Int J Comput Vis (2017) 123:334–349
DOI 10.1007/s11263-016-0986-2

Visual Object Detection Using Cascades of Binary and One-Class Classifiers

Hakan Cevikalp[1] · **Bill Triggs**[2]

Received: 8 November 2014 / Accepted: 23 December 2016 / Published online: 11 January 2017

Abstract We describe an efficient approach to visual object detection that uses short cascades of asymmetric 'one class' classifiers to quickly reject negatives (windows not centered on an object of the desired class) within a sliding window framework. Current detectors typically use binary discriminants such as Support Vector Machines or Boosting to implement each stage of the cascade. These treat the positive and negative classes symmetrically. We argue that this is suboptimal because object detectors typically see a great many negative windows with extremely diverse contents and only a few positive ones with comparatively coherent contents. We show that asymmetric representations that focus on tightly modeling the extent of the rare, coherent positive class can lead to simpler classifiers and faster rejection. Our cascades use asymmetric classifiers based on simple convex models to progressively tighten the bound on the positive class. They typically start with a conventional linear SVM for initial pruning, followed by a cascade of linear distance-to-hyperplane and interior-of-hypersphere classifiers and finishing with a kernelized hypersphere classifier. We show that the resulting detectors have competitive performance on the Labeled Faces in the Wild dataset and state-of-the-art performance on the FDDB face detection, ESOGU face detection and INRIA Person datasets. The results on the PASCAL VOC 2007 dataset are also respectable given that they use neither object parts nor context. The one-class formulations provide significant reductions in classifier complexity relative to the corresponding two-class ones, making them suitable for real-world applications.

Keywords Object detection · Rejection cascade · One-Class Classifier · Latent training

Communicated by Takayuki Okatani.

✉ Hakan Cevikalp
hakan.cevikalp@gmail.com

Bill Triggs
Bill.Triggs@imag.fr

[1] Electrical and Electronics Engineering Department, Eskisehir Osmangazi University, Eskisehir, Turkey

[2] Laboratoire Jean Kuntzmann, Grenoble, France

1 Introduction

Object class detection is an important computer vision task in which all instances of a given generic object class that occur in an image must be recovered and labeled with their image positions and scales. Given its many applications (video surveillance, automatic target detection, content-based image retrieval, driver-assistance systems, interactive games,...) it has received a great deal of attention, but despite significant advances it remains challenging. Natural categories such as people, dogs or chairs have a bewildering variety of shapes, deformations and appearances, and even for a known instance, view-point changes can produce large variations in image layout and scale. Lighting variations, complex backgrounds, occlusion and truncation make the problem still harder.

Two factors that are critical for an object detection method are the features used to encode the image content and the classifiers used to make object/non-object decisions based on them. Regarding features, early detectors used raw pixel values (Rowley et al. 1998), wavelets (Papageorgiou and Poggio 2000), edges (Amit and Geman 1999), and Gabor filter responses (Shams and Speslstra 1996). Histogram-based features have also become very popular owing to their efficiency and good performance. Many of the histogram-based feature sets are based on oriented image gradients, including

SIFT (Lowe 2004), SURF (Bay et al. 2008), Histogram of Oriented Gradients (HOG) (Dalal and Triggs 2005), PHOG (Vedaldi et al. 2009), Generalized Shape Context (Belongie et al. 2002) and Local Edge Orientation Histograms (Levi and Weiss 2004). Others are based on local patterns of qualitative gray-level differences, including Local Binary Patterns (LBP) (Ahonen et al. 2006; Wang et al. 2009), and Local Ternary Patterns (LTP) (Tan and Triggs 2010). More recently, CNN (Convolutional Neural Network) (Girshick et al. 2014; Li et al. 2015; Angelova et al. 2015) features learned by the supervised deep neural networks dominated the field. Empirically, the best feature set depends on the application and new ones are being developed all the time. Many recent methods combine several sets for better results, *e.g.* , simply concatenating them to form an extended feature vector (Harzallah et al. 2009; Wang et al. 2009; Hussain and Triggs 2010), finding optimal combination coefficients at the learning stage (Vedaldi et al. 2009; Varma and Ray 2007), or using boosting or other sparse methods to select informative subsets of features (Viola and Jones 2004; Perrotton et al. 2010).

Regarding the decision rule, this must use the features to decide whether the moving detection window currently contains a correctly framed class instance or something else (background, a partial or incorrectly framed instance, another class, *etc*.). This discrimination problem is usually formulated in a way that treats the "object" and "anything else" categories symmetrically. Many formalisms have been used (nearest neighbors, classification trees, probabilistic models, neural, convolutional or deep networks,...) but two have received much of the attention owing to their interesting properties: Boosting based cascades, and Support Vector Machines. The seminal work of Viola and Jones (2004) produced a very efficient face detector by using AdaBoost to train a cascade of pattern-rejection classifiers over rectangular wavelet features. Li et al. (2015) and Angelova et al. (2015) applied boosting cascade with CNN features whereas Benenson et al. (2012) and Orozco et al. (2015) used boosting cascade classifiers with HOG like features for face and pedestrian detection tasks. Each stage of the boosting cascade is designed to reject a considerable fraction of the negative examples that survived to that stage while passing almost all of the positives. As a result, the majority of windows that do not contain object class are rejected early in the cascade with comparatively little computation. Rejection typically becomes harder as the cascade progresses, so the one-stage classifiers grow in complexity.

Although Boosting based cascades give excellent results for real-time face/pedestrian detection, Support Vector Machine (SVM) classifiers are currently a more common choice for general object detection under less stringent time constraints (Harzallah et al. 2009; Felzenszwalb et al. 2008; Dalal and Triggs 2005; Vedaldi et al. 2009; Aldavert et al. 2010; Girshick et al. 2014). Linear SVMs are usually preferred for their simplicity and speed although it is well-established that kernel SVMs typically give higher accuracy at the cost of greatly increased computational complexity (Vedaldi et al. 2009). For this reason, several state-of-the-art methods use short cascades in which the early stages use linear SVMs to reject most of the negative windows quickly, whereas the later stages use nonlinear SVMs to make the final decisions (Harzallah et al. 2009; Vedaldi et al. 2009). Instead of using a cascade, Vedaldi and Zisserman (2012) approximate the nonlinear SVM kernels by explicitly mapping data onto a higher dimensional space, and report much higher accuracies over linear SVMs for pedestrian detection. Malisiewicz et al. (2011) introduced Ensemble of Exemplar SVMs which uses an ensemble of linear SVM classifiers trained with a single positive example. However using a single positive example for each classifier makes the classification problem extremely imbalanced and the method does not generalize well as demonstrated in our experiments. Furthermore, the detector is too slow for real-time applications since the testing time is linearly related to the number of positive samples in the training set.

Although symmetric binary classifiers are currently the norm, several detectors have exploited 'one class' methods, *i.e*. classifiers that abandon the formal equivalence between the positive and negative classes and adopt asymmetric representations or loss functions designed to provide tighter modeling of the positive class.[1] For example, Jin et al. (2004) used a kernelized hypersphere classifier for face detection. This gives an accurate approximation to the face class but it is computationally expensive owing to the need to evaluate kernels against many support vectors. To decrease the run time they divide the detection window into nine blocks and apply the nonlinear classifier only if it passes various heuristic tests such as eye regions being darker than cheeks and the bridge of nose. Thus, the method applies only to face detection. The preliminary version of the current paper used a cascade of linear hyperplane and linear/kernelized hypersphere models for face and person detection (Cevikalp and Triggs 2012). In work done independently of ours, Scheirer et al. (2013) confirm that object detectors can be improved by using one-class methods.

Another important issue is the method used to conduct the image search. Naive sliding window approaches tend to be prohibitively slow because they must evaluate their full feature vectors and window-level classifiers at every possible location and scale in the image. Methods such as integral images (Viola and Jones 2004), integral his-

[1] The name "one class" is conventional. It emphasizes the origin of these methods in density modeling and the predominant role of the positive class but it is something of a misnomer in that negative examples usually can be, often are, and in some formulations must be included during training.

tograms (Porikli 2005), distributive histograms (Sizintsev et al. 2010) and other histogram-based approaches (Wei and Tao 2010) are often used to facilitate feature computation. Similarly, cascade-based classifier architectures (Viola and Jones 2004) allow unpromising windows to be pruned without a full evaluation, and alternatives such as branch-and-bound search have also been developed (Lampert et al. 2008). More recently, Girshick et al. (2014) used a sophisticated algorithm to return the most promising candidate windows to evaluate classifiers.

Many authors have demonstrated the benefits of flexible object models that allow several possible pose or appearance classes to compete and that incorporate movable object parts or fragments to deal with finer pose and shape variations. Notably, Felzenszwalb et al. (2010b) describe an elegant two-level framework in which several root appearance models compete, each with its own set of movable parts located in such a way as to optimize the score of a global feature vector that includes both root and part features. One of its key contributions is a 'latent SVM' methodology that not only estimates the unknown part position variables and appearance-class labels during both training and testing, but also provides a fine-tuning of the labeled instance positions during training, thus greatly sharpening the resulting models and allowing training from less precise annotations. This very successful approach has been extended in several ways, for example to multiple layers (Zhu et al. 2010) and cascade deformable part models (DPMs) (Felzenszwalb et al. 2010a) which speeds up the DPMs detector hierarchically by pruning low scoring hypotheses obtained from the best configurations of subsets of the parts.

