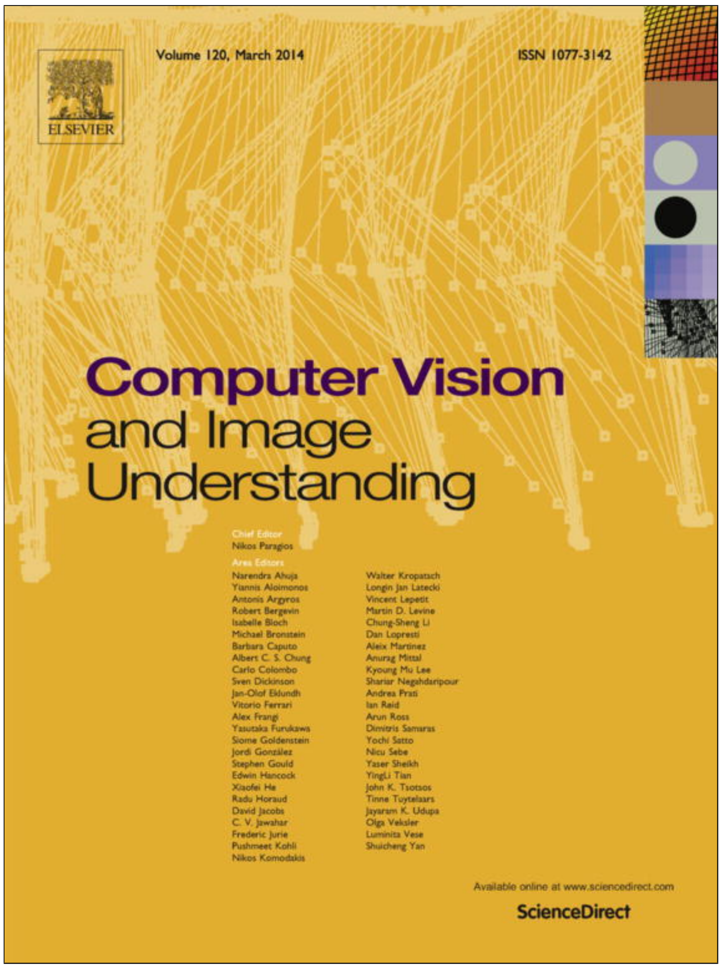

Computer Vision and Image Understanding 120 (2014) 103–116

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

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

Boosting masked dominant orientation templates for efficient object detection ☆

Reyes Rios-Cabrera [a,b,*], Tinne Tuytelaars [b]

[a] CINVESTAV, Robotics and Advanced Manufacturing, Av. Industria Metalurgica 1062, Ramos Arizpe 25900, Mexico
[b] KU Leuven, ESAT-PSI-VISICS, iMinds, Kasteelpark Arenberg 10, Leuven B-3001, Belgium

ARTICLE INFO

Article history:
Received 18 October 2012
Accepted 21 December 2013
Available online 29 December 2013

Keywords:
Binary templates
Meta-data transfer
Oriented gradients
Vehicle detection
Template-based object detection

ABSTRACT

In this paper we present a novel template-based approach for fast object detection. In particular we investigate the use of Dominant Orientation Templates (DOT), a binary template representation introduced by Hinterstoisser et al., as a means for fast detection of objects even if textureless. During training, we learn a binary mask for each template that allows to remove background clutter while at the same time including relevant context information. These mask templates then serve as weak classifiers in an Adaboost framework.

We demonstrate our method on detection of shape-oriented object classes as well as multiview vehicle detection. We obtain a fast yet highly accurate method for category level detection that compares favorably to other more complicated yet much slower approaches. We further show how to efficiently transfer meta-data using the top most similar activated templates.

Finally, we propose an optimization scheme for detection of specific objects using our proposed masks trained by the SVM, resulting in an increment of up to 17% in performance of the DOT method, without sacrificing testing speed and it is able to run the training on real time.

