Image and Vision Computing 30 (2012) 480–491

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

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

Boosted translation-tolerable classifiers for fast object detection ☆

Wei Zheng [a,b], Luhong Liang [a], Hong Chang [a,*], Cher-Keng Heng [c], Shiguang Shan [a], Xilin Chen [a]

[a] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[b] Graduate School of the Chinese Academy of Sciences, Beijing, 100039, China
[c] Panasonic R&D Center Singapore, Singapore

ARTICLE INFO

Article history:
Received 27 April 2011
Received in revised form 9 March 2012
Accepted 20 April 2012

Keywords:
Object detection
Boosting
Maximal Translation-Tolerable Region (MTTR)
Granularity-Adaptively-Tunable (GAT)

ABSTRACT

Different classifiers show different sensitivities to translation-variance. The translation-insensitive classifiers are capable of accelerating the detection process by searching over a coarse grid as well as guaranteeing the recall rate.

In this paper, we define a concept of Translation-Tolerable Region (TTR) for a classifier. The TTR is such a region that all the detection windows in it have consistent (stable) results output by the classifier. We use the classifier's Maximal Translation-Tolerable Region (MTTR) to measure its sensitivity to the translation-variance. For object detection, we propose an algorithm for training the discriminative classifiers as well as learning the associated MTTRs. The discriminative classifiers are assembled into a cascaded classifier in descending order of their MTTR sizes. To speed up the detection process, we propose a Granularity-Adaptively-Tunable (GAT) search strategy according to the classifiers' MTTRs. Furthermore, we prove that the recall rate is Probably Approximately Admissible (PAA) in the GAT search, which means that the proposed approach can theoretically guarantee the accuracy while accelerating the detection process.

Based on the boosting framework with Histograms of Oriented Gradients (HOG) features, we evaluate the proposed approach on the public datasets containing both rigid and non-rigid object classes. The experimental results show that our approach achieves considerable results with a fast speed.

1. Introduction

Object detection is not only an essential and challenging problem in computer vision, but also an indispensable technology in many emerging applications, e.g., video surveillance and driver assistance. The mainstream object detection approaches adopt a detection window (i.e., a normalized bounding box) to represent a target object and a binary classifier is trained to distinguish whether the detection window contains a target object or not. So far, the detection window based approaches are widely used for detecting many classes of objects, such as faces [1], cars [2] and humans [3–5]. In order to pinpoint the locations and scales of the target objects, this kind of approaches always utilizes the brute-force search strategy, i.e., searching over all possible scales and locations in the image.

Accuracy and speed are two main factors to be considered in designing an ideal object detection system. Aiming at accurate detection results, some researchers focus on designing powerful descriptors to represent objects, such as region descriptors [6–9], contour features [2,10,11], and heterogeneous features [12–14]. Besides image descriptors, some researchers turn to design discriminative classifiers to distinguish the object and non-object windows, such as multiple instance learning [15,16] and multiple kernel learning [17]. Besides the above holistic approaches, some researchers also propose to describe objects by part based models [18] to handle various situations (e.g., occlusion, deformation and illumination variance). Another important issue in object detection is speed, which is critical for real-time application. Many researchers have made great advances on this issue. For example, [19–22] search over a coarse grid or filter out the background regions to reduce the number of detection windows, [1,2,5,13] utilize a rejection-based cascaded classifier to efficiently reject most of the negative detection windows, [1,23] focus on how to efficiently calculate features and [24] reduces the computation complexity of the non-linear kernel classifiers, etc.

Most of the above approaches adopt a detection window to represent an object. To pinpoint the exact detection windows of objects, the detectors need to search in a three-dimensional space, i.e., horizontal locations, vertical locations and scales. The brute-force search needs to process a large number of detection windows. For example, the detector needs to evaluate 2,206,890 detection windows for a 640×480 image (given the detection window size of 80×40 and the scale stride of 1.05). Enlarging the search grid is a straightforward method to speed up the detection process by reducing the number of

☆ This paper has been recommended for acceptance by Jianmin Jiang.

* Corresponding author at: No. 6, Kexueyuan South Road, Haidian District, Beijing, 100190, China. Tel./fax: +86 10 62600523.
E-mail addresses: wei.zheng@vipl.ict.ac.cn (W. Zheng), luhong_liang@yahoo.com.cn (L. Liang), hong.chang@vipl.ict.ac.cn (H. Chang), CherKeng.Heng@sg.panasonic.com (C-K. Heng), shiguang.shan@vipl.ict.ac.cn (S. Shan), xilin.chen@vipl.ict.ac.cn (X. Chen).

0262-8856/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2012.04.003

the detection windows. However, it will cause accuracy loss [3,17]. It is a challenging problem to accelerate the search process and preserve the accuracy at the same time.

In this paper, we aim at accelerating the search procedure as well as preserving the accuracy. Our motivation is based on a simple observation that different classifiers show different sensitivities to the translation-variance, i.e., some classifiers are susceptible to translation-variance while others are not. In Fig. 1, we take the HOG features [5] as an example and give an intuitive explanation for classifiers' sensitivities to translation-variance. To obtain the five blocks in Fig. 1, we manually specify the locations and sizes for these blocks to capture the typical structures of side-view cars. Each block is fixed relative to the detection window and corresponds to a feature (HOG) vector. We collect 100 car samples and 300 randomly generated background samples to train a linear classifier for each block (see Section 3.2.1 for details). The linear classifier classifies detection windows as side-view cars when the associated responses are positive and vice versa. For each linear classifier, we translate the detection window surrounding the car and plot the responses of all the detection windows in the translation region. Since the blue block describes the overall structure of the car, the classifier response is insensitive to the translation-variance in both x-direction and y-direction in the translation region. The classifier based on the magenta block describes a corner structure of the car, thus it is sensitive to translation-variance in both x-direction and y-direction surrounding the car. The classifier based on the purple block describes the chassis structure, thus the classifier is insensitive to the translation-variance in x-direction. Although the magenta block and purple block are of the same size, they describe different structures of cars and thus show different sensitivities to translation-variance. The green block describes the roof structure while the red block describes the rear bumper structure. Comparing with the magenta one, they are more insensitive to the translation-variance in x-direction or y-direction. There are similar observations on other object classes, such as cows, bicycles, pedestrians and so on. The object detectors usually consist of many HOG classifiers with different sensitivities in x-direction and/or y-direction. These classifiers combine with and compensate for each other, thus the final classifier response may be translation-insensitive in both x-direction and y-direction. Intuitively, we can use the translation-insensitive classifiers to efficiently reject most of the negative windows by searching over a coarse grid and guarantee the recall rate at the same time.

Inspired by the above observation, we define a concept of Translation-Tolerable Region (TTR) in this paper. All the detection windows in the TTR should have consistent classification results, i.e., always being positive or negative. We use the classifier's Maximal Translation-Tolerable Region (MTTR) to measure its sensitivity to the translation-variance. Then, we propose an algorithm for training discriminative classifiers and learn the associated MTTRs at the same time. The discriminative classifiers are assembled into a cascaded classifier in descending order of their MTTR sizes. In the detection process, we propose a Granularity-Adaptively-Tunable (GAT) search that is a variant of the coarse-to-fine search. In the GAT search, the search grid of a classifier equals to its MTTR. We prove that the recall rate is Probably Approximately Admissible (PAA) in the GAT search, which means that the GAT search can guarantee the accuracy with a certain probability. To evaluate our approach, we implement a boosting framework based on the HOG features similar to [5]. We evaluate the proposed approach on PASCAL (VOC2006) dataset [25] and INRIA person dataset [6], containing both rigid objects and non-rigid objects. Comparing with brute-force search, extensive experimental results show that the proposed approach speeds up the search process by 10–90 times and guarantees the loss of the recall rate less than 1.4% at the same time.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 presents our methodology. Section 4 shows the experimental results. The last section draws the conclusions.

2. Related works

There are extensive literatures studying the speed issue of object detection. These literatures can be classified into two categories, namely reducing either the computational cost of one detection window or the number of the detection windows.