The present paper describes object detectors based on short cascades consisting of a linear SVM pre-processor followed by series of one-class methods, first a sequence of linear distance-to-hyperplane classifiers, then linear and kernelized interior-of-hypersphere classifiers. The algorithm allows for multiple roots (appearance classes) and latent training, but for simplicity it does not currently incorporate parts. These could easily be added after the final stage classifier (*c.f.* Cevikalp et al. 2013) but we have not done this below so that the experiments can focus on the performance of the cascade process itself. It is not clear how useful parts would be in earlier stages of the cascade owing to their high computational cost relative to the root and the issue of whether and how to maintain part-position consistency down the cascade. To this end, a similar approach can be used as in Felzenszwalb et al. (2010a) by replacing SVM classifier with the proposed cascade of linear classifiers and estimating a separate threshold for each classifier in the cascade and using these thresholds hierarchically for pruning low scoring hypotheses.

A preliminary version of this work appeared in Cevikalp and Triggs (2012). The current paper adds multiple roots, latent training, an alternative form of hyperplane classifier, a reduced set method, many small refinements, and additional experiments and discussion.

2 Method

In sliding window object detectors, the positives are the detection windows that correctly frame a valid instance of the desired class, whereas the negatives are the windows that contain anything else at all, including background, other classes and invalid or incorrectly framed instances. The classification problem is thus both imbalanced—only a tiny fraction of windows are positives—and asymmetric—the positives typically form a compact, coherent group, while the negatives (being defined negatively) are much more diverse and hence harder to model. This is reflected in the learned classifiers. For example in SVM based detectors it is common to find that most of the support vectors are negative ones, needed to characterize the many ways in which a window can fail to be a valid, well-framed class instance. This happens even when the bulk of the negative population is well separated from the decision surface and hence comparatively easy to classify: the remaining 'hard negatives' (ones close to the decision surface) still outnumber the positives and it is the necessity of finding and incorporating these that makes the learning problem so challenging. In fact, with current feature sets it is not uncommon to find that the hard negatives completely surround the positives in feature space—*c.f.* the scatter plots of projected class densities in Hussain (2011), and also its observation that many detectors that work extremely well using their final thresholds actually misclassify most of their training positives if the raw threshold returned by the SVM is used instead. As a result, linear SVMs are not very reliable for object detection in the sense that they need a fine tuning of the error penalty parameter C (small changes of this parameter causes a large drop in accuracy), and increasing training set size may decrease the detection performance instead of an improvement in the accuracy (Zhu et al. 2012).

Given this, it seems appropriate to use asymmetric ('one class') classifiers that focus on tightly bounding the positive class and/or multi-stage architectures that can rapidly prune away the mass of easy negatives and progressively cut out a compact, coherent positive region from a broad sea of harder negatives. Our approach combines these principles by using a cascade of one class classifiers to enforce increasingly restrictive convex bounds on the positive class. The current version has four stages. The first uses a standard linear SVM to eliminate the bulk of the easy negatives and thus reduce the training set to a more manageable size. The second applies a series of distance-to-hyperplane tests (*i.e.* absolute values of linear forms, $|\mathbf{w}_k^\top \mathbf{x} + b_k| \leq \tau_k$), thus lim-

iting the positives to a (usually open-ended) parallelepiped. The third limits them to the interior of an affine hypersphere by enforcing a distance constraint $\|\mathbf{x}-\mathbf{c}\| \leq r$. Finally, the fourth stage uses either a kernelized hypersphere classifier or a kernelized SVM to enforce further nonlinear bounds. We do not claim that this particular sequence is optimal but in practice it does seem to provide effective search pruning at a modest computational cost—essentially a single dot product for each classifier except the last. There are many other forms of region-bounding classifiers that could be tried including lower-dimensional affine subspaces, convex hulls, and hyper-disks and hyper-ellipsoids (Cevikalp and Triggs 2008; Cevikalp et al. 2010).

Our proposed cascade classifier differs from other cascade detectors (Viola and Jones 2004; Harzallah et al. 2009; Malisiewicz et al. 2011; Li et al. 2015; Angelova et al. 2015; Benenson et al. 2012) in the way that we focus on approximating positive class regions by using one-class classifiers and use the distances to the estimated positive class regions to accept/reject test windows. As a result, the proposed cascade classifier achieves very good accuracies for face and pedestrian detection tasks where the positive class samples have a compact distribution. To deal with diversified appearances resulting from sparse and irregular distributions, we need a sophisticated clustering algorithm that will discover compact subgroups before application of the method. However, as the number of positive samples is increased, the sparse and irregular distributions of positive classes will become denser and it will be a more smooth nonlinear manifold as illustrated in Fig. 1. Therefore, we believe that our proposed cascade classifier will also be successful for such cases. We now present each stage in detail, then describe the training methodologies used for our basic and multi-root object detectors.

2.1 Stage 1: Linear SVM

The first stage of the cascade simplifies the task of the later stages by using a linear SVM to eliminate as many of the easier negatives as possible. To fix notation, we work with input examples (feature vectors) $\mathbf{x} \in \mathbb{R}^d$ and their class labels $y \in \{-1, +1\}$: $+1$ for the positive (object) class and -1 for the negative (background) one. The SVM classifies an example as a positive if and only if $\mathbf{w}^\top \mathbf{x} + b > 0$, where $\mathbf{w}$ is the SVM's weight vector and b is its offset. For training we are given a set of labeled examples $\{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, n$. The formulation (Cortes and Vapnik 1995) tries to find a $\mathbf{w}$, b that not only correctly classify the training samples, but that separate the positives from the negatives by a fixed margin of ± 1, *i.e.* $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$ for all training samples. It does this by solving a convex program that penalizes both large $\|\mathbf{w}\|$ and margin violations:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1 - \xi_i . \end{aligned} \tag{1}$$

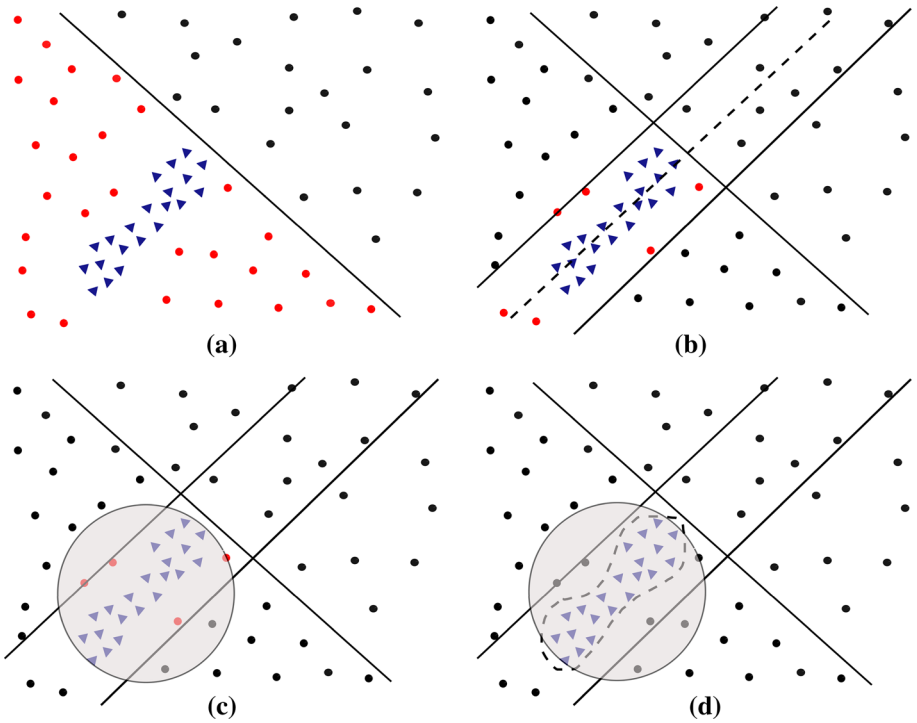

Fig. 1 (Best viewed in *color*). An illustration of pruning in our four-stage cascade. Samples from the object (positive) and background (negative) classes are shown respectively as *blue triangles* and *black dots*. **a** The first stage eliminates most of the easier negatives using a linear SVM. The surviving negatives are shown as *red dots*. **b** The second stage retains only the samples that lie close to each of a set of hyperplanes (the *dashed line* and its two borders). **c** The third stage retains only the samples that lie within a hypersphere (*i.e.* close to its central point). **d** The final stage is a kernelized classifier that confines the surviving samples to a nonlinear acceptance region (Color figure online)

Here, the ξ_i are slack variables that quantify any margin violations and C is an error penalty set by the user. The formalism can be used to learn nonlinear classifiers by applying the 'kernel trick'.

SVM training has received a lot of attention and there are efficient algorithms that allow large problems to be handled reliably. We used LIBOCAS[2] to train linear SVMs and LIBSVM[3] to train kernelized ones.