1. Introduction

To date, the main application of template matching in image processing has been the detection in an image of an *a priori* known object instance from an *a priori* known viewpoint. To incorporate viewpoint variations (or non-rigid deformation), usually multiple templates are stored in memory and matched to the image one by one. The maximum response is selected and if this is greater than a threshold value, then it is counted as a detection. Fig. 1 illustrates this procedure.

Thanks to its speed, template matching has traditionally been a popular method in manufacturing environments, where occlusions and clutter can mostly be avoided and lighting and pose variations can be carefully controlled, although it has also been applied in more challenging settings such as, e.g., pedestrian detection [1]. Another advantage of template matching is that it is also possible to transfer offline annotated/created meta data to the object, once it is detected.

In other contexts, objects are mostly recognized based on local features (e.g. SIFT [2]). This usually works well with changing lighting conditions, partial occlusion and, to some extent, changes in the viewpoint. However, this kind of approach usually fails on texture-less objects such as mugs or bottles that are mostly determined by their projected contours or shapes. Another method, that seems to work well in both situations, is the histogram of oriented gradients (HOG) representation [3], combined with an SVM. However, its computation time is high. Here we investigate a simpler and more efficient alternative.

Hinterstoisser et al. proposed a template representation, coined *Dominant Orientation Templates* or DOT [4]. DOT is inspired by the Histograms of Oriented Gradients [3] descriptor, but designed to be fast. They show impressive results, recognizing 3D non-textured objects over cluttered background in realtime, using several templates per object. At the core of their method is a binary representation of the image and template, based on dominant gradient orientations. Evaluating the template for a given location in a new image is highly accurate and requires only simple bit-wise operations, making it very fast. By jittering the training image when building the template, they ensure invariance to small translations. This not only increases the robustness, but also allows skipping many locations while parsing an image. However, there are also some shortcomings. One of them is that the method cannot model negative data during training, which during detection, results in a relatively high number of false positives. Another limitation is that it cannot handle objects at the category level,

☆ This paper has been recommended for acceptance by J.-O. Eklundh.
* Corresponding author at: CINVESTAV, Robotics and Advanced Manufacturing. Av. Industria Metalurgica 1062, Ramos Arizpe 25900, Mexico.
E-mail addresses: reyes.rios@cinvestav.edu.mx (R. Rios-Cabrera), tinne.tuytelaars@esat.kuleuven.be (T. Tuytelaars).

1077-3142/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2013.12.008

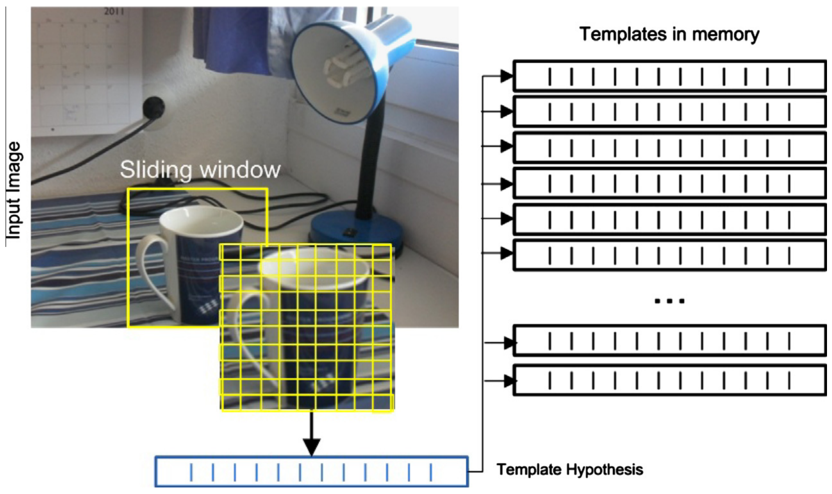

Fig. 1. Template matching. A sliding window is run over the image obtaining a template representation for each position of the window. Then it is evaluated against each of the templates stored in memory.

but only *a priori* known instances of the *same* object. Another more robust method was introduced recently by the same authors, called LINE2D [5]. It improves over DOT, but suffers from the same limitations.