For the former category, highly efficient classification methods and fast features are proposed for reducing the computation cost of one detection window. The rejection-based classification strategy (e.g., the attentional classifier [1,5,26] and the coarse-to-fine classifiers [27,28]) is commonly used to speed up the detection process. Such classifiers efficiently reject most of the non-object

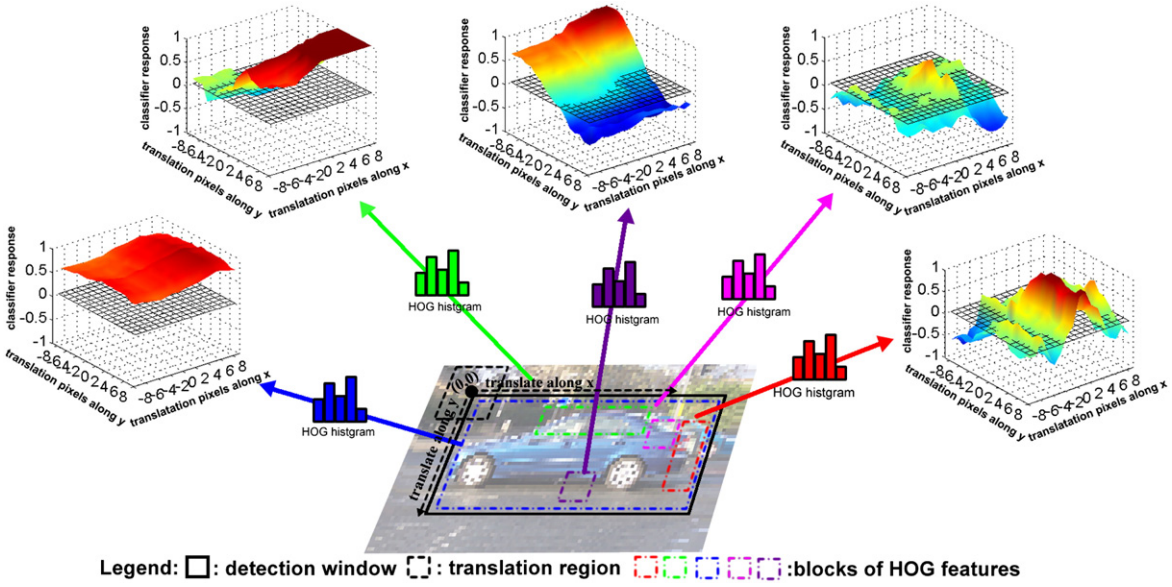

Fig. 1. Relationship between classifiers' responses based on HOG features and translation-variance. Different classifiers show different sensitivities to translation-variance.

detection windows with a cascaded classifier. Some approaches speed up the detection process by designing fast algorithms for feature extraction. The integral image approach is a typical example, which is proposed for many kinds of features, e.g., Haar-like features [1], HOG features [5], Covariance matrix [8] and image strip features [2]. Others speed up the feature extraction by approximating features using a part of features in the scale pyramid [29] or removing the computation redundancy between the overlapped detection windows [23]. Different from the above approaches, the proposed classifier reduces not only the computation cost of one detection window, but also the number of the detection windows at the same time.

The other category of approaches focuses on reducing the total number of detection windows. Most of them reduce the number of detection windows by searching over a coarse grid [3,8]. However, experiments show that this straightforward approach might miss object detection windows [3,17]. Recently, branch-and-bound search schemes [19,21] are proposed to fast localize the objects cooperating with the holistic SVM classifiers. Such approaches rely on a quality bound function that guarantees the globally optimal solution. Such quality bound functions are capable to bound the classifier response over a set of detection windows, thus it can fast reject a set of detection windows at one time. However, the quality bound functions are usually difficult to design. Some researchers are inspired by the mechanics of human visual attention, and they propose to fast search objects in different resolutions [20,30]. This kind of works uses the same search configuration for different object classes without considering the class specific structures and their impacts on the classifiers. For the proposed approach, the search configuration is adaptively tuned by the classifiers' MTTRs that are learned from the training data rather than manually designed. Therefore, the search configuration is adaptive for the object class, which is considered as a major difference between the proposed approach and existing approaches.

3. Proposed method

3.1. Translation-tolerable regions

A classifier's output is influenced by many factors, such as variance in translation, scales, and illumination. Specifically, this paper accelerates the detection process by studying the classifier's sensitivity to translation-variance.

To measure the classifier's sensitivity to translation-variance, we define a Translation Tolerable Region (TTR) as a local region in which the associated classifier outputs consistent classification results. In other words, all the detection windows in a TTR should have stable classification results (i.e., always being positive or negative). The output of a classifier is represented as $h(\boldsymbol{\chi})$, where $h(\bullet)$ is the classifier and $\boldsymbol{\chi}$ represents the corresponding detection window. Since the detection windows are fixed in size for a specific object class, we can represent a detection window using its left-top point $\chi=(x,y)$ and omitting its width and height. All the detection windows in a specific scale form a *grid map*, on which each point corresponds to the left-top point of a detection window. Then, we define a rectangular region $\boldsymbol{\tau}(\boldsymbol{\chi},\Delta x,\Delta y)$ on the grid map as

$$\tau(\boldsymbol{\chi},\Delta x,\Delta y)=\tau(x,y,\Delta x,\Delta y)=\{(u,v)||u-x|<\Delta x/2,|v-y|<\Delta y/2\},(1)$$

where $(\Delta x,\Delta y)$ decides the size of the region. $\boldsymbol{\tau}(\boldsymbol{\chi},\Delta x,\Delta y)$ is called TTR for the classifier $h(\bullet)$ if it satisfies

$$\forall\xi\in\tau(\chi,\Delta x,\Delta y),\quad h(\xi)h(\chi)>0, \tag{2}$$

where $\boldsymbol{\xi}$ represents a point in the region $\boldsymbol{\tau}(\boldsymbol{\chi},\Delta x,\Delta y)$ on the grid map. According to Eq. (2), the classifier $h(\bullet)$ is translation-tolerable in the region $\boldsymbol{\tau}(\boldsymbol{\chi},\Delta x,\Delta y)$ if it outputs consistent (stable) classification results in this region. Among all the TTRs surrounding $\boldsymbol{\chi}$, the maximal TTR (MTTR) satisfies the following two conditions

$$\begin{aligned}&\forall\xi\in\tau(\chi,\Delta x,\Delta y),\ h(\xi)h(\chi)>0,\\&\forall(\Delta x'>\Delta x\vee\Delta y'>\Delta y),\exists\xi'\in\tau(\chi,\Delta x',\Delta y'),\ h(\xi')h(\chi)<0,\end{aligned} \tag{3}$$

where $\Delta x'$ ($\Delta y'$) is any scalar that is larger than Δx (Δy) and $\boldsymbol{\xi}'$ is a point in the region $\boldsymbol{\tau}(\boldsymbol{\chi},\Delta x',\Delta y')$. According to Eq. (3), the classifier $h(\bullet)$ outputs consistent classification results surrounding $\boldsymbol{\chi}$ when it translates within the MTTR. Intuitively, the classifiers with large MTTRs are insensitive to translation-variance and vice versa. Hereby, we adopt the MTTR to measure the associated classifier's sensitivity to translation-variance.

For the detection problem, the positive samples (objects) have some typical translation-insensitive structures, e.g., the chassis structure in Fig. 1. Classifiers that describe such structures may be translation-insensitive surrounding the object location. Different from positive samples, the negative samples could be any patterns except the positive one. Therefore, it is difficult to find such a classifier that is translation-insensitive surrounding any non-object window. For simplicity, we only discuss the classifier's MTTRs surrounding the object windows in this paper. Then, Eq. (3) can be rewritten as

$$\begin{aligned}&\forall\xi\in\tau(\chi^+,\Delta x,\Delta y),\ h(\xi)>0,\\&\forall(\Delta x'>\Delta x\vee\Delta y'>\Delta y),\exists\xi'\in\tau(\chi^+,\Delta x',\Delta y'),\ h(\xi')<0,\end{aligned} \tag{4}$$

where $\boldsymbol{\chi}^+$ denotes the location of an object window. The object window can be aligned to the origin of the image, i.e., $\boldsymbol{\chi}^+=(0,0)$. Thus, $\boldsymbol{\tau}(\bullet)$ can be simplified as $\boldsymbol{\tau}(\Delta x,\Delta y)$ by omitting the location parameter $\boldsymbol{\chi}^+$.

Taking Fig. 1 as an example again, we can find the linear classifiers' TTRs and MTTRs according their definitions. Suppose that the object window is located at $\boldsymbol{\chi}^+=(0,0)$, the classifier based on the blue HOG block, $h_b(\bullet)$, always outputs positive in the region $\boldsymbol{\tau}(16,16)$. Therefore, $\boldsymbol{\tau}(16,16)$ is a TTR for $h_b(\bullet)$. Naturally, $\boldsymbol{\tau}(8,8)$, $\boldsymbol{\tau}(4,4)$ and all the other sub regions within $\boldsymbol{\tau}(16,16)$ are all TTRs for $h_b(\bullet)$. Likewise, the region $\boldsymbol{\tau}(2,4)$ is not only the TTR but also the MTTR of the classifier based on the magenta HOG block.

3.2. Learning cascaded classifier

The boosted cascaded classifiers have been successfully applied for object detection [1,2,5]. First, the boosting algorithm can select the proper weak features out of a large feature pool and assemble these weak features into a strong classifier. Second, the cascaded structure can efficiently reject most of the negative detection windows that is critical for fast object detection. Therefore, we adopt a boosted cascaded classifier that consists of several stages of strong classifiers as shown in Fig. 2.[1] Each strong classifier consists of several weak classifiers. The popular boosting algorithms, such as AdaBoost [1] or RealBoost [31], can be used for learning and selecting the weak classifiers based on the features. The final cascaded classifier should satisfy two conditions. First, the earlier strong classifiers' MTTRs should be large, thus these classifiers can search over a coarse grid without missing candidate objects. Second, the latter strong classifiers' MTTRs should be small, thus these classifiers can pinpoint the precise location of the object window. Intuitively, the above two conditions can be satisfied if the strong classifiers' MTTRs in the

[1] To verify our considerations, we perform an additional experiment on VOC2006 car dataset to compare the boosted cascaded classifier and a linear classifier using all the features in the feature pool. We train the linear classifier by minimizing the least square loss criterion (see Eq. (6)), the same criterion for training the weak classifiers in the boosted cascaded classifier. The average precision rates are 52.4% and 20.3% for the boosted cascaded classifier and the linear classifier respectively. Furthermore, the boosted cascaded classifier is about 500 times faster than the linear classifier.