2.2 Stage 2: Distance-to-Hyperplane Classifiers

The second stage of the cascade applies a series of filters of the form $|\mathbf{w}_k^\top \mathbf{x} + b_k| \leq \tau_k$ to the examples, *i.e.* it constrains them to lie in a (usually open and unbounded) parallelepiped formed by the intersection of a set of slab shaped regions [the points within distance τ_k of hyperplane $(\mathbf{w}_k, b_k)$]. Ideally, for each hyperplane in turn, the remaining positives should lie close to it while the remaining negatives lie far from it. The problem is thus one of robust hyperplane fitting to the positives, penalized by closeness to the negatives. This is potentially non-convex with a combinatorial number of local minima: if the negatives lie in several groups the method must decide which groups should lie to the left of the hyperplane and which to the right of it. We tested two algorithms that contrive to avoid this issue. The first adopts an algebraic distance instead of a Euclidean one, reformulating the fit as a generalized eigenproblem. The second requires all of the negatives to lie to the left, reformulating the fit as a quadratic program that generalizes linear SVM. In either case, a sequence of filters is then found by repeatedly fitting a new hyperplane to the surviving training examples and deleting the examples that it eliminates.

Algebraic Method: Let $\mathbf{X}_+$ and $\mathbf{X}_-$ be matrices whose rows are the surviving training samples of respectively the positive and the negative classes. For convenience, define extended matrices $\bar{\mathbf{X}}_\pm = [\mathbf{X}_\pm \ \mathbf{e}_\pm]$ where $\mathbf{e}_\pm$ are corresponding column vectors of ones. The hyperplane that gives the best least-squares fit to the positive data alone is

$$\min_{\mathbf{w},b,\|\mathbf{w}\|=1} \|\mathbf{X}_+ \mathbf{w} + \mathbf{e}_+ b\|^2 \ = \ \min_{\mathbf{z}} \frac{\mathbf{z}^\top \mathbf{G}\, \mathbf{z}}{\|\mathbf{w}\|^2}, \tag{2}$$

where $\mathbf{z} = \binom{\mathbf{w}}{b}$ and $\mathbf{G} = \bar{\mathbf{X}}_+^\top \bar{\mathbf{X}}_+$. To give a background-sensitive fit (Mangasarian and Wild 2006), we instead minimize the regularized Rayleigh quotient

$$\min_{\mathbf{w},b,\|\mathbf{w}\|=1} \frac{\|\mathbf{X}_+ \mathbf{w} + \mathbf{e}_+ b\|^2}{\|\mathbf{X}_- \mathbf{w} + \mathbf{e}_- b\|^2 + \delta\,(\|\mathbf{w}\|^2 + b^2)}, \tag{3}$$

which can be re-expressed as

$$\min_{\mathbf{z}} \frac{\mathbf{z}^\top \mathbf{G}\, \mathbf{z}}{\mathbf{z}^\top \mathbf{H}\, \mathbf{z}} \tag{4}$$

where $\mathbf{H} = \bar{\mathbf{X}}_-^\top \bar{\mathbf{X}}_- + \delta \mathbf{I}$ and δ is a user-set regularization constant. The solution reduces to finding the smallest-λ eigenvector of the generalized eigenproblem $\mathbf{G}\,\mathbf{z} = \lambda\, \mathbf{H}\,\mathbf{z}$ and renormalizing it to find $\mathbf{w}, b$.

Although this method is simple, it implements an algebraic notion of distance that is typically far from the Euclidean one, it is not robust to outliers because it is based on minimizing sums of squared distances, and it is computationally expensive because it must solve a large generalized eigensystem. It worked well in the face and person detection experiments but not in the PASCAL VOC ones, presumably because these have more outliers and poorer conditioning owing to their high intra-class variability and inter-root competition for examples.

QP Method: The second method adopts a margin-based cost function that requires the positives to lie within $\pm\Delta$ (*i.e.* Euclidean distance $\pm\Delta/\|\mathbf{w}\|$) of the hyperplane and the negatives to lie at least $1+\Delta$ to the left of it (negative samples are separated from the positives by a margin of $1/\|\mathbf{w}\|$), formulating this as an SVM-like program that can be solved using any Quadratic Program solver:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}\geq 0} \ & \frac{1}{2}||\mathbf{w}||^2 + C_+ \sum_i (\xi_i + \xi_i^*) + C_- \sum_j \xi_j \\ \text{s.t.} \ & \mathbf{w}^\top \mathbf{x}_i + b \leq \Delta + \xi_i, \\ & \mathbf{w}^\top \mathbf{x}_i + b \geq -\Delta - \xi_i^*, \ \ i \in I_+, \\ & \mathbf{w}^\top \mathbf{x}_j + b \geq \Delta + 1 - \xi_j, \ \ j \in I_-. \end{aligned} \tag{5}$$

Here, $C_\pm$ are user-defined weightings on errors in positive and negative training examples, $I_\pm$ are sets that index these examples, and Δ is a parameter that defines the assumed ratio between the width of the positive class and the width of the positive-to-negative margin.[4] By shifting b by proper amount, it is easy to see that this model can also be viewed as linear SVM with an additional penalty on positives that have over-large values of $\mathbf{w}^\top \mathbf{x} + b$. Also note that although this is a one-class method in the sense that it treats the classes asymmetrically and encourages only the positive one to be compact, negative training examples are essential here:

[2] http://cmp.felk.cvut.cz/~xfrancv/ocas/html/index.html.

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] It would be possible to learn Δ by including a (weight) $\cdot\ \Delta$ term in the cost function but we have not done this here owing to a limitation of the QP solver that we used. Instead we set Δ directly using cross validation. (Cross validation might be needed in any case, to set the weight).

without examples to enforce the ± 1 positive-to-negative margin, $\mathbf{w}$ would simply shrink to zero. (The most obvious negative-free formulation—robust L1 or SVM-regression-loss hyperplane fitting with a $\|\mathbf{w}\| = 1$ constraint – is non-convex). Overall, this method is restricted to problems in which the hyperplane can be placed so that most of the negatives lie to one side of it, but when this occurs it tends to be significantly more robust than (3).

Other Approaches. A related formulation was used for 'open set recognition' problems in which there are samples belonging to unknown classes during testing (Scheirer et al. 2013). As here, the positives are constrained to lie between two parallel hyperplanes. Their offsets are obtained using validation data but their orientation is only fixed suboptimally, using the hyperplane returned by either a 1-class SVM (Schölkopf et al. 2001) or a classical binary one.

One could also intersect half-spaces instead of slabs, thus bounding the positives to a polyhedral region. There are several algorithms for constructing such polyhedra (Murat Dundar et al. 2008; Gasimov and Ozturk 2006; Tenmoto et al. 1998; Murty et al. 1994) but most of them do not scale well with training set size and some need ancillary clusterings or labellings. In any case, enforcing a well-chosen upper bound on $\mathbf{w}^\top \mathbf{x} + b$ in addition to the lower one has a negligible computational cost and it can only improve the results, particularly as its tighter pruning may allow more scope for optimizing the hyperplane orientation in both the current stage and subsequent ones.

2.3 Stage 3: Linear Hypersphere Classifier

The third stage of the cascade applies a single linear SVDD classifier (Tax and Duin 2004). SVDD is a one class approach that finds a hypersphere $\|\mathbf{x}-\mathbf{c}\| \leq r$ that bounds most of the positives and excludes most of the negatives. Here, $\mathbf{c}$ is the sphere's center and r is its radius. The parameters are found by solving the quadratic program

$$\begin{aligned} \min_{\mathbf{c},\, r\geq 0,\, \boldsymbol{\xi}\geq 0} \quad & r^2 + \gamma_+ \sum_i \xi_i + \gamma_- \sum_j \xi_j \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 + \xi_i, \quad i \in I_+ \\ & \|\mathbf{x}_j - \mathbf{c}\|^2 \geq r^2 - \xi_j, \quad j \in I_- \end{aligned} \tag{6}$$

or its dual

$$\begin{aligned} \min_{\boldsymbol{\alpha}\geq 0} \quad & \sum_{i,j} \alpha_i\, \alpha_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{l,m} \alpha_l\, \alpha_m \left\langle \mathbf{x}_l, \mathbf{x}_m \right\rangle \\ & - 2 \sum_{l,j} \alpha_l\, \alpha_j \left\langle \mathbf{x}_l, \mathbf{x}_j \right\rangle + \sum_l \alpha_l \|\mathbf{x}_l\|^2 - \sum_i \alpha_i \|\mathbf{x}_i\|^2 \\ \text{s.t.} \quad & \sum_i \alpha_i - \sum_l \alpha_l = 1,\ \alpha_i \leq \gamma_+,\ \alpha_l \leq \gamma_- \\ & i, j \in I_+,\ \ l, m \in I_-. \end{aligned} \tag{7}$$

Here, $\langle - \rangle$ represents the (possibly kernelized) inner product, the α_i are Lagrange multipliers, and the $\gamma_\pm \in [1/n_\pm, 1]$ are ceiling parameters that can be set to values less than one to reduce the influence of outliers. The objective is strictly convex so a unique global minimum exists. The solution depends only on the active support vectors (the training examples lying exactly on the hypersphere), which makes evaluating the model more efficient in the kernelized case. One can also enforce a nonzero margin between the positives and the negatives. An alternative to (6) is to append an additional component $\|\mathbf{x} - \mathbf{c}'\|^2$ to the feature vector for some $\mathbf{c}'$ and train a linear SVM: the resulting model implicitly encodes hypersphere constraints similar to (6) so long as $\mathbf{c}'$ is not too far from the final $\mathbf{c}$ and the new feature is scaled so that the corresponding coordinate of $\mathbf{w}$ does not perturb $\|\mathbf{w}\|^2$ too much.