In this paper, we investigate whether binary Dominant Orientation Templates (DOT) can also be used successfully for learning to detect instances of an object *class*, without sacrificing efficiency. This requires a method that can generalize beyond the examples seen during training, so as to also recognize previously unseen instances of the object class. Given the within-class variability, a single template per viewpoint is no longer sufficient, and the greedy selection of templates as in [4] is suboptimal (see Section 5.2). Increasing the amount of jittering would allow for more variability, but also reduces the discriminative power leading to more false positives. Instead, starting from multiple training images possibly covering different object poses, we need a method for template selection, as well as a scheme for combining the outputs of the different templates, that effectively allows to generalize beyond the individual examples (templates). To this end, we propose to learn masks for the DOT templates and to use these as weak classifiers in an Adaboost framework [6]. We also show, that these masks can be applied to template based methods for specific object detection, showing up to 17% increment in performance.

1.1. Main contribution

Our main contribution is the construction of an efficient, category-level object detector based on a computationally efficient, yet discriminative template representation. We propose to learn a binary mask for each template, based on feature selection using a linear support vector machine. We also show how the masks can be used for meta data transfer.

By learning the masks we refine the bounding box annotation to a more representative template, including relevant context cues (e.g. the shadow beneath a car) while at the same time removing background clutter (e.g. the buildings behind the car) or areas on the object itself that do not generalize well (e.g. the prints on mugs). This turns out to be a crucial step for obtaining a highly accurate system. We show that, by learning appropriate binary masks and using the masked templates as weak classifiers in a boosting framework, the output of several templates can be combined into a powerful detector. Moreover, it is inherently fast thanks to the use of bitwise operations only.

Additionally, we propose a method based on the masks, to optimize template tables for specific object detection, incrementing the performance without sacrificing detection speed, at the cost of a very small training time (ms). The same scheme can be applied to other binary based template methods such as LINE2D or LINE-MOD [5] to improve their performance.

1.2. Efficient template based object detection

While the Adaboost framework is a standard classifier in vision applications, it has mostly been applied in combination with localized weak classifiers (e.g. [7,8]).

Here we show it can also be applied successfully in combination with global features (templates). This results in a strong classifier that combines multiple DOT templates at classification time. Note how this is significantly different from applying several templates one by one as in [4] or using a mixture model with latent SVM [9]: We do not pick (implicitly or explicitly) one template for a given test image, but instead always evaluate multiple templates, combining their scores with the weights determined by Adaboost. We believe that this is essential to make the scheme generalize well to previously unseen object instances.

Initially, the mask learning scheme we propose seemed to be limited to the case of well aligned training data (e.g. a single viewpoint). By unsupervised clustering of the training data, we show it can also deal with multiview settings. Here it is important to stress that this clustering is only used for learning the masks. Unlike e.g. [10] we do not learn separate viewpoint specific classifiers, but instead run a single detector that has learnt to deal with the viewpoint changes.

The remainder of the paper is structured as follows: We first discuss related work in Section 2. Then we briefly explain the dominant orientation templates [4] in Section 3. In Section 4, we show how to adapt DOT to work as features (weak classifiers) and present our method for object class learning, including the Adaboost template selection and weighting, as well as the learning of the masks. Section 5 describes our experimental results, Section 6 presents discussion and future work and Section 7 concludes the paper.

2. Related work

Here we focus on the most relevant work on shape oriented methods, efficient multiview, template-based as well as boosting-based methods. Examples of *template-based* category-level object detection methods are rare. Probably the best known example is the method proposed by Gavrilla and Philomin [1], which was applied to pedestrian detection. Their method uses a template hierarchy to capture the variability of object shapes using Chamfer matching. It is a fast method (using Distance Transform on a binary edge image), but a disadvantage of the Chamfer transform is its sensitivity to outliers which often result from occlusions.

In [11], Feng and Hai presented a multiscale template matching system. The templates are based on groups of Haar-like binary features to speed up the matching process and to handle re-scaling. However, Haar-like features are less discriminative than gradients. Recently, Danhan et al. [12] combined dominant orientation templates with a patch based Random Forest to detect pedestrians, showing an accurate 5 fps detector.