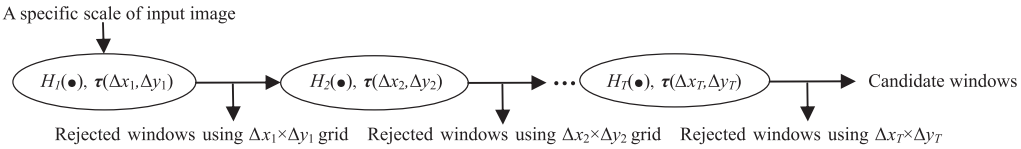

Fig. 2. Proposed cascaded classifier consists of many strong classifiers. Strong classifiers' MTTRs are learned from training data. Search grids equal to strong classifiers' MTTRs.

cascaded classifier are arranged in descending order of their sizes. Therefore, we adopt such strategy for the proposed classifier.

Supposing $H_t(\bullet)$ represents the t^{th} strong classifier, we represent the cascaded classifier in Fig. 2 as

$$\left\{H_t(), \tau(\Delta x_t, \Delta y_t)\right\}_{t=1}^{T}, \quad s.t.\ \Delta x_t \geq \Delta x_{t+1}, \Delta y_t \geq \Delta y_{t+1}, \tag{5}$$

where T is the total number of the strong classifiers and $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$ is the MTTR of the strong classifier $H_t(\bullet)$. In order to obtain a cascaded classifier satisfying Eq. (5), we need to solve the following two problems: (i) How to learn the strong classifier $H_t(\bullet)$ with a specified MTTR; (ii) How to guarantee the convergence of the training process.

3.2.1. Learning the strong classifier $H_t(\bullet)$

We adopt a baseline framework based on the boosting algorithm and the HOG features similar to [5]. The boosting algorithm learns a weak classifier for each HOG block, then selects the best weak classifiers, finally adds the weak classifiers into the strong classifiers. These strong classifiers are sequentially assembled into the cascaded classifier, i.e., object detector.

First, we learn linear weak classifiers for HOG representations. Given the training set $S=\{\mathbf{I}_i,l_i\}, i=1,\ldots,L$ and feature pool $\mathbf{F}=\{f_j(\bullet)\}$, $j=1,\ldots,N$, where $\mathbf{I}_i$ and $l_i\in\{+1,-1\}$ respectively represent the training image and the label for the i^{th} training sample, and $f_j(\bullet)$ represents the j^{th} HOG block. To obtain the feature pool, we uniformly sample the HOG blocks in the detection windows (see details in Section 4) and list the total number of the HOG blocks N in Table 1. A training sample can be represented by the triplet $(\mathbf{I}_i,u_i,v_i)$, where (u_i,v_i) represents the location of the detection window on the grid map. We use $f_j(\mathbf{I}_i,u_i,v_i)$ to represent the j^{th} HOG vector of the i^{th} training sample. For each HOG block, we train a linear weak classifier $\omega f_j(\mathbf{I}_i, u_i, v_i)+b$ to map the feature vector into a scalar. Many existing discriminative algorithms can be used for training the linear classifiers, such as Least Square Regression (LSR) [2] and linear SVM [5]. We adopt LSR and derive the parameters $(\boldsymbol{\omega},b)$ of the linear classifier by minimizing the square error

$$\min_{\omega,b}\left(\omega f_j(\mathbf{I}_i,u_i,v_i)+b-l_i\right)^2. \tag{6}$$

Then, with the feature responses from the learned linear classifiers, we adopt the RealBoost algorithm to learn the strong classifiers. The t^{th} strong classifier is represented as an additive classifier

$$H_t(\mathbf{I},u,v)=\sum_{m=1}^{M_t} h_{t,m}\left(f_{t,m}(\mathbf{I},u,v)\right), \tag{7}$$

where $h_{t,m}(f_{t,m}(\mathbf{I},u,v))$ is the m^{th} weak classifier of the t^{th} strong classifier, $f_{t,m}(\mathbf{I},u,v)$ is the HOG vector of the detection window (u,v) on the image $\mathbf{I}$, M_t is the total number of weak classifiers of the t^{th} strong classifier.

As discussed in Section 3.1, we suppose that $\mathbf{I}$ contains only one object, which locates at the origin of $\mathbf{I}$. If $\mathbf{I}$ contains more than one object, we consider each object separately by removing the other objects. Thus, the MTTR can be represented by $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$ by omitting the location parameter. To guarantee that the MTTR equals to $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$, $H_t(\bullet)$ should satisfy the two conditions in Eq. (4). First, all the detection windows within $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$ should be treated as positive samples

$$\forall(u,v)\in\boldsymbol{\tau}(\Delta x_t,\Delta y_t),\ H_t(\mathbf{I},u,v)>0. \tag{8}$$

In the boosting framework, Eq. (8) can be guaranteed with a certain probability

$$P(H_t(\mathbf{I},u,v)>0|(u,v)\in\boldsymbol{\tau}(\Delta x_t,\Delta y_t)))=P_t, \tag{9}$$

where P_t is the positive pass rate for the t^{th} strong classifier. Second, there is at least one negative detection window out of $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$. To satisfy this condition, all the detection windows out of $\boldsymbol{\tau}(\Delta x_t,\Delta y_t)$ are treated as negative samples

$$\forall(u,v)\notin\boldsymbol{\tau}(\Delta x_t,\Delta y_t),\ H_t(\mathbf{I},u,v)<0. \tag{10}$$

Eq. (10) can be guaranteed wtih a certain probability using

$$P(H_t(\mathbf{I},u,v)<0|(u,v)\notin\boldsymbol{\tau}(\Delta x_t,\Delta y_t))=N_t, \tag{11}$$

where N_t is the rejection rate of negative samples for the t^{th} strong classifier. For training the t^{th} strong classifier, the boosting algorithm iteratively adds new weak classifiers into the strong classifier until the rejection rate of the negative samples reaches N_t. Simultaneously, the pass rate of the positive samples is always guaranteed with a probability of P_t. In our experiments, we set P_t as 0.998 and N_t as 0.5.

In practice, we do not evaluate all the features for selecting the best feature (weak classifier). For efficiency consideration, we only sample 5% features from the feature pool to evaluate and select the best feature from the sampled features in each boosting round. Suppose that the feature pool consists of 5000 features, we randomly sample 250 features to evaluate. It can be guaranteed that at least one of the top 2% features will be selected with a probability of 99.4% $(=1-(1-0.02)^{250})$.

3.2.2. Attenuating MTTRs

The sizes of the MTTRs should be attenuated during the training process for two reasons. First, the negative patterns become more and more difficult to be distinguished from the positive pattern after several steps of bootstrapping. Consequently, it is difficult to classify the hard negative and positive samples, and the training algorithm may not converge if we do not attenuate the strong classifiers' MTTRs. Second, we let the search grids equal to the sizes of MTTRs (as will be discussed in Section 3.3). In order to pinpoint the precise location of the object window, we need to attenuate the MTTRs in the training process.

In this paper, we adaptively attenuate the MTTRs according to Z value that measures the discriminative power of the weak classifier in RealBoost algorithm [31]:

$$Z=\sum_j\sqrt{W_j^+W_j^-}, \tag{12}$$

where $W_j^+(W_j^-)$ represents the j^{th} partition of the probability distribution of the feature response for positive (negative) samples. The smaller Z is, the more discriminative the weak classifier is. Z value is also known as Bhattacharyya distance that measures the distance between two distributions. Since the negative samples become more

[2] Our additional experiments show that utilizing LSR and utilizing linear SVM achieve similar performance. However, training LSR is much faster than training SVM for the same data. Thus, we adopt LSR in this paper.

Table 1
Detection window sizes, total number of features in feature pool and number of positive training samples.

	Car	Bicycle	Bus	Motorbike	Person	Cow	Horse	Sheep	Cat	Dog	INRIA
Size	80×40	96×60	80×80	96×60	48×108	80×48	80×48	80×48	80×80	80×80	48×108
Feature	2520	6208	7568	6208	4960	3344	3344	3344	7568	7568	4960
Samples	486	199	169	181	492	188	202	314	205	200	1239

and more difficult during the training process, Z becomes more and more close to 1. If Z is very close to 1, the training process may not converge. One possible reason is that the size of the MTTR is set so large that the extended positive training set contains too much translation-variance. Similar to [11], we can make the training process converge by attenuating the MTTRs when Z is close to 1.

To guarantee the convergence of the training progress, we propose a threshold θ^Z and adaptively attenuate the MTTRs if Z is larger than θ^Z. A small θ^Z makes the algorithm attenuate MTTRs to $\boldsymbol{\tau}(1,1)$ quickly. As a result, the detection speed will become slow since many detection windows cannot be efficiently rejected with a large grid search. On the contrary, a large θ^Z makes the training algorithm attenuate the MTTRs slowly. The convergence of the training process will become very slow if the MTTRs are not attenuated on time. In this situation, the strong classifiers may contain an increasing number of features. Consequently, the detection speed may become slow due to the increasing number of features in the strong classifiers. However, it is difficult to find a criterion to learn an optimal θ^Z. To specify a proper value for θ^Z, we train four classifiers using VOC2006 car dataset (see Section 4 for details) according to the algorithm in Fig. 3. We set θ^Z as 0.98, 0.99, 0.995 and 1 respectively. When θ^Z equals to 1, the training algorithm will not attenuate the strong classifiers' MTTRs. We show the convergence of the false positive rate and evolution of the Z value in the training process in Fig. 4. As shown in Fig. 4(a), the larger θ^Z is, the slower the training process convergence is. If we do not attenuate the strong classifiers' MTTRs (i.e., $\theta^Z = 1$), the convergence of the false positive rate becomes slow after the feature number reaches 50. From Fig. 4(b), we can see

Algorithm: Training cascaded classifier.