Large-scale problems of the form (7) can be solved with Sequential Minimal Optimization (SMO) (Platt 1998) using only the Hessian of the currently-active set of examples at each iteration. We revised the CMP quadratic programming software[5] for this, allowing us to solve problems with millions of variables in a reasonable time. Given the optimal α, the sphere center is $\mathbf{c} = \sum_i \alpha_i\, \mathbf{x}_i - \sum_j \alpha_j\, \mathbf{x}_j$, from which the radius r can be found using any active support vector.

In practice we find that the hypersphere stage complements the preceding ones well, rejecting most of the surviving false positives. Including the negative examples significantly improves the performance of (6) (particularly when the positive training set is small, *c.f.* the person detector below) but it is also costly in training time. For this reason we include only the positives during the initial rounds of latent training described in Sect. 2.6.

2.4 Stage 4: Kernelized Hypersphere Classifier

The final stage of our cascade is a single kernelized hypersphere classifier. A kernelized SVM can also be used if that works better. Kernelization supplies nonlinear classifiers that allow finer discrimination than the preceding linear stages in return for increased computation for the few examples that reach this stage. A kernel is a function that encapsulates the result of mapping points to some hidden internal feature space and evaluating the inner product there, $k(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle$, where $\phi()$ is the implicit mapping. To kernelize (7) we simply replace the inner products $\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$ with kernel evaluations $k(\mathbf{x}_i, \mathbf{x}_j)$. Training remains straightforward, but evaluating the distance from an incoming sample $\mathbf{x}$ to the center of the bounding hypersphere requires kernel evaluations $k(\mathbf{x}_i, \mathbf{x})$ against all of the support vectors $\mathbf{x}_i$. In practice only a small fraction of the training examples turn out to be support vectors but there can still be a considerable num-

[5] http://cmp.felk.cvut.cz.

ber of these, making kernel SVDD classifiers significantly more expensive than their linear counterparts. However in our applications they at typically least an order of magnitude faster than the corresponding kernel SVM. Kernel SVM's seem to require large numbers of negative support vectors to deal with their hard negatives whereas kernel SVDD's need comparatively few owing to their stronger reliance on positive modeling—it is harder for a negative to lie within the well-bounded acceptance region defined by a hypersphere than it is for it to lie within an unbounded half-space. We believe that this makes kernel SVDD a better default choice than kernel SVM for terminal stages of detection cascades.

2.5 Speed Improvement Using Reduced Set Methods

The speed of the kernelized final-stage classifier can be improved by reducing its set of support vectors. There are many methods for doing this for kernelized SVM and PCA (Schölkopf et al. 1999; Mika et al. 1999; Burges 1996). Here we use a Reduced Set method based on the simple iterative subspace estimation algorithm of Mika et al. (1999).

Running a kernelized classifier essentially reduces to evaluating sums of the form $\sum_{i=1}^{n_s} \alpha_i\, k(\mathbf{x}_i, \mathbf{x})$ for given sets of weights $\{\alpha_i\}$ and support examples $\{\mathbf{x}_i\}$. This corresponds to a dot product $\langle \Psi, \phi(\mathbf{x}) \rangle$ in the implicit feature space, where $\Psi = \sum_{i=1}^{n_s} \alpha_i\, \phi(\mathbf{x}_i)$. (In SVDD Ψ encodes the hypersphere center, in SVM the hyperplane normal). Reduced Set methods attempt to find a smaller set of examples $\mathbf{z}_i$ and weights β_i that approximates Ψ well, *i.e.*

$$\sum_{i=1}^{n_s} \alpha_i\, \phi(\mathbf{x}_i) \approx \sum_{i=1}^{n_z} \beta_i\, \phi(\mathbf{z}_i), \tag{8}$$

where $n_z \ll n_s$. Given any $\{\mathbf{z}_i\}$, their optimal reduced weights $\{\beta_i\}$ can be found by least squares regression, giving

$$\boldsymbol{\beta} = (\mathbf{K}^{zz})^{-1}\, \mathbf{K}^{zx}\, \boldsymbol{\alpha}, \tag{9}$$

where $\mathbf{K}^{zz}$ and $\mathbf{K}^{zx}$ are respectively the matrices with entries $k(\mathbf{z}_i, \mathbf{z}_j)$ and $k(\mathbf{z}_i, \mathbf{x}_j)$. Finding an optimal set of $\{\mathbf{z}_i\}$ is hard in general, but for Gaussian kernels and positive $\{\alpha_i\}$ the best single $\mathbf{z}$ vector is the global maximum of $\sum_i \alpha_i\, k(\mathbf{x}_i, \mathbf{z})$. It is also useful to place $\mathbf{z}$'s near the other local maxima of this, and such maxima can be found by simple mean shift hill climbing:

$$\mathbf{z}_{t+1} = \frac{\sum_i w_i(\mathbf{z}_t)\, \mathbf{x}_i}{\sum_i w_i(\mathbf{z}_t)} \quad \text{with } w_i(\mathbf{z}) = \alpha_i\, \exp\left(-\tfrac{||\mathbf{z}-\mathbf{x}_i||^2}{\gamma}\right). \tag{10}$$

Algorithm 1 Algorithm for Finding a Reduced Set of Gaussian Support Vectors

Input:
$\mathbf{X} = [\mathbf{x}_1 ... \mathbf{x}_{n_s}]$: initial set of support vectors
$\boldsymbol{\alpha} = [\alpha_1 ... \alpha_{n_s}]$: initial set of expansion coefficients
n_z: number of reduced support vectors
Output: $\mathbf{Z}$ and $\boldsymbol{\beta}$
Initialization:
$\mathbf{X}_0 = \mathbf{X},\ \boldsymbol{\alpha}_0 = \boldsymbol{\alpha},\ \mathbf{Z} = [\,],\ \boldsymbol{\beta} = [\,]$.
Description:
for $i = 1 : n_z$ **do**
  $\mathbf{X} = [\mathbf{X}_0,\ \mathbf{Z}],\ \boldsymbol{\alpha} = [\boldsymbol{\alpha}_0;\ -\boldsymbol{\beta}]$;
  $\mathbf{z}_0 = \text{randn}(d, 1)$;
  while $||\mathbf{z}_{t+1} - \mathbf{z}_t|| \geq \text{tol}$ **do**
    $w_{it} = \alpha_i\ \exp(-||\mathbf{z}_t - \mathbf{x}_i||^2/\gamma)$;
    $\mathbf{z}_{t+1} = (\sum_i w_{it}\, \mathbf{x}_i)/(\sum_i w_{it})$;
  end while
  $\mathbf{Z} = [\mathbf{Z},\ \mathbf{z}_{t+1}]$;
  $\boldsymbol{\beta} = (\mathbf{K}^{zz})^{-1}\, \mathbf{K}^{zx_0}\, \boldsymbol{\alpha}_0$
end for

To find $\{\mathbf{z}_i\}$ we thus proceed greedily, repeatedly finding a random local mode and subtracting out its contribution from the cost—see Algorithm 1.

2.6 Training Methodology

For the rigid (single root) object detection problems given in Sects. 3.1 and 3.2 we proceed as follows. We choose the detector window, using statistics of the training annotations to set its aspect ratio and setting its size either by finding the smallest size that provides sufficient accuracy on a cross-validation set (face detection), or by using the default size from the data set (person detection). We rescale and crop the positive training samples to the chosen window dimensions, and randomly sample windows from the negative (object-free) training images to create an initial negative training set. The visual feature vectors of these windows are used to train an initial cascade. This is scanned over the training images to collect hard examples: false positives—detections that either do not overlap an annotation or that overlap by less than 25%—and annotated objects that were missed. Hard negatives are collected from the positive images as well as the negative ones because this improves the results, especially for person detection. The hard examples are added to the initial training set, trimming it to fit into RAM by sorting the negatives by detector score and keeping those with the highest scores, then the cascade is retrained. This procedure is repeated several times. For person detection we also enlarged the positive training set by including windows from slightly perturbed annotations—this significantly improves the results.

The PASCAL VOC data set has large within-class pose and appearance variations and also significant inter-annotator variability. To handle this we used multiple root detectors in

Algorithm 2 Latent Training for Multi-Root Classifier Cascade

Input:
A set of positive and negative training images annotated with object bounding boxes, the number of roots k, the number of iterations of latent search N_{latent}, the number of iterations of hard negative search N_{hardneg}, the maximum number of negative samples to use M.

Phase I - Initialization:
(i) Sort the positives by bounding box aspect ratio and divide the list into k equal groups.
(ii) For each group $m = 1, ..., k$:

1. Determine its detection window dimensions from its bounding box statistics.
2. Rescale and crop the positive examples to the window size. Compute feature vectors for these and for their left-right reflections. Cluster the vectors to initialize a mirror-symmetric pair of root components.
3. Create a negative training set by randomly sampling windows from the negative images.
4. Train the linear stages of the cascade using these training sets.

Phase II - First Latent Layer:
For $i = 1, ..., N_{\text{latent}}$:

1. Using the current linear cascade, re-estimate the position of each positive example by searching all locations whose bounding boxes overlap its annotated box by at least 70%.
2. For $j = 1, ..., N_{\text{hardneg}}$:
   (a) Randomly sample a subset of negative images and scan them. Retain the hardest M negatives (the negative windows with the highest scores).
   (b) Re-train the linear cascade.