To deal with *multiple viewpoints*, a common approach is to use multiple detectors in parallel, each focusing on a particular object pose (e.g. side views of cars, frontal cars, etc.). Various more integrated multiview detectors have been proposed as well (e.g. [13]). These are often combined with simultaneous pose estimation. However, they rarely focus on efficiency. Variations of the standard boosting cascade to better deal with viewpoint changes include the work of Perrotton et al. [14]. They propose a multiview object detection method consisting of a single cascade with implicit hierarchy. They argue that in an implicit multiview system, selected features should focus only on a subset of the training data, while rejecting all negatives. This means that a feature should not be penalized if it does not generalize for all views. We explore this idea, and train the features to only focus on a subset of positive training samples (a cluster).

In [15], Kuo and Nevatia proposed a method for multiview car detection. They start from unsupervised categorization, and then train a boosting-based tree-structured detector, with separate branches for the different clusters of positive and negative images. In our work, we successfully integrate all the viewpoints in a single strong classifier, that is further speeded up by training a cascade structure.

AdaBoost has mostly been used to select and combine a set of local features such as Haar features [7] from a large pool. In [16], Zhang and Viola introduced Multiple Instance Pruning (MIP). This cascade is easily trained turning a single bootstrapped strong classifier into an efficient cascade structure. They argue that the trained cascade reaches the same accuracy as the single strong classifier. We used this cascade structure in our experiments.

Quite recently, Bangpeng et al. [17] used local DOT template matching for fine-grained image categorization. The objective was to classify objects that belong to the same basic-level class, and share similar shape or visual appearance. However, they did not focus on efficiency.

Examples where AdaBoost is used to select and combine global features, as in our case, are rare. In [18], Zhu et al. used AdaBoost in combination with HOG features. However, the features are again mostly local (just a single block from the original HOG descriptor), although the size of the cell/region is relaxed and therefore sometimes also larger blocks are selected. Similar features have been used in [15]. Also [19] uses somewhat larger features, albeit again not global. Using local features has the advantage that it is robust to image clutter and changing background, thus avoiding the need for a mask learning preprocessing step. However, for object classes with limited texture, local features may not be as discriminative as global ones.

When using Adaboost with local features, robustness to small misalignments is often a built-in property of the local features (e.g. by integrating information over a small neighborhood), while the combination of weak classifiers is mostly responsible for capturing the overall spatial configuration. This scheme is known to be sensitive to misalignments and therefore usually requires well-aligned input (e.g. only frontal faces). When using Adaboost with global features, on the other hand, the spatial configuration is encoded directly in the features themselves, while the combination of weak classifiers and re-weighting of training samples is responsible for adding robustness to misalignments. This is an important shift in paradigm. Surprisingly, we show that Adaboost in combination with global features can cope with different viewpoints within a single strong classifier.

Some methods focusing especially on *shape-dominated object classes* have been proposed as well. In [20,21] Ferrari et al. train models that are able to detect more accurately the object borders. In [22] Ommer et al. propose a multiscale method to detect shape based objects. In [23], Maji and Malik present a discriminative Hough transform based object detector, where each local part casts a weighted vote for the possible locations of the object center, and in [24] Fritz and Schiele propose a method for the discovery and detection of object classes based on decompositions using topic models. Our template-based approach also seems most suited for shape-dominated object classes, and since it is based on DOT, it can handle textureless objects. Contrary to the other shape based methods described above, our implementation is designed to work in real time.

In [25], Malisiewicz et al. propose an exemplar-SVM based object detector. Each exemplar is defined by a single positive sample and millions of negatives. The proposal shows surprisingly good results on PASCAL VOC. However, it is very slow. They also show how to achieve meta-data transfer, using the close relation of the trained exemplar and the detected object. In our method, we apply a similar idea to retrieve annotated data from training samples.