INPUT: Feature pool: $\mathbf{F}$, positive training image set and annotations: $\mathbf{I}^+$, negative training image set: $\mathbf{I}^-$, threshold for attenuating MTTR: θ^Z, initial MTTR: $\boldsymbol{\tau}(\Delta x_0, \Delta y_0)$

Output: Cascaded classifier $\{H_t(\bullet), \boldsymbol{\tau}(\Delta x_t, \Delta y_t)\}_{t=1,2,\ldots,T}$

for t = 1:*MAX_STRONG_CLS* **do**
  Uniformly sample positive instances in MTTR $\boldsymbol{\tau}(\Delta x_t, \Delta y_t)$ in $\mathbf{I}^+$;
  Bootstrap hard negative instances using current cascaded classifier in $\mathbf{I}^+$ and $\mathbf{I}^-$ according to Eqn. (11);
  Estimate false positive rate ρ_t;
  for m=1:*MAX_WEAK_CLS* **do**
    Randomly sample L features in $\mathbf{F}$ and learn a linear classifier according to Eqn. (6) for each feature;
    Learn L weak classifiers using RealBoost based on outputs of linear weak classifiers;
    Choose best weak classifier $h_{t,m}(\bullet)$ according to Z in Eqn. (12) and add into strong classifier $H_t(\bullet)$;
    Select thresholds for $H_t(\bullet)$ to guarantee positive pass rate P_t in Eqn. (9);
    if Rejection rate of $H_t(\bullet)$ is larger than N_t in Eqn. (11) **then**
      Break;
    end if
  end for
  if $Z_t > \theta^Z$ **then**
    Attenuating MTTR: Δx_{t+1}=max(Δx_t/2,1), Δy_{t+1}= max(Δy_t/2,1);
  else
    $\Delta x_{t+1}=\Delta x_t$, $\Delta y_{t+1}=\Delta y_t$;
  end if
  if False positive rate of cascaded classifier $\rho_t < \rho^Z$ **then**
    Δx_{t+1}= 1, Δy_{t+1}= 1;
  end if
  if False positive rate of cascaded classifier $\rho_t < \rho^B$ **then**
    Break;
  end if
end for

Fig. 3. Training proposed cascaded classifier.

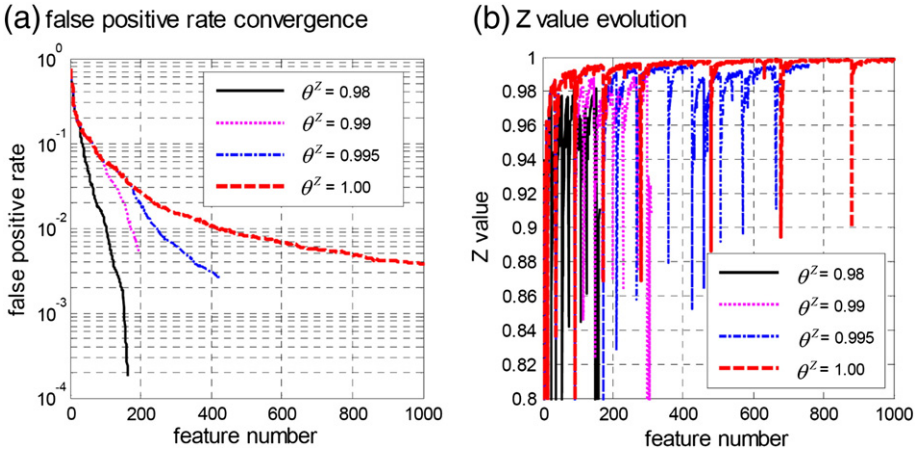

Fig. 4. Convergence of false positive rate and evolution of Z value.

that the Z value is in (0.98, 0.99) when the feature number is 50. Thus, the algorithm should attenuate the MTTRs to accelerate the convergence of the training process when Z value is around 0.98 or 0.99. The convergence of false positive rate becomes much faster after we attenuate the MTTRs. For other object classes, the convergence of training algorithm also becomes slow when Z is around 0.98 or 0.99. Similar conclusion is drawn in [11]. In our experiments, we set θ^Z as 0.98 for rigid objects and 0.99 for non-rigid objects in our experiments. We set a larger θ^Z for non-rigid objects, since the detectors may need more features to classify the non-rigid objects.

We give the training algorithm in Fig. 3 and discuss the parameters in the following. We set the initial MTTR as $\boldsymbol{\tau}(16,16)$. The false positive rate ρ^t will converge to 0 during the training process, where ρ^t is the ratio of the number of misclassified negative samples to the total number of negative samples. We stop the training process when the false positive rate is smaller than ρ^B. In the experiments, we set ρ^B as 10^{-6}. According to the GAT search (see Section 3.3), the MTTR of the last strong classifier should be $\boldsymbol{\tau}(1,1)$ if the object detector can densely scan the objects with 1×1 search grid. Therefore, we propose the threshold ρ^Z in case that the MTTR of the last strong classifier is larger than $\boldsymbol{\tau}(1,1)$. If the false positive rate ρ^t is smaller than ρ^Z, we attenuate the MTTR to $\boldsymbol{\tau}(1,1)$ directly. It is worth noting that ρ^Z should be larger than ρ^B, or else ρ^Z can be ignored. If ρ^Z is too large, the detection speed will become slow as the algorithm will attenuate the MTTR to $\boldsymbol{\tau}(1,1)$ too early. Based on these considerations, we set ρ^Z as 10^{-5}. ρ^Z is useful only when the MTTR is larger than $\boldsymbol{\tau}(1,1)$ and the false positive rate is smaller than ρ^Z. In our experiments, ρ^Z is useless since all the MTTRs are attenuated to $\boldsymbol{\tau}(1,1)$ before ρ^t reaches 10^{-5}. In case the training process does not converge, we set *MAX_STRONG_CLS* as 40 and *MAX_WEAK_CLS* as 200 to constrain the maximum number of strong classifiers and weak classifiers respectively.

3.3. GAT Search

In this section, we propose the GAT search using the cascaded classifier in Fig. 2. Moreover, we show that the recall rate is PAA in the GAT search.

The proposed cascaded classifier is a superior case of the popular cascaded classifier [1], since the strong classifiers' MTTRs learned in the training process can speed up the search process further. The remaining problem is how to design a search strategy according to the strong classifiers' MTTRs, so that the detection accuracy can be guaranteed. Similar to [26], we adopt the concept of PAA and prove that the recall rate is PAA in the GAT search.

If the search grid equals to $\Delta x\times\Delta y$ and the recall rate is PAA, it should satisfy the following equation

$$\forall(u,v)\in\tau(\Delta x,\Delta y),\ P(P(H(\mathbf{I},u,v)<0)>\varepsilon)<\delta, \tag{13}$$

where $\mathbf{I}$ represents an image that contains only one object in it, the origin of the image is the left-top point of the object window, $H(\bullet)$ is the strong classifier, ε and δ are two positive real numbers close to 0. According to Eq. (13), if one detection window is located in the region $\boldsymbol{\tau}=(\Delta x,\Delta y)$, it guarantees that the recall rate is larger than $(1\text{-}\varepsilon)$, or else, the probability is less than δ.

Theorem 1. *If the search grid equals to the size of the associated strong classifier's MTTR, there exists (ε,δ) satisfying that the recall rate is (ε,δ) PAA in the search process.*

Proof. *Supposing that the coordinate of the object window is (0,0), i.e., the origin of the image **I**. If the search grid $(\Delta x,\Delta y)$ equals to the size of the strong classifier's MTTR, it guarantees that there is at least one window (x',y') in the region $\boldsymbol{\tau}(\Delta x,\Delta y)$ to be evaluated by the strong classifier. A basic assumption is that the training set has the same distribution as the testing set. Thus, the probability of misclassifying the object window (x',y') should be less than ε, i.e., $P(H(\mathbf{I},x',y')<0)<\varepsilon$, unless all the passed positive training samples are above the ε^{th} percentile of their distribution. The probability of such event is less than $(1-\varepsilon)^{NP}$, where N is the total number of the positive training samples and P is the positive pass rate of this strong classifier. It means that the miss rate or recall rate is (ε,δ) PAA in the GAT search, where δ equals to $(1-\varepsilon)^{NP}$.*

In Theorem 1, we suppose there is only one object in image I. If the object contains more than one object, Eq. (13) can guarantee the recall rate for each object separately. For example, we use 1000 positive training samples and set positive pass rate as 99.8% for each strong classifier. In this situation, 0.5% loss of recall rate is bounded by 0.7% $(=(1-0.005)^{1000*0.998})$. Thus, the recall rate of each strong classifier can be theoretically guaranteed with a high probability even when the classifier searches over a coarse grid. Since we attenuate the MTTRs during the training process, the search grids of the strong classifiers are arranged in descending order in the cascaded classifier. As a result, the search process is from coarse grid to fine grid. Fig. 5 shows the GAT search algorithm in a specific scale of an input image.

