucchiara, "Multistage particle windows for fast and accurate object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 8, pp. 1589–1604, Aug. 2012.
[14] S. Hinterstoisser *et al.*, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 876–888, May 2012.
[15] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?" in *Proc. Brit. Mach. Vis. Conf.*, Nottingham, U.K., 2014.
[16] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015. Doi: 10.1109/TPAMI.2015.2465908.
[17] S.-Y. Kung, "From green computing to big-data learning: A kernel learning perspective," in *Proc. IEEE Conf. Appl. Specific Syst. Archit. Processors*, Washington, DC, USA, 2013, p. 1.
[18] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2129–2142, Dec. 2009.
[19] A. D. Lehmann, P. V. Gehler, and L. Van Gool, "Branch&rank for efficient object detection," *Int. J. Comput. Vis.*, vol. 106, no. 3, pp. 252–268, 2013.
[20] A. Lehmann, B. Leibe, and L. Van Gool, "Fast PRISM: Branch and bound hough transform for object class detection," *Int. J. Comput. Vis.*, vol. 94, no. 2, pp. 175–197, 2011.
[21] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 259–289, 2008.
[22] Y.-F. Liu, J.-M. Guo, and C.-H. Chang, "Low resolution pedestrian detection using light robust features and hierarchical system," *Pattern Recognit.* vol. 47, no. 4, pp. 1616–1625, 2014.
[23] M.-W. Mak and S.-Y. Kung, "Low-power SVM classifiers for sound event classification on mobile devices," in *Proc. IEEE Conf. Acoust. Speech Signal Process.*, Kyoto, Japan, 2012, pp. 1985–1988.
[24] Y. Pang, K. Zhang, Y. Yuan, and K. Wang, "Distributed object detection with linear SVMs," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2122–2133, Nov. 2014.
[25] Y. Pang, Z. Song, J. Pan, and X. Li, "Truncation error analysis on reconstruction of signal from unsymmetrical local average sampling," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2100–2104, Oct. 2015.
[26] Y. Pang, Z. Ji, P. Jing, and X. Li, "Ranking graph embedding for learning to rerank," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 8, pp. 1292–1303, Aug. 2013.
[27] Y. Pang, S. Wang, and Y. Yuan, "Learning regularized LDA by clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2191–2201, Dec. 2014.
[28] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient HOG human detection," *Signal Process.*, vol. 91, no. 8, pp. 773–781, 2011.
[29] Y. Pang, H. Yan, Y. Yuan, and K. Wang, "Robust CoHOG feature extraction in human-centered image/video management system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 458–468, Apr. 2012.
[30] J. Pan, Y. Pang, K. Zhang, Y. Yuan, and K. Wang, "Energy-saving object detection by efficiently rejecting a set of neighboring sub-images," *Signal Process.*, vol. 93, no. 8, pp. 2205–2211, 2013.
[31] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.
[32] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1359–1371, Jul. 2014.
[33] C. Shen, P. Wang, S. Paisitkriangkrai, and A. van den Hengel, "Training effective node classifiers for cascade classification," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 326–347, 2013.
[34] R. Sznitman and B. Jedynak, "Active testing for face detection and localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1914–1920, Oct. 2010.
[35] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian detection aided by deep learning semantic tasks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 5079–5087.
[36] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.
[37] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, Mar. 2012.
[38] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[39] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proc. IEEE Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 32–39.
[40] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Fast and robust object detection using asymmetric totally corrective boosting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 33–46, Jan. 2012.
[41] W. Wang, J. Shen, X. Li, and F. Porikli, "Robust video object cosegmentation," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3137–3148, Oct. 2015.
[42] W. Wang, J. Shen, and F. Porikli, "Saliency-aware geodesic video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 3395–3402.
[43] P. Yan, Y. Cao, Y. Yuan, B. Turkbey, and P. L. Choyke, "Label image constrained multiatlas selection," *IEEE Trans. Cybern.*, vol. 45, no. 6, pp. 1158–1168, Jun. 2015.
[44] X. Yu, J. Yang, T. Wang, and T. Huang, "Key point detection by max pooling for tracking," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 444–452, Mar. 2015.
[45] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, 2006, pp. 1491–1498.
[46] W. Zhang, G. Zelinsky, and D. Samaras, "Real-time accurate object detection using multiple resolutions," in *Proc. Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, 2007, pp. 1–8.
[47] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 391–405.

Yanwei Pang (M'07–SM'09) received the Ph.D. degree in electronic engineering from the University of Science and Technology of China, Hefei, China, in 2004.

He is currently a Professor with the School of Electronic Information Engineering, Tianjin University, Tianjin, China. His current research interests include object detection and recognition and image processing, in which he has published over 90 scientific papers including 20 IEEE TRANSACTION papers.

Jiale Cao received the B.S. degree in electronic engineering from Tianjin University, Tianjin, China, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include object detection and image analysis.

Xuelong Li (M'05–SM'09–F'12) is a Full Professor with the Center for Optical Imagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.