In the context of efficient detectors, Dollar et al. propose a fast pedestrian detector [8]. They introduce the use of integral channels and efficiently approximate scales of features, such that the image needs to be resized only a few times. More recently, Benenson et al. [26] presented a new pedestrian detector that further improves in speed and quality over the state-of-the-art, by efficiently handling different scales and transferring computation from test time to training time. In monocular images the system can work at 50 fps. They reach that speed, exploiting geometric context extracted from the images, image constraints and the processing power of a GPU. However, both these methods require textured objects in order to work properly, which e.g. in the context of robots trying to grab something, may fail, due to the texture-less nature of some objects. Moreover, their ideas are complementary to our work, since we focus on the basic image representation as in the case of Danhang et al. [12]. We expect that ideas from [26] can be integrated into ours to further improve speed.

3. Background: dominant orientation templates

DOT [4] is related to the Histogram of Oriented Gradients or HOG representation of [3]. As in HOG, the area-of-interest is divided into a fixed number of regions or blocks. However, instead of creating an orientation histogram for each cell/region, DOT only keeps the strongest dominant orientations. We use color images, and compute the gradients for each channel independently. Then we select the channel whose gradient magnitude is largest and compute its orientation as in [5]:

$$I_G(x) = ori\left(\frac{\partial C^*}{\partial x}\right) \qquad \text{with } C^* = \underset{C\in\{R,G,B\}}{\arg\max}\left\|\frac{\partial C}{\partial x}\right\| \tag{1}$$

These orientations are coded in a binary fashion, with one bit for each of the n_0 different orientations considered ($n_0 = 7$) and using steps of $180/n_0$ degrees (the sign is not taken into account). The presence or lack of a specific orientation (after thresholding its magnitude) then results in a 1 or 0 for one of the first n_0 bits in the byte corresponding to this region. The last bit corresponds to the lack of any gradients, i.e. a homogeneous patch. While learning a template, jittering is applied around each region over a range $[-\frac{T}{2}, +\frac{T}{2}] \times [-\frac{T}{2}, +\frac{T}{2}]$ where T is the size of the region in pixels. By jittering the templates, robustness to misalignments and small deformations is achieved. However, at test time, no jittering is applied, and only the single strongest gradient for each region is stored. Using only the strongest gradients further speeds up the detection process.

Fig. 2 shows a learnt template $\mathcal{L}(\mathcal{O},\mathcal{R})$(top right) being compared to a feature vector $do(\mathcal{I}, c+\mathcal{R})$ (bottom right) extracted from an image at test time (left). $\mathcal{L}(\mathcal{O},\mathcal{R})$ represents a list of dominant orientations of region $\mathcal{R}$ in the reference image $\mathcal{O}$ and $do(\mathcal{I}, c+\mathcal{R})$ is the template hypothesis of image I located at c. The learnt template saves up to n_0 orientations, but the test vector only keeps the single most dominant orientation for each region, as seen in the figure. Comparison is then performed by simply AND-ing the two binary vectors, and counting the number of ones in the final vector (or, actually, looking up the sum in a lookup table). A similar idea was used in [27]. Using binary operations only, and further speeding up with SSE operations, a very efficient detection system is achieved. LINE2D or LINE-MOD [5], work also on bitwise operations, and our proposal can be applied to them as well.

4. Our method

Here, we want to apply template matching based on DOT to the problem of (multiview) category-level object detection. Given a set of training images (both positive as well as negative examples), we need to construct a classifier that can be applied to a new test image. While a single template extracted from a random training image works fine in the case of specific object detection, as shown in [4,5], it does not have sufficient generalization capabilities to cope with the intra-class variability (especially variations in shape). Moreover, we would like to train, in a discriminative setting, what modes of variability are allowed for the given class and what modes are not. We claim that this can be achieved by combining the output of multiple templates obtained from multiple training images. Here, we show that Adaboost is well suited to solve these issues.