4. Experiments

We evaluate the proposed approach on the widely used datasets, namely VOC2006 dataset [25] and INRIA human dataset [6], containing both rigid objects (such as, bicycles, buses, cars and

Algorithm: Granularity-Adaptive-Tunable (GAT) search.

INPUT: Cascaded classifier: $\{H_t(\bullet),\boldsymbol{\tau}(\Delta x_t,\Delta y_t)\}_{t=1,2,\ldots,T}$, a specific scale of input image: **I**, pushing all detection windows in this scale into object queue: **Q.**

Output: Object queue: **Q**

```
for t = 1 : T do
    for All detection windows in Q on search grid Δx_t×Δy_t do
        Run strong classifier H_t(•) on detection window (x,y);
        if Detection window (x,y) is rejected by H_t(•) then
            Popup detection windows in region τ(x-Δx_t/2, y-Δy_t/2, Δx_t, Δy_t) from Q;
        end if
    end for
end for
```

Fig. 5. GAT search in a specific scale of input image.

motorbikes) and non-rigid objects (such as, cows, sheep, horses, cats, dogs, persons, INRIA human). For the VOC2006 dataset, we evaluate our approach on the overall 10 object classes. We use the given training set of VOC2006 dataset consisting of 2618 images and more details about this dataset can be found in [25]. For the positive training set, we utilize all the non-occluded objects in the training set. For the negative training set, we remove the objects and utilize the remaining images. We list the number of positive training samples used for each object class in Table 1. For INRIA human dataset, we utilize the positive samples and negative samples in the given training sets. We pad the images on the borders for detecting the objects around the borders, thus the average image sizes are 925×619 for VOC2006 testing set and 789×772 for INRIA testing set. The object classifier may generate multiple positive detection windows around one object, thus we adopt the mean-shift algorithm to merge the multiple detection results of the same object. To evaluate the accuracy, we follow the Average Precision (AP) protocol [25]. For a bounding box, it is considered as a correct detection when the overlapping area between the predicted bounding box B_p and ground-truth bounding box B_{gt} exceeds 50% by the following formula

$$\alpha_0 = \frac{\text{area}\left(B_p \cap B_{gt}\right)}{\text{area}\left(B_p \cup B_{gt}\right)}. \tag{14}$$

We manually specify the sizes of detection windows according to two criterions. First, the detection window sizes should be larger than most of the objects to be detected since the object detectors down-sample images to detect multiple scale objects. Second, the aspect ratio of the detection window should be close to the mean aspect ratio of the object class. After specifying the size of the detection window, a feature pool is offline generated in the detection window. To train our classifier, we generate different HOG feature pools for different object classes and sample the blocks in the detection window from 8×8 to the whole window in size and from 1:4 to 4:1 in aspect ratio. The detection window sizes, total numbers of features

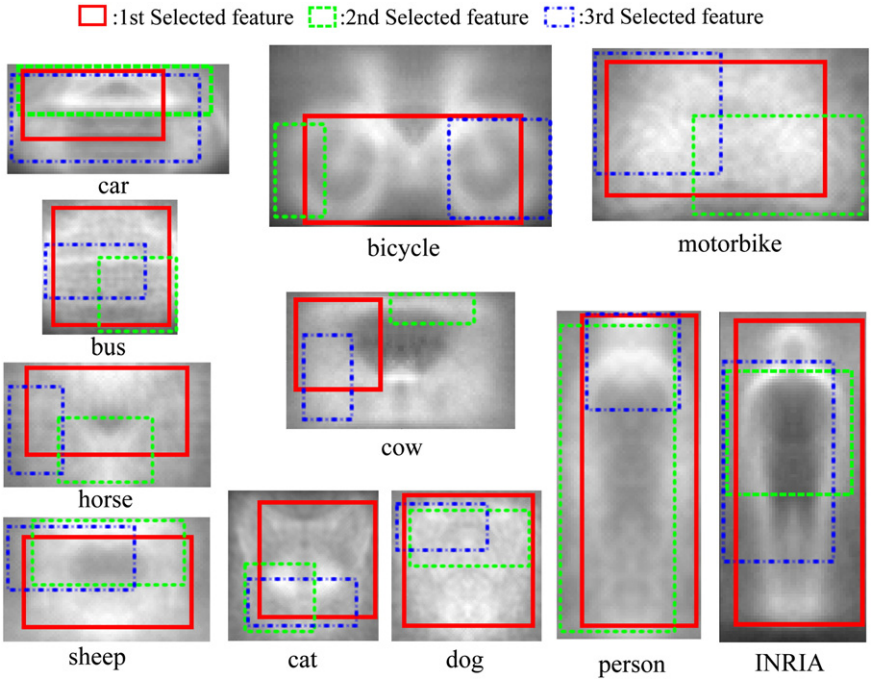

Fig. 6. First three selected features.

Table 2
Total numbers of selected features and ratios of selected features to total features in feature pools.

	Car	Bicycle	Bus	Motorbike	Person	Cow	Horse	Sheep	Cat	Dog	INRIA
Selected	1297	890	1014	1148	3198	1834	1946	3196	1299	1630	2022
Total	2520	6208	7568	6208	4960	3344	3344	3344	7568	7568	4960
Ratios	0.515	0.143	0.134	0.185	0.645	0.548	0.582	0.956	0.172	0.215	0.408

in the feature pool and positive training samples for each object class are listed in Table 1. During training, we set the initial MTTRs as $\boldsymbol{\tau}(16,16)$ and attenuate the MTTRs to $\boldsymbol{\tau}(8,8)$, $\boldsymbol{\tau}(4,4)$ and $\boldsymbol{\tau}(1,1)$ without considering $\boldsymbol{\tau}(2,2)$ (as will be discussed in Section 4.1).

We implement the proposed approach, baseline approach and multi-resolution SVM [30] with C++ using single thread programming without any code optimization. All the experiments are conducted on a desktop with an Intel Xeon E5440 (2.83 GHz) CPU and 16 GB memory.

4.1. Relationship between features and MTTRs

In this section, we study the relationship between the selected features and the strong classifiers' MTTRs. We visualize the first three features selected by our approach in Fig. 6, where the backgrounds are the averaged images of the aligned object edges. It is shown that most of the selected features are of large blocks. The possible reason is that the large blocks capture the global information and therefore the classifiers based on these features are insensitive to translation-variance. Besides the large blocks, some small blocks (e.g., the green block at the back of the cow) selected by the training algorithm and the classifiers based on these blocks may be only translation-insensitive in x-direction or y-direction. Nevertheless, there are many weak classifiers in a strong classifier and these weak classifiers can combine with and compensate for each other. As a result, the strong classifier is translation-insensitive in both x-direction and y-direction. In Table 2, we list the total numbers of the selected features by our approach and the ratios of the selected features to the overall features in the feature pools. It can be seen that the rigid object classes need less features since they have compact patterns. On the contrary, the non-rigid object classes (e.g., person, horse and sheep) have complex deformations and thus need more features for classifying them from background.

We also study the relationship between the positive pass rates and the strong classifiers' MTTRs on the INRIA person dataset. We select four strong classifiers from the proposed cascaded classifier. The sizes of their MTTRs are respectively 16×16, 8×8, 4×4 and 1×1. We resize the image to normalize the person window into the size of 48×108, then we translate the strong classifiers on the normalized image. We calculate positive pass rates (i.e., the ratios of the detection windows that are classified as objects to the total objects) on different translation-offsets for each strong classifier and show the results in Fig. 7. It can be seen that the larger the strong classifier's MTTR is, the more insensitive to the translation-variance the strong classifier is. The strong classifiers in Fig. 7 guarantee that 92% of the positive samples are correctly classified in its MTTR. Noticing that if the MTTR is $\boldsymbol{\tau}(1,1)$, it still guarantees 95% positive pass rate within the region $\boldsymbol{\tau}(2,2)$. The possible reason is that the annotations of the training samples cannot perfectly be aligned. Therefore, the strong classifier is still insensitive to the translation-variance to some extent even when its MTTR equals to $\boldsymbol{\tau}(1,1)$. In practice, we directly attenuate the MTTR to $\boldsymbol{\tau}(1,1)$ after $\boldsymbol{\tau}(4,4)$ without distinguishing the MTTRs that are smaller than $\boldsymbol{\tau}(4,4)$.

4.2. Comparing speed-versus-accuracy with other approaches

To compare speed and accuracy simultaneously, we propose a speed-versus-accuracy curve similar to [20]. A speed-versus-accuracy curve can be obtained by varying the scale stride, which controls the scanning granularity in the scale space. We use 20 different scale strides, i.e., 1.05^1, 1.05^2, 1.05^3, … , 1.05^{20}. After fixing a scale stride, we evaluate the AP and average execution time for the testing set, where the execution time is the total processing time consisting of the feature extraction time and search time. We can get 20 points using the 20 different scale strides, then we plot the envelop of the 20 points to visualize the relationship between speed and accuracy.