Phase III - Second Latent Layer (Merging Linear and Kernelized Classifiers):

1. Re-estimate the positive locations as in Phase II. Scan all of the negative images retaining the hardest M negative windows.
2. Train the nonlinear classifier stage (kernelized hypersphere or SVM).
3. Re-estimate the positive locations and re-scan for the M hardest negatives using the full (linear and nonlinear) cascade.
4. Re-train the entire cascade using this final training set.

Phase IV (Optional) - Run Time Optimization:
If the nonlinear classifier is too slow, simplify it using the Reduced Set algorithm.

a latent training framework similar to Felzenszwalb et al. (2010b). Roots are competing sub-detectors that capture different possible pose or appearance subclasses, for example front-on and side-on views of a person. The basic idea of latent training is that object instances have hidden variables that encode unknowns such as their pose or appearance subclass, the positions of their parts, or their exact locations relative to their annotation boxes. Training proceeds by hill-climbing: finding hidden variable values that maximize the score of the current detector, then re-training using these assignments. The resulting flexibility makes latently trained multi-root detectors particularly effective for highly variable data sets like PASCAL VOC.

In our case each root is a separate cascade. The roots compete as usual (the one with the highest score on an example 'owns' that example for the current training round) and they can have different window sizes and aspect ratios. The user sets the number of roots $2k$. To choose the root aspect ratios we sort the training annotations by aspect ratio, partition them into k equal groups, and find the mode of each group by histogramming. To choose the root window sizes we take the largest window that is smaller in area than 80% of the group's annotation boxes. For each group we train two roots that are left-right mirror images of one another (Hussain 2011; Felzenszwalb et al. 2010b). The initial 'left facing' versus 'right facing' partition for this is obtained using a specialized k-means algorithm that separates mirror symmetric pairs (Felzenszwalb et al. 2010b). [We have developed a more sophisticated method for this that uses convex hull classifiers instead of k-means (Cevikalp et al. 2013), but here we use k-means to facilitate comparison with (Felzenszwalb et al. 2010b)].

Latent training is used both for the root assignment decisions and to refine the locations of the training samples within their annotation boxes, thus mimicking the freedom that the final detector has when localizing detections (Hussain 2011; Felzenszwalb et al. 2010b; Zhu et al. 2010). During training, initial detectors are trained using the human-supplied annotation boxes, then in later rounds all locations near the annotation box are evaluated using the current detector and the one that gives the best score is used for retraining. Similarly, the hard negative locations are the ones found using the current detector. The complete procedure is summarized in Algorithm 2. To reduce training times, the linear hypersphere classifier (6) is trained using positives alone in Phases I and II, and using both positives and negatives in Phase III. Therefore, the training times of Phase I and Phase II stages are very similar to the DPM training time and we need an additional time to train the kernelized classifier. The time needed for training kernelized classifier will depend on the number of training samples returned by negative hard-mining and positive latent search layers.

3 Experiments

We evaluated our approach[6] with non-latent single-root detectors on the LFW, FDDB and ESOGU face datasets and on the INRIA Person dataset, and with multi-root latent detectors on the PASCAL VOC 2007 dataset. The PASCAL VOC metrics were used to assess accuracy: we report Average Precision (AP), *i.e.* area under the precision-recall curve, and we declare a detection to be a true positive if and only if its bounding box R overlaps any ground-truth annotation's box Q by more than a certain percentage, where overlap is computed as $\frac{\text{area}|Q \cap R|}{\text{area}|Q \cup R|}$. The overlap threshold was 50%, except for face detection which used 45%. (The existing face detectors that we tested disagree on how faces should be framed. We modified their outputs to resemble our annotations and to compensate for this adjustment we slightly reduced the overlap threshold required).

3.1 Face Detection

We tested our detectors on three datasets: the 13,127 image 'Labeled Faces in the Wild' (LFW) (Huang et al. 2007), the 2845 image 'Face Detection Dataset and Benchmark' (FDDB) (Jain and Learned-Miller 2010), and ESOGU Faces,[7] a new dataset that contains 667 high resolution color images with 2042 annotated frontal faces. LFW contains many faces but it is principally a face recognition dataset not a face detection one. Its images are relatively small and normalized so that most of the faces appear near the middle of the image with similar scales. This limits its value for testing multi-scale detectors. We developed the ESOGU (ESkisehir OsmanGazi University) dataset to provide more realistic testing on images from real world consumer snapshot collections. The ESOGU images contain faces appearing at a wide range of positions and scales, with complex lighting, backgrounds and occlusions. The FDDB images have a similar degree of realism. Note that we only used annotated frontal face images from these datasets in our experiments.

Training: Given the limitations of current publicly-available face detector training sets, we collected 12,500 subimages of frontal upright faces from the web for training. Most of these are from real-world images and there is a high degree of variability in appearance and lighting conditions. The faces were rescaled and cropped to a resolution of 35×28 (lower resolutions reduce the performance). For the negative set we randomly sampled 10,000 windows from face-free image regions with complex backgrounds. We used LBP + HOG for the visual features. For LBP, we divided the images into four non-overlapping quadrants and extracted descriptors from each region using circular (8,1) neighborhoods. The resulting histograms were normalized to sum to 1 and concatenated to produce the final feature vector. For HOG, we used a grid of 6×6 pixel cells with 9 bins of unsigned gradient orientations over color images, grouping each cell into overlapping 2×2 cell blocks for normalization as in Felzenszwalb et al. (2008).

We trained a four stage cascade with respectively linear SVM, linear hyperplane, linear hypersphere and kernelized hypersphere classifiers. The initial cascade was used to scan a set of thousands of images to collect additional false negatives and false positives. These hard examples were added to the training set, increasing the numbers of positives and negatives to respectively about 20 and 112 k, then the cascade was retrained. When scanning images we used detection window steps of 3 pixels horizontally and 4 vertically, and image pyramid scales spaced by 1.15. We apply non-maximum suppression by iteratively finding the top-scoring candidate detection and eliminating any others that overlap it. We also penalize candidates that have little support by suppressing groups with less than 4 overlapping candidates and for the remainder heuristically adding log(# participating candidates)/3 to the group's top score.

Results: Figure 2 shows some examples of face detections on the LFW and ESOGU test sets. Table 1 gives Average Precision scores for our 4-stage full cascade, our 3-stage linear cascade, and a simple 1-stage linear SVM on the LFW, FDDB and ESOGU test sets. It also gives the corresponding scores for three publicly-available detectors, the part-based detector of Zhu and Ramanan (2012), the boosted frontal face detector of Kalal et al. (2008), and the OpenCV Viola–Jones cascade (Viola and Jones 2004). However note that these scores are not strictly comparable because these detectors used different, non-publicly-available training sets.

Our full cascade was the best method tested on FDDB and ESOGU and the second best on LFW. Our linear cascade also performs respectably, especially on LFW. As expected, the simplistic linear SVM performs poorly, as does the aging (but efficient) Viola-Jones approach. The Zhu–Ramanan detector returns only face parts so its face regions were estimated as the tightest bounding box of the parts. This method works well with high-resolution images where the faces are typically larger than 80 × 80 pixels. In our experiments, it achieved the best result on LFW, but gave the second worst accuracy for the ESOGU faces database since there are many faces smaller than 80 × 80 pixels in this database.

Figure 3 gives Precision-Recall curves for some of the above detectors on FDDB and ESOGU. We also tested the commercial Google Picasa person tagging tool[8] informally on ESOGU, visually counting the faces returned and treating

[6] The code is available from http://mlcv.ogu.edu.tr/softwares.html.

[7] http://mlcvdb.ogu.edu.tr/facedetection.html.

[8] http://picasa.google.com.

Fig. 2 Some examples of the output of our face detection cascade on images from the Labeled Faces in the Wild (*top*) and ESOGU Faces (*bottom*) datasets. Most of the faces are correctly detected, but there are a few missed detections and false positives

Table 1 Average precision (%) for various face detectors on the LFW, FDDB and ESOGU face datasets

Method	LFW	FDDB	ESOGU faces
Full cascade	95.96	**74.10**	**87.38**
Linear cascade	94.08	69.21	79.46
Linear SVM	70.97	37.60	44.66
Zhu and Ramanan (2012)	**96.90**	73.47	50.55
Kalal et al. (2008)	87.89	66.25	79.67
Viola and Jones (2004)	80.23	67.14	73.61

Bold values show the best accuracies

every detection near a face as a true positive (even though some would not satisfy the PASCAL overlap criteria) and ignoring all of the false positives. This gave a recall rate of 91.52%, as plotted in Fig. 3.