4.1. Converting templates to weak classifiers

First, we need to convert the binary templates into weak classifiers. We refer to these as *DOT Classifiers*. For this we use the similarity measure of [4] (following their notation):

$$\varepsilon(I,\mathcal{O},c) = \sum_{\mathcal{R}\ \text{in}\ \mathcal{O}} \delta(do(\mathcal{I}, c+\mathcal{R}) \in \mathcal{L}(\mathcal{O},\mathcal{R})) \tag{2}$$

where $\delta(do(\mathcal{I}, c+\mathcal{R}) \in \mathcal{L}(\mathcal{O},\mathcal{R})) = 1$ iff $L \otimes D \neq 0$ with $\otimes$ the bitwise AND operator and L, D representing the bytes corresponding to region $\mathcal{R}$ in the template and hypothesis region respectively, and $do(\mathcal{I}, c+\mathcal{R})$, $\mathcal{L}(\mathcal{O},\mathcal{R})$ as defined in Section 3. In words, $\varepsilon(I,\mathcal{O},c)$ counts for how many regions the image dominant gradient is in agreement with one of the gradients in the corresponding region of the template. To adapt it to the Adaboost framework we define:

$$\top(I) = \begin{cases} -1 & \textit{if}\ \varepsilon(I,\mathcal{O},c) \leqslant \tau \\ +1 & \textit{otherwise} \end{cases} \tag{3}$$

where $\top$ is a binary response (weak classifier) used to build a strong classifier $H(I)$:

$$H(I) = sign\left(\sum_{t=1}^{T_0} \alpha_t \top_t(I)\right) \tag{4}$$

$H(I)$ builds on T_0 templates selected by Adaboost. We set the weights α and thresholds τ (in the definition of $\top_t(I)$) automatically for each template using the standard Adaboost procedure. Note that this is different from the DOT baseline [4], where these thresholds are defined manually, which may result in a suboptimal performance.

4.2. Boosting templates

For each positive training sample, we create a template by fixing a grid on the image sample as shown in Fig. 2, and computing the $n_0 = 7$ dominant orientations for each region. The orientations with magnitude smaller than a threshold are ignored, as in the baseline [4]. Likewise, we use only the orientation without the magnitude. Later, we learn a discriminative binary mask for each template: In the case of single view, we use the Nearest Neighbors (with similarity of about 90%) to train an SVM as described in following sections. In case of multiview, we use clusters.

Then we generate negative templates from negative data. Thus we have as weak classifiers a set of 'positive' templates created based on the positive data and a set of 'negative' templates created based on the negative data. We use the positive training samples, and a bigger amount of negative samples for training the strong classifier with discrete Adaboost. Note the subtle but important difference between 'templates' and 'training samples': the templates have up to n_0 orientations in each byte, whereas training samples have only 1 (the strongest). Moreover, jittering is also applied only for constructing the templates as described in Section 3. This asymmetry is used in the DOT framework to incorporate robustness and to speed up the detection process. After training an initial detector, we apply bootstrapping to reduce the number of false positives further. This has an enormous positive effect on the accuracy of the final classifier.

In order to keep high efficiency using binary operations, we use also SSE operations. We based our implementation on the publicly available code from [4].

4.3. Learning discriminative masks

The method as described above directly uses the DOT representation extracted from the training images. This may not be optimal, since the provided bounding boxes always include some background. For the DOT classifiers, working with binary templates, each non-zero entry in a template has the same weight. And since we work with global features, the weights determined by Adaboost can only reweight different templates with respect to each other,

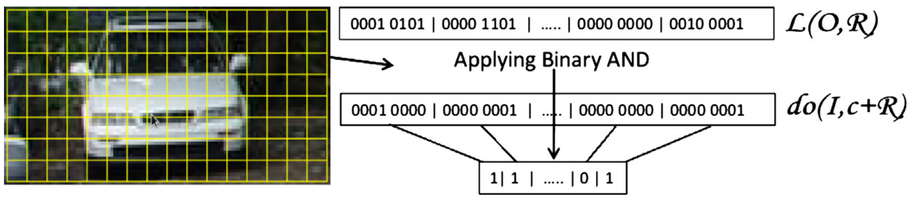

Fig. 2. Computing a template similarity. We use a lookup table of pre-computed responses to determine similarity using bitwise operations.