In this section, we compare our approach with three other approaches, namely the baseline approach, the multi-resolution SVM approach [30] and the holistic SVM approach [3]. The baseline approach learns strong classifiers based on the HOG features in the boosting framework similar to [5]. The only difference between the baseline approach and approach in [5] is the training algorithm of weak classifiers, i.e., LSR in the baseline approach and linear SVM in [5]. For the baseline approach, we evaluate four configurations by setting different search grids, i.e., 1×1, 4×4, 8×8 and 16×16. For comparison, we also implement the multi-resolution SVM approach [30] and the holistic SVM approach [3]. The multi-resolution SVM approach adopts a strategy from a coarse resolution to a fine resolution. The holistic SVM approach adopts the fixed grid search. The speed-versus-accuracy curves are shown in Fig. 8. For most

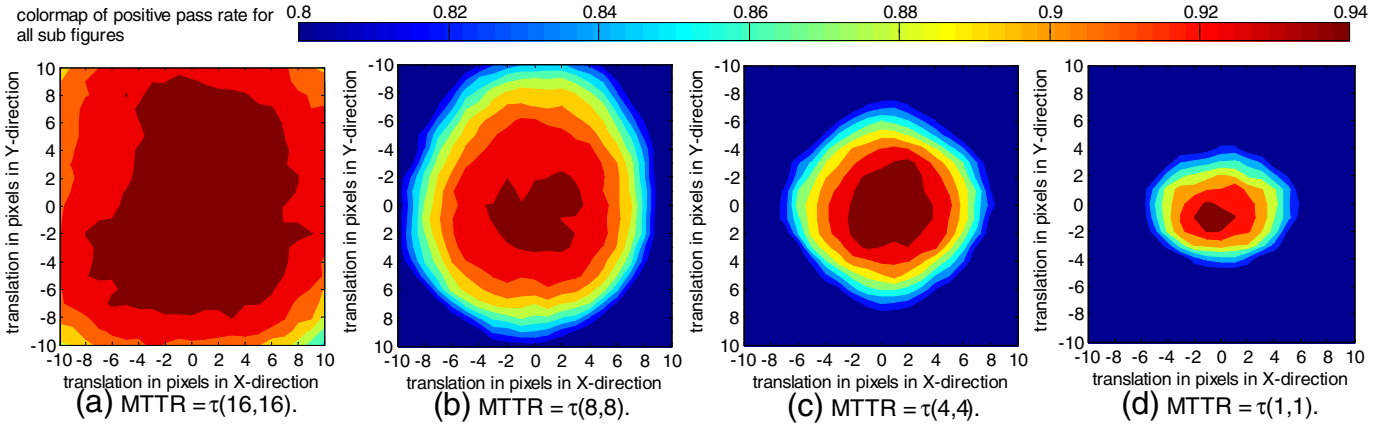

Fig. 7. Relationship between positive pass rates and strong classifiers' MTTRs.

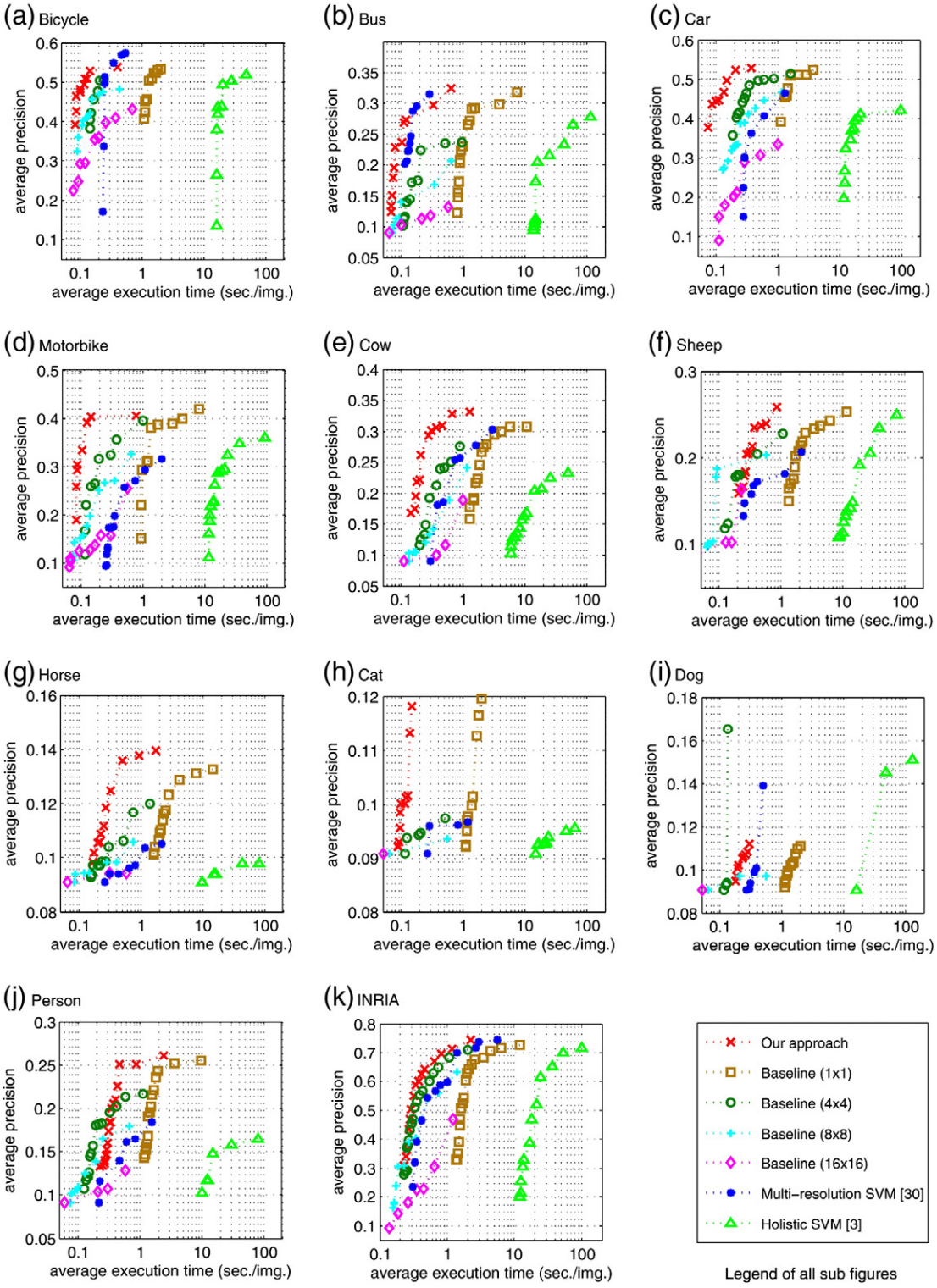

Fig. 8. Simultaneously comparing speed and accuracy between our approach and other related approaches.

Table 3
Comparing APs between other approaches and our approach. (Best APs are highlighted in bold.)

	Bicycle	Bus	Car	Motor	Cat	Dog	Horse	Sheep	Cow	Person	INRIA
Best in competition [25]	0.440	0.169	0.444	0.390	**0.160**	0.118	**0.140**	0.251	0.252	0.164	–
Holistic SVM [3]	0.520	0.278	0.412	0.360	0.095	**0.151**	0.098	0.250	0.232	0.165	0.718
Multi-resolution SVM [30]	**0.575**	0.282	0.465	0.315	0.097	0.139	0.106	0.206	0.303	0.184	0.743
Baseline approach (1×1)	0.537	0.318	0.524	**0.418**	0.120	0.111	0.133	0.253	0.309	0.256	**0.746**
Our approach	0.542	**0.324**	**0.530**	0.405	0.118	0.112	0.139	**0.259**	**0.321**	**0.261**	0.745

Table 4
Comparing maximal recall rates between baseline approach and our approach using different search strategies. Baseline approach adopts fixed grid search while our approach adopts both fixed grid search and GAT search.

	1×1		4×4		8×8		16×16		GAT
	Basel. (%)	Our (%)	Basel. (%)	Our (%)	Basel. (%)	Our (%)	Basel. (%)	Our (%)	Our (%)
Bicycle	63.6	63.8	61.8	62.5	58.7	58.9	51.6	51.8	63.2
Bus	67.0	65.7	60.5	60.0	50.2	53.2	31.3	34.7	65.6
Car	67.0	67.0	63.4	64.1	57.2	57.3	47.9	47.9	65.7
Motorbike	70.4	70.1	67.8	67.2	63.5	62.6	46.4	48.5	69.4
Cow	60.3	60.3	50.2	51.1	44.1	44.1	29.2	29.1	59.1
Sheep	53.8	54.0	46.4	49.2	39.8	42.6	25.1	28.9	53.6
Horse	54.9	54.0	50.0	47.9	45.1	44.5	27.8	24.9	53.4
Cat	35.4	36.1	22.6	24.2	11.1	12.3	6.3	8.8	34.7
Dog	44.6	43.7	35.2	36.7	2.2	9.2	1.6	6.7	43.7
Person	50.7	50.6	42.8	45.3	31.3	35.3	18.7	24.6	50.1
INRIA	91.1	90.3	86.7	83.4	79.4	79.5	71.2	71.3	90.3

cases, it can be seen that our approach is faster than the other three approaches at the same accuracy. The best APs of our approach are similar to the best APs of the baseline approach. The reason is that we use the same feature pool and the same training data to train the classifiers for both the baseline approach and our approach. However, our approach is always faster than the baseline approach when they achieve the same AP. Comparing with the multi-resolution SVM, the best APs of our approach are higher for most of the object classes. Moreover, our approach is faster than the multi-resolution SVM approach when they achieve the same AP. One possible reason is that our approach utilizes the boosting algorithm to select optimal features from a very large feature pool, while the multi-resolution SVM approach utilizes the handcrafted blocks of HOG features. Therefore, the features of our approach might be more efficient to reject the negative windows than that of the multi-resolution SVM approach. The holistic SVM approach is the slowest one. This approach does not adopt a rejection strategy to fast filter out the negative detection windows and it usually takes about one hundred seconds for processing one image if the search grid is 1×1. In Table 3, we compare the best APs of our approach with these three approaches. We also list the best competition results in [25]. Comparing with the other approaches, our approach achieves considerable accuracy on most of the object classes. For the classes with large variations, such as horses, cats and dogs, our approach also achieves comparable results. These three object classes are much harder to learn as the images contain large variations in poses and viewpoints. It is very hard to use a bounding box of a fixed aspect ratio to describe these animals. Other detection window based approaches [3,18,30] cannot achieve good performance on these classes either. Recently, some researchers have focused on the accuracy issue and achieved the state-of-the-art accuracy on these two datasets. For example, [18] proposes the deformable part model and mixture models to handle occlusion, deformation and different aspect ratios. [13,14] utilize heterogeneous features to build more powerful classifiers. These approaches achieve higher APs than our approach. However, this work mainly focuses on the speed issue. Our approach can cooperate with these off-the-shelf part models and/or heterogeneous features, thus the accuracy of our approach is promising to be improved.