To give an idea of the degree of pruning provided by each stage of the cascade, of the 77 M windows scanned on ESOGU, 686 k (0.9%) passed the linear SVM, 392 k (0.5%) passed the linear hyperplane classifier, 65 k (0.084%) passed the linear hypersphere classifier, and 33 k (0.043%)

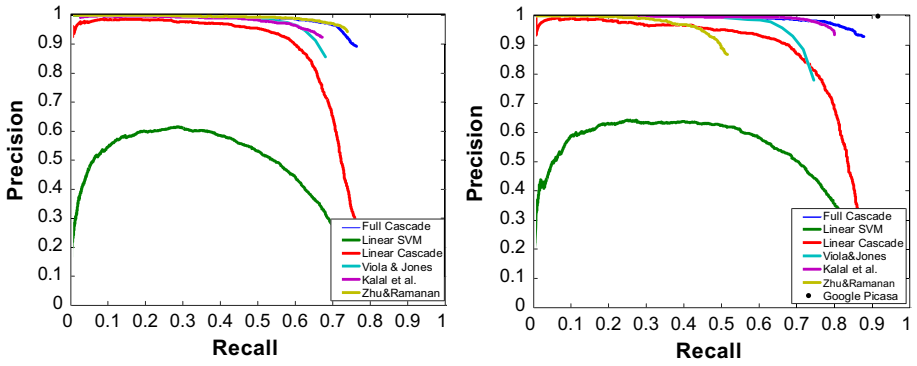

Fig. 3 Precision-Recall curves for various face detectors on FDDB (*left*) and ESOGU Faces (*right*)

passed the kernel hypersphere one. Non-maximum suppression merged these into 1887 detections (an average of 17.7 candidate windows per detection), of which 1792 (95%) were correct. 250 (12%) of the 2042 annotated faces were missed. Similarly, of the 175 M windows scanned on LFW, 2.5 M (1.4%) passed the linear SVM, 1.6 M (0.9%) passed the linear hyperplane classifier, 450 k (0.26%) passed the linear hypersphere classifier, and 312 k (0.18%) passed the kernel hypersphere one. Non-maximum suppression then merged these into 13,566 detections (22.9 candidate windows per detection) of which 13,223 (97.5%) were correct, and 517 (3.8%) of the 13,740 annotated faces were missed.

In full cascade, we used a nonlinear hypersphere classifier (kernel SVDD) at the last stage. Using a nonlinear SVM instead of nonlinear hypersphere in the cascade typically achieves slightly better results, but it is too slow compared to the cascade using nonlinear hypersphere classifier. The kernel SVDD classifier method returned 1716 support vectors whereas nonlinear SVM classifier algorithm returned 15,691 support vectors. Therefore, the cascade using a nonlinear hypersphere classifier is approximately 8 times faster than the one using a nonlinear SVM on face detection problems.

3.2 Human Detection

Training: We used the INRIA Person dataset (Dalal and Triggs 2005) for our human ('pedestrian') detection experiments, with LBP + HOG features on a grid of 8×8 pixel cells for HOG and the detection window divided into a 5×3 set of rectangular regions for LBP. We artificially enlarged the positive training set by including small random perturbations of the ground-truth annotations, and randomly sampled 12,180 negative windows from the dataset's negative (person-free) training images. Initial detectors trained on these examples were scanned over all of the training images to collect hard examples, followed by retraining. The nonlinear SVM classifier returns 28,251 support vectors on the final training set, whereas nonlinear hypersphere classifier returns only 2818 support vectors (therefore, the cascade with a nonlinear hypersphere classifier is approximately 20 times faster than the cascade using a nonlinear SVM). During detection, the search window was moved in 4 pixel steps horizontally and 6 pixel ones vertically, and pyramid scales were spaced by a factor of 1.15. Test images are up-sampled with a scale factor of 1.2 before detection.

Table 2 Average precision scores (%) for human detection on the INRIA Person dataset

Method	Average precision
Full cascade	**92.28**
Exemplar SVMs (Li et al. 2015)	44.36
Felzenszwalb et al. (2008)	90.17
Hussain and Triggs (2010)	84.10
Dalal and Triggs (2005)	75.00

Bold values show the best accuracies

Results: Table 2 gives Average Precison scores for several detectors trained and tested on the INRIA Person training and test sets. Some illustrative detections from our full cascade are shown in Fig. 4. We compared our method to those of Felzenszwalb et al. (2008, 2010b) (linear latent SVM using multiple roots and parts over HOG, supplied with the software from Felzenszwalb et al. 2010b), Ensemble of Exemplar SVMs (Malisiewicz et al. 2011; Hussain and Triggs 2010) (a two stage, linear then quadratic, cascade based on single root latent SVM over HOG + LBP + LTP), and Dalal and Triggs (2005) (simple linear SVM over HOG). Our full cascade using kernelized hypersphere classifier outperforms all of these methods, improving on the best previous AP by 2.1% despite the fact that it uses only a single root and no parts or latent training. Ensemble of Exemplar SVMs is the worst performing method although it uses an ensemble that includes 1237 linear SVM classifiers trained for each positive example.

3.3 PASCAL Visual Object Challenge Dataset

The PASCAL VOC 2007 dataset contains images of everyday scenes with annotations for all full or partial instances of 20

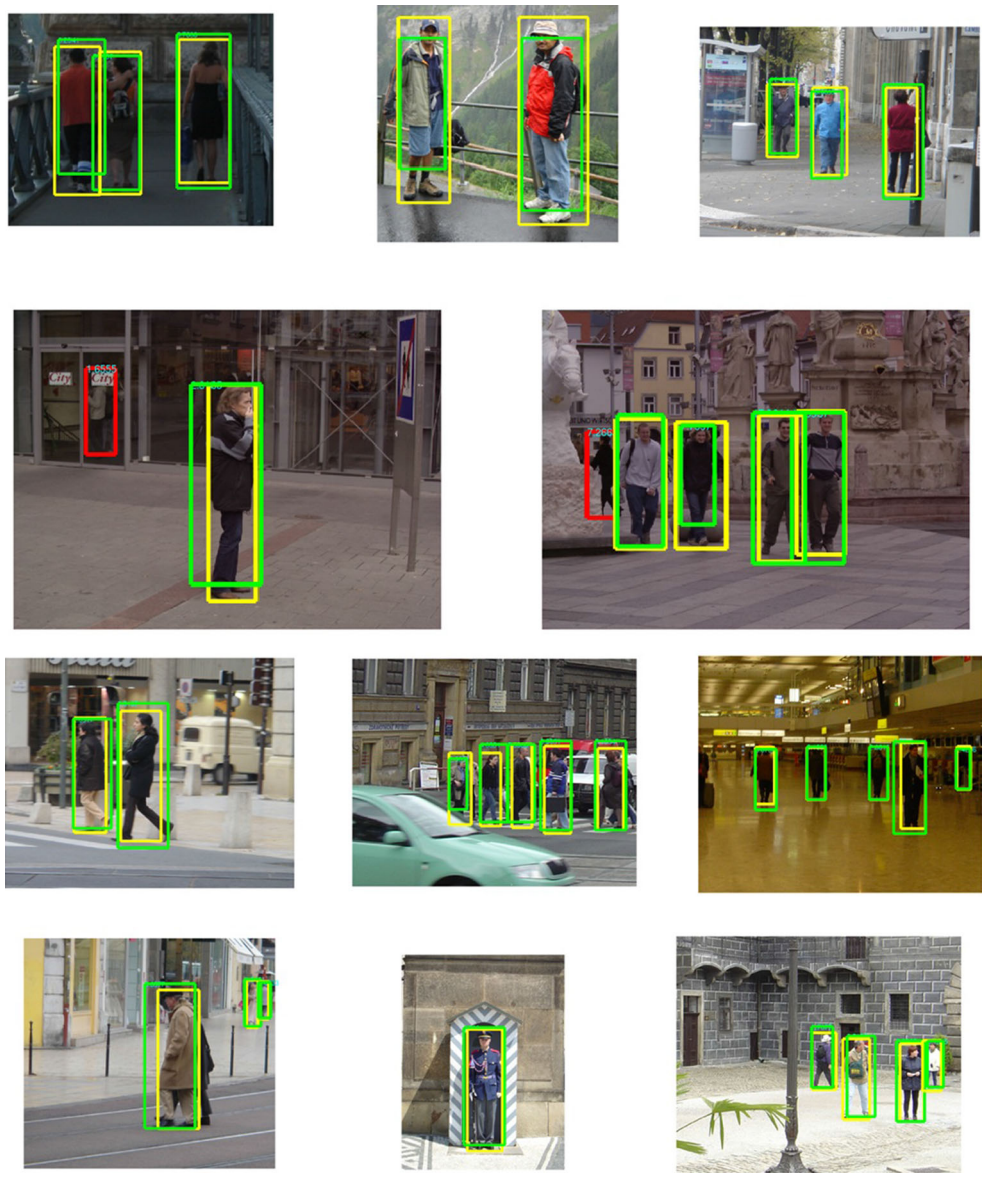

Fig. 4 Some examples of detections from our full cascade on the INRIA Person dataset. The *yellow rectangles* show ground-truth annotations, the *green* ones valid detections based on the PASCAL VOC criteria, and *red* ones false positives. (The false positives in the *second row* are actually valid detections with missing annotations) (Color figure online)

common object classes: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, and tv monitor. Its training/validation subset contains 5011 images with 12,608 annotated instances, and its test set contains 4952 images with 12,032 annotated instances.

Training: We used the latent training methodology described in Sect. 2.6 to train a full cascade detector for each VOC class. As in Felzenszwalb et al. (2010b), we use a feature pyramid with HOG features on a grid of 8×8 pixel cells and windows steps of 8 pixels. The pyramid scales were spaced by a factor of 1.07. We used kernel SVM's for the final cascade stages: they worked better than kernel SVDD owing to the modest number of positive training samples available,[9] but they tended to accumulate large numbers of negative support vectors to deal with the many hard negatives so we applied the Reduced Set algorithm to limit them to 500 support vectors per root. On a workstation, each detector took about 5 seconds to run on a VOC image. In addition, we also made comparisons to the more recent CNN based method of Girshick et al. (2014). This detector has three stages, where the first stage returns region proposals, the second stage extracts 4096 dimensional CNN features, and the last stage

[9] Typically only a few hundred—about a thousand per class partitioned among 3 pairs of roots.