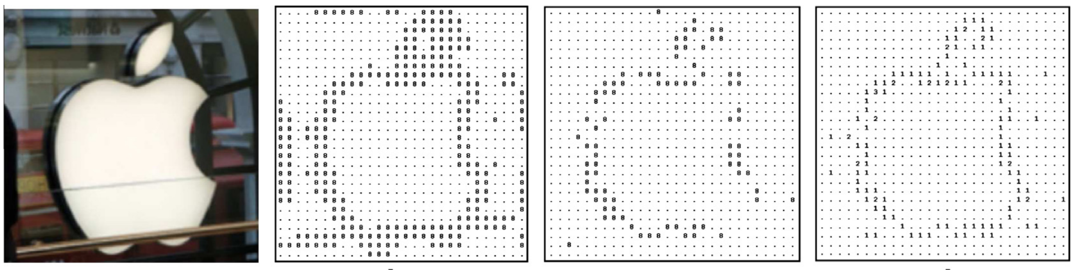

Fig. 3. (a) An example of a positive training image. (b) A template using 256 active regions with $T = 5$ pixels. (c) The mask we learnt using a linear SVM."8" = Dominant Region, $C_j > 0$, see Eq. (6). (d) It is the bit level mask, here the number indicates the amount of different strong orientations found in the region. It shows the top 128, based on the weights learnt by the SVM.

keeping the relative weights of different elements within a single DOT template the same.[1] Therefore, our framework so far cannot learn to ignore regions that correspond to irrelevant gradients.

Using a mask instead of a bounding box can reduce the negative effect of gradients that are not on the object, but manually generating such masks for each positive training image is labor-intensive. Moreover, not all gradients outside the object are irrelevant. As shown in [3,28], context information can significantly improve detection accuracy. Likewise, not all gradients within the object are equally relevant (e.g. some may be less stable than others). Therefore, we propose to *learn* a mask for each positive template prior to applying Adaboost. Any feature selection algorithm that can cope with binary input can be used for this step. Here, we use the weights of a linear SVM for this purpose. We emphasize the fact that once we have learnt a mask, we can directly AND the mask with the template, as a precomputing step before learning the strong classifier. Hence applying the mask does not take extra time during testing. In [29] it was shown experimentally that feature selection using weights from linear SVMs yields better classification performance than other feature weighting methods. A Support Vector Machine trains a linear classifier of the form $sgn(w^Tx + b)$. Learning is addressed as an optimization problem with the goal of maximizing the margin, i.e., the distance between the separating hyperplane $w^Tx + b = 0$ and the nearest training vectors. Inspired by the findings in [29], we explore the use of the weights of a linear classifier, to discriminate which regions of the templates are most important to compare with a test sample, and which are actually damaging the performance.[2] The weights vector w is computed from the support vectors as follows:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{5}$$

with y the labels ($+1$ for positive samples, -1 for negative samples), n the number of support vectors, and x_i, $i = 1:n$ the support vectors.

The elements of the weights vector w that are negative are considered to be damaging the training of templates, since those were generated mainly by support vectors from the background. We use a threshold on the elements of w so as to select only those elements that actually contribute to the object itself (or at least contain relevant, i.e. positively correlated, context information).

Fig. 3 shows an example of a training image and the generated masks. Clearly, the learnt masks successfully remove some of the gradients found on the background, that do not contribute to the quality of the classification. We can also see the difference between a region based mask and a bit level one. The latter is better distributed. Fig. 3 may give the impression that we learn a mask at the level of regions, but we have the option of applying the mask at the bit level too (i.e., for each orientation within a region independently). We found out experimentally that a bit level mask outperforms consistently a region level mask when the risk of overfitting is small, i.e. when the number of positive training samples is large enough (more than 200). In Fig. 3 we only show the dominant regions. These are calculated using the w weights learned by the SVM. For a region j we define it as dominant if $C_j > 0$, where:

$$C_j = \sum_{i=1}^{n_0+1} w_{ji} \tag{6}$$

where n_0 is the number of orientations per region.