Table 6
Comparing maximal recall rates of different search strategies based on our approach.

	1×1		4×4		8×8		16×16		GAT	
	Recall (%)	Loss	Recall (%)	Loss (%)	Recall (%)	Loss (%)	Recall (%)	Loss (%)	Recall (%)	Loss (%)
Bicycle	63.8	–	62.5	1.3%	58.9	4.9	51.8	12.0	63.2	0.6
Car	65.7	–	60.0	5.7	53.2	12.5	34.7	31.0	65.7	0.0
Bus	67.0	–	64.1	2.9	57.3	9.7	47.9	19.1	65.6	1.4
Motorbike	70.1	–	67.2	2.9	62.6	7.5	48.5	21.6	69.4	0.7
Cow	60.3	–	51.1	9.2	44.1	16.2	29.1	31.1	59.1	1.2
Sheep	54.0	–	49.2	4.8	42.6	11.4	28.9	25.1	53.6	0.4
Horse	54.0	–	47.9	6.1	44.5	9.5	24.9	29.1	53.4	0.6
Cat	36.1	–	24.2	11.9	12.3	13.8	8.8	27.3	34.7	1.4
Dog	43.7	–	36.7	7.0%	9.2	34.5	6.7	37.0	43.7	0.0
Person	50.6	–	45.3	5.3	35.3	15.3	24.6	26.0	50.1	0.5
INRIA	90.3	–	83.4	6.9	79.5	10.8	71.3	19.0	90.3	0.0

4.3. Comparing different search strategies

In this section, we study the search strategy. Enlarging the search grids is a straightforward way to accelerate the search process [3,8]. We show that this straightforward strategy causes loss of the recall rate, while the GAT search is capable to guarantee the recall rate as well as accelerate the detection process.

We set the scale stride as 1.05^2 and evaluate the baseline and proposed classifiers trained in Section 4.2. The baseline approach adopts the fixed gird search strategy, while the proposed approach adopts both the fixed grid search and GAT search. The fixed grid search includes four different configurations, i.e., 1×1, 4×4, 8×8 and 16×16. The maximal recall rates and APs are respectively listed in Tables 4 and 5. Apparently, the 1×1 grid search always achieves the best performance for both the baseline approach and our approach. The accuracy dramatically drops when we enlarge the search grid, especially when the search grid is larger than 4×4. Comparing with the fixed grid search, the GAT search can guarantee the accuracy. Our approach utilizes more positive training samples than the baseline approach, since additional positive samples are generated in the MTTRs for our approach. From Tables 4 and 5, we can see that such additional positive samples do not obviously

Table 5
Comparing APs between baseline approach and our approach using different search strategies. Baseline approach adopts fixed grid search while our approach adopts both fixed grid search and GAT search.

	1×1		4×4		8×8		16×16		GAT
	Basel.	Our	Basel.	Our	Basel.	Our	Basel.	Our	Our
Bicycle	53.5	53.9	50.4	53.3	48.2	49.3	43.1	44.1	54.2
Bus	31.8	31.5	23.6	27.2	20.6	23.4	13.1	15.5	32.4
Car	52.5	54.2	51.4	51.8	46.6	45.3	33.5	35.0	53.0
Motorbike	42.0	40.8	39.6	35.3	32.5	32.5	25.5	24.8	40.5
Cow	30.8	33.0	27.2	28.6	24.1	24.7	18.8	14.8	32.1
Sheep	25.3	24.8	22.8	22.7	20.3	20.3	16.4	15.7	24.8
Horse	13.3	13.8	12.0	11.6	10.6	11.2	9.4	10.2	14.0
Cat	12.0	10.6	9.7	9.7	9.4	9.8	9.1	9.1	11.8
Dog	10.2	10.5	10.0	10.4	9.7	10.0	9.1	9.4	11.2
Person	25.7	26.4	21.7	23.6	17.9	19.6	12.9	14.7	26.1
INRIA	72.8	73.6	71.1	73.4	63.2	66.3	46.9	49.3	74.5

Table 7
Comparing detection speed between baseline approach and our approach.

	Baseline 1×1 (sec.)		Our approach (sec.)		Speedup	
	Total	Search	Total	Search	Total	Search
Bicycle	4.9	4.5	0.40	0.05	×12	×90
Bus	4.0	3.6	0.34	0.04	×12	×90
Car	5.3	4.9	0.39	0.08	×14	×61
Motorbike	4.3	3.9	0.42	0.12	×10	×33
Cow	5.8	5.4	0.66	0.23	×8.8	×24
Sheep	6.2	5.8	0.87	0.57	×7.1	×10
Horse	7.7	7.4	0.91	0.61	×8.5	×12
Dog	5.0	4.6	0.69	0.39	×7.2	×12
Cat	4.9	4.6	0.47	0.18	×10	×26
Person	5.3	4.9	0.85	0.51	×6.2	×10
INRIA	7.6	7.0	1.13	0.58	×6.9	×12

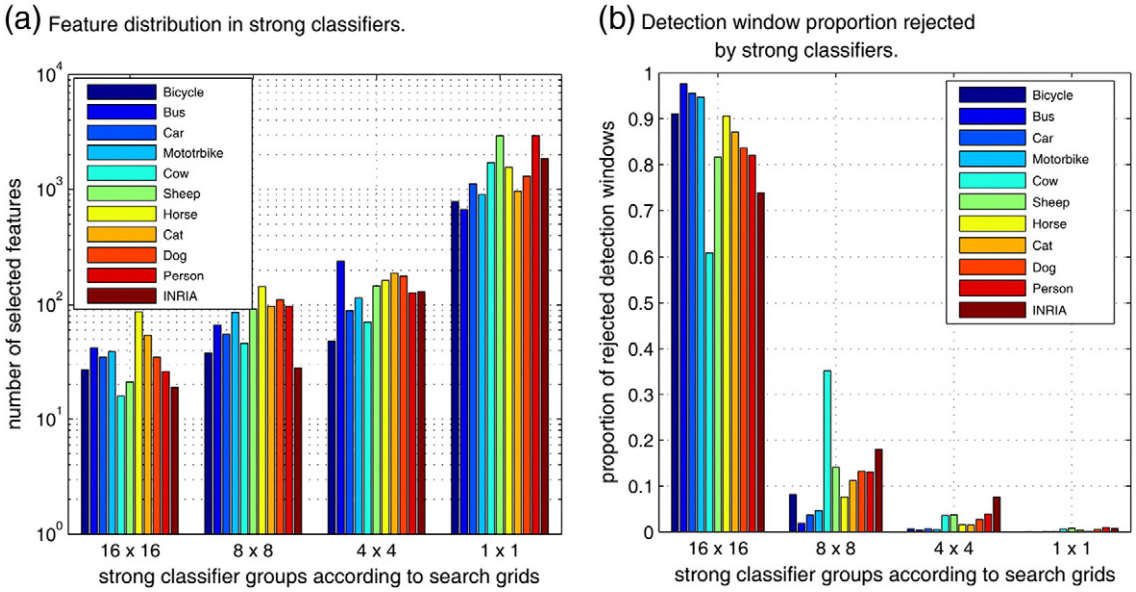

Fig. 9. Feature distributions and proportions of rejected detection windows in groups. Strong classifiers are categorized into four groups according to their search grids.

improve the maximal recall rates and APs comparing with the baseline approach. However, these samples contribute to learning the strong classifiers' MTTRs and thus accelerate the detection process.

Enlarging the search grid is a trade-off between accuracy and speed, and the object windows may be missed when the detector searches over a coarse grid. For our approach, we utilize five search configurations, i.e., 1×1, 4×4, 8×8, 16×16 and GAT. We compare the maximal recall rates and the loss rates of different search strategies in Table 6. It can be seen that the 1×1 grid search always gives the highest recall rates. Enlarging the search grid unavoidably decreases the recall rate especially when the search grid is larger than 4×4. However, the GAT search successfully guarantees that the loss of recall rate is less than 1.4%.