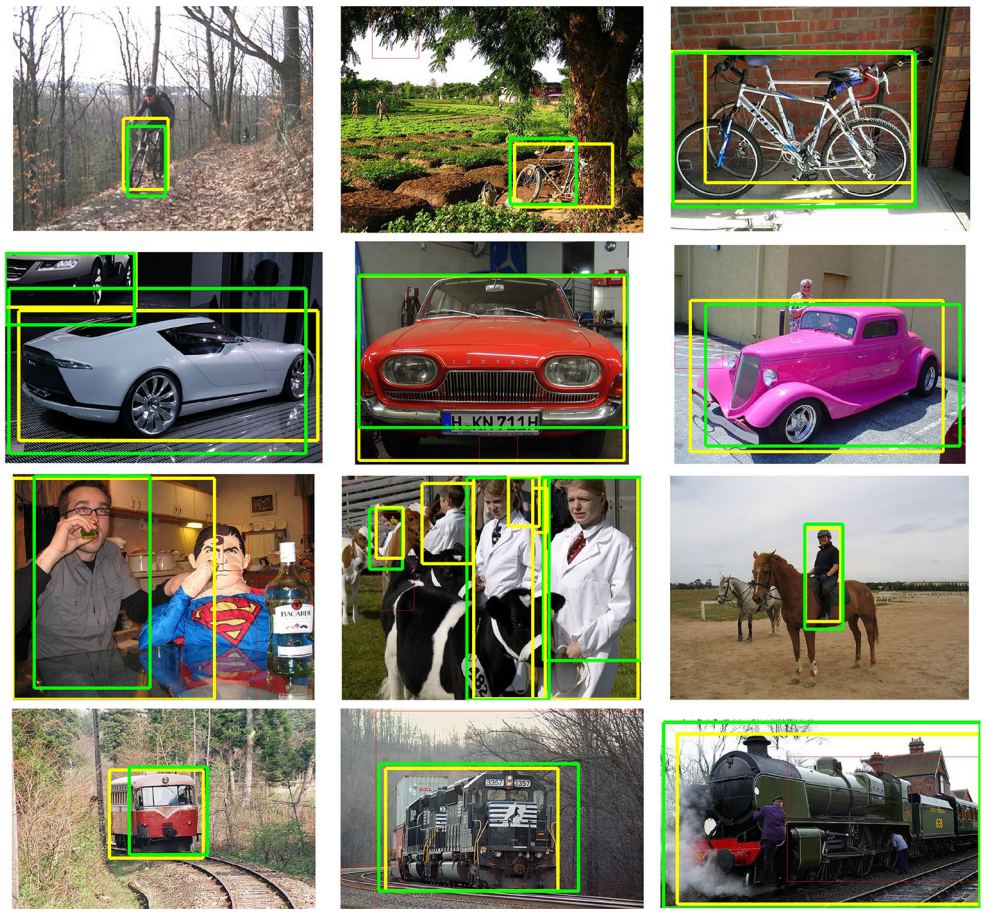

Fig. 5 Some examples of detections from our full cascade method on images from the PASCAL VOC 2007 'bicycle', 'car', 'person', and 'train' categories

includes a set of class-specic linear SVMs. Linear SVM classifiers are trained with negative hard mining. We replaced the linear SVMs with our cascade classifiers in this setting and obtained detection results on PASCAL VOC 2007. It should be noted that a single cascade classifier is trained since all candidate regions are warped onto the same size (256×256 images) before extracting CNN features.

Results: Some examples of outputs from our detectors are shown in Fig. 5. Table 3 gives Average Precision scores on the PASCAL VOC 2007 dataset for our full cascade object detectors and for several others, including the official winner of the original VOC 2007 challenge for the class[10] and a selection of methods based on the Felzenszwalb et al. (2010b) approach. 'Felzenszwalb 2R + P' denotes results from Felzenszwalb et al. (2010b) using two roots and parts. 'Felzenszwalb 6R' denotes results obtained by using the publicly-available code from Felzenszwalb et al. (2010b)[11] to train a detector with 6 roots (3 symmetric pairs) and no parts—the same configuration that our cascade uses. Similarly, 'Felzenszwalb 6R + P' denotes results obtained using 6 roots (3 pairs), each with 6 parts. We also included the accuracies of Exemplar SVMs from Malisiewicz et al. (2011). Test images are up-sampled with a scale factor of 1.2 as before. For CNN features, we used the same setting given in Girshick et al. (2014). More precisely, we apply the cascade classifier to the CNN features extracted from approximately 2000 candidate regions for each image.

As can be seen in Table 3, using CNN features significantly improves the results over HOGs. The best accuracy is obtained by the proposed cascade classifier using CNNs followed by the method of Girshick et al. (2014) using linear SVMs. Our method slightly outperforms linear SVMs on 14 categories of the 20 object categories whereas linear SVMs beat our cascade only on 5 categories and the accuracies are equivalent on 1 category. For HOG features, our full cascade outperforms the original VOC 2007 challenge winner on the class for 13 of the 20 object categories, and also outperforms the directly comparable Felzenszwalb 6R method for 13 categories. However it is the best of the methods tested only for two categories, bird and cow: on VOC,

[10] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/results/index.shtml.

[11] http://www.cs.berkeley.edu/~rbg/latent/index.html.

Table 3 Average precision score (%) on PASCAL VOC 2007

	HOGs						CNNs	
Class\method	Full cascade	VOC 2007 winner	Exemplar SVMs (Li et al. 2015)	Felzenszwalb 2R + P (Felzenszwalb et al. 2010b)	Felzenszwalb 6R	Felzenszwalb 6R + P	Full cascade	Girshick et al. (2014)
Aeroplane	23.9	26.2	20.4	**29.0**	19.1	28.5	**63.2**	63.0
Bicycle	46.4	40.9	40.7	54.0	47.5	**57.2**	68.4	**68.5**
Bird	**9.9**	9.8	9.3	0.6	2.3	3.0	**49.7**	49.6
Boat	11.6	9.4	10.0	13.4	13.3	**17.1**	40.3	**40.7**
Bottle	18.8	21.4	10.3	**26.2**	14.9	24.9	**32.5**	31.9
Bus	39.7	39.3	31.0	39.4	40.4	**47.5**	**62.9**	**62.9**
Car	46.2	43.2	40.1	46.4	41.8	**53.9**	**67.5**	67.1
Cat	11.4	**24.0**	9.6	16.1	6.7	13.6	58.5	**59.4**
Chair	17.9	12.8	10.4	16.3	16.2	**22.1**	**32.5**	32.2
Cow	**25.1**	14.0	14.7	16.5	21.0	**25.1**	**57.5**	57.2
DiningTable	17.1	9.8	2.3	**24.5**	15.3	20.4	**47.7**	47.4
Dog	7.9	**16.2**	9.7	5.0	10.6	3.9	55.4	**55.7**
Horse	40.4	33.5	38.4	43.6	40.1	**57.5**	**59.5**	58.7
Motorbike	36.5	37.5	32.0	37.8	34.1	**47.7**	**66.5**	66.1
Person	33.1	22.1	19.2	35.0	30.1	**42.4**	**50.2**	49.5
Potted plant	9.5	12.0	9.6	8.8	11.3	**12.2**	29.1	**30.1**
Sheep	13.5	**17.5**	16.7	17.3	15.0	17.4	49.7	**50.0**
Sofa	18.8	14.7	11.0	21.6	21.9	**31.5**	**48.9**	48.7
Train	36.7	33.4	29.1	34.0	34.1	**44.6**	**58.9**	58.7
TV monitor	39.4	28.9	31.5	39.0	30.0	**40.6**	**64.7**	64.3
Average	25.2	23.3	19.8	26.2	23.3	**30.6**	**53.2**	53.1

Bold values show the best accuracies

including parts greatly improves performance and as a result part-based methods using HOGs top 16 of the 20 categories here. Note that we have not included any methods that use context or inter-category non-maximum suppression for both HOGs and CNNs in these tests. Such methods would almost certainly improve the results, but they would take us outside the scope of the individual detector development considered here and they can usually be applied to any kind of detector including ours.

We find that most of our missed detections are due to the strictness of the box overlap based acceptance criteria. There are also some failures for objects that are truncated, occluded or smaller than the search window size, and occasional confusion of visually similar classes (horse *vs*. cow, bicycle *vs*. motorbike, car *vs*. bus).