4.4. Comparing computation cost between baseline approach and our approach

In this section, we give an end-to-end speed comparison of the detection process as well as a detailed analysis. In Table 7, we set the scale stride as 1.05^2 and compare the detection speed between the baseline approach and our approach. The baseline approach adopts the 1×1 grid search while our approach adopts the GAT search. The total execution time consists of two major parts, i.e., feature extraction time and search time. We list the total execution time and the search time in Table 7. It can be seen that our approach is about 10–90 times faster in the search process and 6.2–14 times faster in the total detection process. For the baseline system with 1×1 grid search, the search time is the bottleneck while the feature

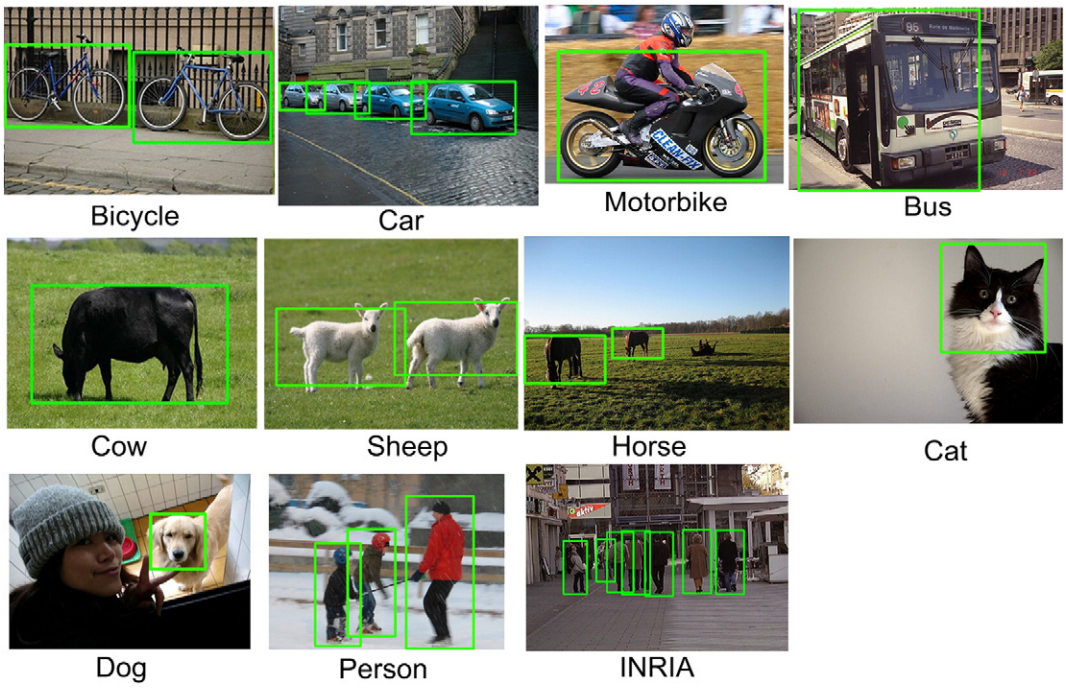

Fig. 10. Some detection results of VOC2006 dataset and INRIA person dataset.

extraction time is about 0.3–0.4 seconds on average for an image. Our approach dramatically reduces the search time especially for the rigid objects. As a result, the feature extraction becomes the bottleneck in the total execution time in our approach, but it is beyond the scope of this work. Accelerating feature extraction is independent with our speedup scheme, which means that our approach can cooperate with the feature acceleration approach [29] for further acceleration.

We also study the distributions of the features and the rejected detection windows in the GAT search process. The cascaded classifier can be categorized into four groups, i.e., 16×16, 8×8, 4×4 and 1×1, according to the strong classifiers' MTTRs. We list the features used in each group in Fig. 9(a) and the proportion of the detection windows rejected by each group in Fig. 9(b). We can see that the rigid objects need fewer features while the non-rigid objects need more features and draw similar conclusions from Table 2. For rigid objects, there are about 99.5% of detection windows rejected by the strong classifiers in the groups of 16×16 and 8×8. Taking the bus detector as an example, the first 42 features can reject 97.7% detection windows with 16×16 grid search and the search time are finally accelerated by 90 times. For non-rigid objects, there are about 95% of the detection windows rejected by the strong classifiers in the groups of 16×16 and 8×8. The possible reason is that there are more translation-insensitive structures on the rigid objects than that on the non-rigid objects, and such structures help to learn the translation-insensitive classifiers to speed up the search process. That is why the speedup of the rigid object detection is larger than that of non-rigid object detection. Finally, we show some detection results in Fig. 10.

5. Conclusion

In this paper, we propose an approach to accelerate object detection with a guaranteed recall rate. We define the concept of TTR for a classifier. Then, we utilize the MTTR to measure the classifiers' sensitivities to the translation-variance. Furthermore, we propose a boosting algorithm to learn the cascaded classifier with the strong classifiers' MTTRs being arranged in descending order of their sizes. As a result, the search grid can be adaptively tuned in a coarse-to-fine manner by the strong classifiers' MTTRs, leading to the acceleration of the detection process. Moreover, we prove that the recall rate is PAA in the GAT search, which means that the accuracy is theoretically guaranteed with a certain probability. Extensive experimental results on VOC2006 dataset and INRIA person dataset show the effectiveness and efficiency of the proposed method.

Since the boosting framework does not rely on the feature types, the proposed approach is able to cooperate with many other existing features such as covariance matrix and heterogeneous features. It is also promising to combine the proposed approach with the feature acceleration approaches to speed up the detection process further.

Acknowledgments

This paper is partially supported by National Basic Research Program of China (973 Program) under contract 2009CB320902; Natural Science Foundation of China under contracts Nos. 60872124, 60833013, and 60832004; and Beijing Natural Science Foundation (New Technologies and Methods in Intelligent Video Surveillance for Public Security) under contract No. 4111003.

References

[1] P. Viola, M. Jones, Rapid object detection using a boosted cascaded of simple features, CVPR, 2001.
[2] W. Zheng, L. Liang, Fast car detection using image strip features, CVPR, 2009.
[3] N. Dalal. Finding People in Images and Videos. PhD thesis, Institut National Polytechnique de Grenoble, 2006.
[4] P. Kelly, N.E. O'Connor, A.F. Smeaton, Robust pedestrian detection and tracking in crowded scenes, Image Vision Comput. 27 (10) (2009) 1445–1458.
[5] Q. Zhu, S. Avidan, M.C. Yeh, K.T. Cheng, Fast human detection using a cascade of histograms of oriented gradients, CVPR, 2006.
[6] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, CVPR, 2005.
[7] I. Laptev, Improving object detection with boosted histograms, Image Vision Comput. 27 (5) (2009) 535–544.
[8] O. Tuzel, F. Porikli, P. Meer, Pedestrian detection via classification on Riemannian manifolds, PAMI 30 (2008) 1713–1727.
[9] Hongming Zhang, Wen Gao, Xilin Chen, Debin Zhao, Object detection using spatial histogram features, Image Vision Comput. 24 (4) (2006) 327–341.
[10] V. Ferrari, L. Fevrier, F. Jurie, C. Schmid, Groups of adjacent contour segments for object detection, PAMI 30 (2008) 36–51.
[11] B. Wu, R. Nevatia, Cluster boosted tree classifier for multi-view, multi-pose object detection, ICCV, 2007.
[12] P. Dollár, Z. Tu, H. Tao, S. Belongie, Feature mining for image classification, CVPR, 2007.
[13] B. Wu, R. Nevatia, Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection, CVPR, 2008.
[14] X. Wang, T.X. Han, S. Yan, An HOG-LBP human detector with partial occlusion handling, ICCV, 2009.
[15] Z. Lin, G. Hua, L.S. Davis, Multiple instance feature for robust part-based object detection, CVPR, 2009.
[16] P. Viola, J.C. Platt, C. Zhang, Multiple instance boosting for object detection, NIPS, 2005.
[17] A. Vedaldi, V. Gulshan, M. Varma, A. Zisserman, Multiple kernels for object detection, ICCV, 2009.
[18] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part based models, PAMI 32 (2010) 1627–1645.
[19] S. An, P. Peursum, W. Liu, S. Venkatesh, Efficient algorithms for subwindow search in object detection and localization, CVPR, 2009.
[20] N.J. Butko, J.R. Movellan, Optimal scanning for faster object detection, CVPR, 2009.
[21] C.H. Lampert, M.B. Blaschko, T. Hofmann, Beyond sliding windows: object localization by efficient subwindow search, CVPR, 2008.
[22] H.A. Rowley, S. Baluja, T. Kanade, Human face detection in visual scenes, NIPS, 1996.
[23] H. Schneiderman, Feature-centric evaluation for efficient cascaded object detection, CVPR, 2004.
[24] S. Maji, A. Berg, J. Malik, Classification using intersection kernel support vector machines is efficient, CVPR, 2008.
[25] M. Everingham, A. Zisserman, C. Williams, L.V. Gool, The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results, 2006.
[26] P. Felzenszwalb, R. Girshick, D. McAllester, Cascade object detection with deformable part models, CVPR, 2010.
[27] François Fleuret, Donald Geman, Coarse-to-fine face detection, Int. J. Comput. Vis. 41 (1/2) (2001) 85–107.
[28] Hichem Sahbi, Donald Geman, Nozha Boujemaa, Coarse-to-fine support vector classifiers for face detection, ICPR, 2002.
[29] P. Dollár, S. Belongie, Pietro Perona, The fastest pedestrian detector in the west, BMVC, 2010.
[30] W. Zhang, G. Zelinsky, D. Samaras, Real-time accurate object detection using multiple resolutions, ICCV, 2007.
[31] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (3) (1999) 297–336.