4 Summary and Conclusions

This study has developed sliding window object detectors based on short cascades of binary and one-class classifiers. It began by arguing that 'one-class' methods—asymmetric classifiers that focus on tightly modeling the extent of a compact positive class, either ignoring the remaining negative examples or treating them as a broader, more diverse background—are well suited to object detection tasks and often provide improvements in accuracy and/or speed. It then developed a specific form of object detector based on a four-stage rejection cascade whose stages are respectively: a linear SVM for initial pruning of easy negatives; a series of linear distance-to-hyperplane filters; a linear interior-of-hypersphere filter; and a kernelized interior-of-hypersphere classifier. The last three stages are all one-class methods. For the second stage we developed a novel convex program based classifier that can be viewed either as linear SVM with an upper bound on positive scores or as SVM-regression hyperplane fitting while avoiding a negative class that lies off to one side. Our cascade framework supports multi-root detectors and latent training (Felzenszwalb et al. 2010b), but not (currently) parts. The final detectors give very competitive results on the LFW, FDDB and ESOGU face detection and the INRIA person detection datasets (with single roots and no latent training), and respectable results on the PASCAL VOC 2007 dataset. Linear cascades including only the first three

stages of our method also work quite well in many cases. For the final stage of the full four-stage method, a kernel SVM simplified using a Reduced Set method is sometimes a better choice than a kernel hypersphere classifier, especially when there are many roots and relatively few positive training examples. However, it should be kept in mind that the cascade using a kernelized hypersphere is much faster compared to the cascade using the kernel SVM since hypersphere classifier returns less support vectors. More precisely, the cascade using a kernel hypersphere classifier is approximately 8 times faster than the one using a kernel SVM on face detection problems whereas it was 20 times faster on people detection.

There are several avenues for improving the proposed approach. The most obvious is to add object parts, which might significantly improve the results on PASCAL VOC. However creating a cascade that uses parts is problematic, at least as they are implemented in Felzenszwalb et al. (2010b). Such parts are expensive to evaluate and hence ill-suited to rapid search pruning, yet if flexible class modeling is needed it must begin early enough in the cascade to prevent highly flexed positives from being eliminated. One of Felzenszwalb et al. (2010b)'s strengths is the robustness that it gains by postponing ancillary decisions (such as whether or not a given part occurred at a given location) until the overall detection decision. Any pruning-based framework for speeding up such computations would need good bounds on, or empirical estimates of, the extent to which a proposed filtering rule (*e.g.* a threshold on a partly evaluated part or root) could perturb the final detection decision. One-class methods might be useful for such bounds owing to their ability to single out coherent groups of responses amidst noise. Conversely, filtering is easier to incorporate into approaches that are based on finding and assembling discrete detections (as opposed to integrating soft presence scores), so an alternative would be, *e.g.* , to filter on possible root locations then run part searches near these as in Cevikalp et al. (2013), or to select parts by their intrinsic detectability (salience relative to the background) as in Ullman and Sali (2000) and use these as anchors for the search.

Another obvious enhancement would be to incorporate inter-class interactions, both elementary ones such as non-maximum-class suppression of overlapping detections from competing classes and higher-level ones such as context.

Finally, we are not yet satisfied with the distance-to-hyperplane classifiers. A better formulation is needed that allows a well-localized positive class to be isolated within a background of negatives using only one or a few dot product evaluations, while yielding a convex learning algorithm that scales to large datasets.

Acknowledgements This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant Number EEEAG-109E279.

References

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE T-PAMI*, *28*(12), 2037–2041.

Aldavert, D., Ramisa, A., Mantaras, R. L., & Toledo, R. (2010). Fast and robust object segmentation with the integral linear classifier. In *CVPR*.

Amit, Y., & Geman, D. (1999). A computational model for visual selection. *Neural Computation*, *11*, 1691–1715.

Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A., & Ferguson, D.(2015). Real-time pedestrian detection with deep network cascades. In *BMVC*.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Surf: Speeded up robust features. *CVIU*, *110*(3), 346–359.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE T-PAMI*, *24*(24), 509–521.

Benenson, R., Mathias, M., Timofte, R., & Van Gool, L. (2010). Pedestrian detection at 100 frames per second. In *CVPR*.

Burges, C. J. C. (1996). Simplified support vector decisions. In *International conference on machine learning*.

Cevikalp, H., & Triggs, B. (2008). Nearest hyperdisk methods for high-dimensional classification. In *International conference on machine learning*.

Cevikalp, H., & Triggs, B. (2012). Efficient object detection using cascades of nearest convex model classifiers. In *CVPR*.

Cevikalp, H., Triggs, B., & Franc, V. (2013). Face and landmark detection by using cascade of classifiers. In *IEEE International conference on automatic face and gesture recognition*.

Cevikalp, H., Larlus, D., Neamtu, M., Triggs, B., & Jurie, F. (2010). Manifold based local classifiers: Linear and nonlinear approaches. *Journal of Signal Processing Systems*, *61*, 61–73.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Felzenszwalb, P. F., & Girshick, R. B., & McAllester, D. (2010a). Cascade object detection with deformable part models. In *CVPR*.

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale deformable part model. In *CVPR*.

Felzenszwalb, P., Girshick, R. B., McAllester, D., & Ramanan, D. (2010b). Object detection with discriminatively trained part based models. *IEEE T-PAMI*, *32*(9), 1627–1645.

Gasimov, R. N., & Ozturk, G. (2006). Separation via polyhedral conic functions. *Optimization Methods and Software*, *21*, 527–540.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR4*.

Harzallah, H., Jurie, F., & Schmid, C.(2009). Combining efficient object localization and image classification. In *ICCV*.

Huang, G., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. Technical Report 07-49, University of Massachusetts, Amherst, October.

Hussain, S. (2011). *Machine learning methods for visual object detection*. PhD thesis, Laboratoire Jean Kuntzmann.

Hussain, S., & Triggs, B.(2010). Feature sets and dimensionality reduction for visual object detection. In *BMVC*.

Jain, V., & Learned-Miller, E., (2010). *Fddb: A benchmark for face detection in unconstrained settings*. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst.

Jin, H., Liu, Q., & Lu, H. (2004). Face detection using one-class-based support vectors. In *International conference on automatic face and gesture recognition*.

Kalal, Z., Matas, J., & Mikolajczyk, K. (2008). Weighted sampling for large-scale boosting. In *BMVC*.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*.

Levi, K., & Weiss, Y. (2004). Learning object detection from a small number of examples: The importance of good features. In *CVPR*.

Li, H., Lin, Z., Shen, X., Brandt, J., & Hua, G.(2015). A convoulutional neural network cascade for face detection. In *CVPR*.

Lowe, D. G. (2004). Distinctive image features from scale invariant keypoints. *IJCV*, *60*, 91–110.

Malisiewicz, T., Gupta, A., & Efros, A. A. (2011). Ensemble of exemplar-SVTS for object detection and beyond. In *ICCV*.

Mangasarian, O. L., & Wild, E. W. (2006). Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE T-PAMI*, *28*, 69–74.

Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., & Ratsch, G. (1999). Kernel PCA and de-noising in feature spaces. In *Neural information processing systems (NIPS)*.

Murat Dundar, M., Wolf, M., Lakare, S., Salganicoff, M., & Raykar, V. C. (2008). Polyhedral clasifier for target detection a case study: Colorectal cancer. In *International conference on machine learning*.

Murty, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, *2*, 1–32.

Orozco, J., Martinez, B., & Pantic, M. (2015). Empirical analysis of cascade deformable models for multi-view face detection. *Image and Vision Computing*, *42*, 47–61.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *IJCV*, *38*, 15–33.

Perrotton, X., Sturzel, M., & Roux, M. (2010). Implicit hierarchical boosting for multi-view object detection. In *CVPR*.

Platt, J. C., (1998). *Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods-support vector learning*. Cambridge, MA: MIT Press.

Porikli, F. (2005). Integral histogram: A fast way to extract histograms in Cartesian spaces. In *CVPR*.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE T-PAMI*, *20*, 23–38.

Scheirer, W. J., Rocha, A., Sapkota, A., & Boult, T. E. (2013). Towards open set recognition. *IEEE Transactions on PAMI*, *35*, 1757–1772.

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K. R., Ratsch, G., et al. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*, 1000–1017.

Schölkopf, B., Platt, J., Smola, A., & Williamson, R. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1471.

Shams, L., & Speslstra, J. (1996). Learning Gabor-based features for face detection. In *World congress in neural networks*.

Sizintsev, M., Derpanis, K. G., & Hogue, A. (2010). Histogram-based search: A comparative study. In *CVPR*.

Tan, X., & Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, *19*, 1635–1650.

Tax, D. M. J., & Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, *54*, 45–66.

Tenmoto, H., Kudo, M., & Shimbo, M. (1998). Piecewise linear classifiers with an appropriate number of hyperplanes. *Pattern Recognition*, *31*, 1627–1634.

Ullman, S., & Sali, E. (2000). Object classification using a fragment-based representation. In *Proceedings of the first IEEE international workshop on biologically motivated computer vision*, BMVC '00, pp. 73–87. London. Springer.

Varma, M., & Ray, D. (2007). Learning the discriminative power-invariance trade-off. In *ICCV*.

Vedaldi, A., Gulshan, V., Varma, M., & Zisserman, A. (2009). Multiple kernels for object detection. In *ICCV*.

Vedaldi, A., & Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on PAMI*, *34*, 480–492.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *IJCV*, *57*(2), 137–154.

Wang, X., Han, T. X., & Yan, S. (2009). A HOG-LBP human detector with partial occlusion handling. In *ICCV*.

Wei, Y., & Tao, L. (2010). Efficient histogram-based sliding window. In *CVPR*.

Zhu, X., & Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *CVPR*.

Zhu, L., Chen, Y., Yuille, A., & Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *CVPR*.

Zhu, X., Vondrick, C., Ramanan, D., & Fowlkes, C. C. (2012). Do we need more training data or better models for object detection. In *BMVC*.