One might think that the masks are useful as a segmentation tool during testing. However, this only works when the training samples are not noisy. In other experiments (e.g. dealing with cars) we observe that the learnt masks look rather noisy. However, in both cases (noisy or not), an accurate segmentation can be obtained by transferring meta data from the annotations of training data at the detection stage (see Section 5.5).

Discussion. At this point, we would like to discuss an alternative scheme that, at first sight, may seem more promising and more straightforward than the scheme we propose here. This alternative scheme consists of simply training a linear SVM on top of the DOT descriptors. By doing so we effectively learn the weights for the different elements of the DOT descriptor. This avoids the need for learning a mask as a preprocessing step and may approximate the good performance obtained with HOG + SVM. However, this cannot be implemented without sacrificing speed, which was the main motivation for choosing DOT instead of HOG in the first place. Indeed, incorporating the weights of the SVM means we can no longer stick to simple bit-wise operations at test time.[3] Without the speed advantage, there is no longer a reason to use DOT instead of HOG. An indirect time/accuracy comparison of HOG + SVM vs our method is provided in Section 5.4.

Also, it is important to realize that DOT is an inherently asymmetric scheme. When creating a template, DOT applies jittering to create robustness and keeps n_0 gradients. At test time, it uses only the strongest gradients and no jittering. This asymmetry is essential, and cannot easily be incorporated in a kernel-based

[1] Note how this is significantly different from the scheme used, e.g., by HOG + SVM, where the SVM learns a weight for each HOG cell/region. We'll come back to this later.

[2] Note that, while we want to avoid using an SVM at test time (as it jeopardizes the speed advantage of using DOT instead of HOG), this is less of an issue during training.

[3] Let us assume we have 256 regions coded into 1 byte each. Training a linear SVM would assign a double type weight for each bit, resulting in 2048 double values to be computed for each sub-window in the test image we want to evaluate. When we learn a binarized mask separately, on the other hand, we can incorporate the binary weights directly in the template (by ANDing the mask and template), so at test time only binary operations are needed. Moreover, using the same similarity function as in [4], we need only $5\lceil \frac{n}{16} \rceil$ operations. This results in just 80 AND binary operations per template.

framework like SVM. Leaving out the jittering for the templates, it would no longer be possible to skip locations while parsing a test image, slowing down the method even further. Moreover, it would make the templates more specific and reduce their generalization capabilities. Adding jittering to the DOTs extracted from the test images, on the other hand, would also introduce a strong computational overhead.

4.4. Multiview extension

Learning the masks works best if all the positive examples provided to the SVM are somewhat similar. However, in a multiview context, the within-class variability is very high, and we cannot expect a single template to be selective for all viewpoints. Therefore, as advocated in [15], it is better to first cluster the data in an unsupervised manner. We then use only the positive examples within one cluster for learning the mask. This results in specialized weak classifiers that focus on rejecting background and only classifying properly a subset of the positive training samples.

Clustering based on the binary DOT representations proved difficult and unstable. Therefore, we start from HOG representations for this step. Since this step is performed offline (during training), the higher computational cost of HOG over DOT is not really an issue. An intuitive way of clustering objects is to apply k-means directly on top of the HOG descriptors. However, this approach performs poorly. Here we follow the clustering approach proposed in [15], using Locally Linear Embedding [30] and then running k-means on the resulting 2D features. Clustering the multiple views in our training set is essential to obtain high quality masks and thus better templates. Fig. 4 shows the 12 clusters we obtain for the dataset proposed in [15].

To create the mask m_a for training image a, we first compute a template t_a (allowing $n_0 = 7$ non-zero orientations), and then AND each of the positive samples of the cluster it belongs to with the current t_a, using only the single strongest orientation for the positive sample. This gives us binary feature vectors of size $R * (n_0 + 1)$ that are used as positively labeled input for the SVM, where R is the number of regions/bytes. Fig. 5 shows this process. We randomly sample sets of negatives, and AND them with the t_a template in a similar fashion, so as to obtain the negative feature vectors. Then we train the SVM and keep those elements of the template for which the corresponding w values are above a threshold. In the experiments we use different thresholds not only ZERO, but also ZERO$\pm$*Small value* to increase the pool of features, and let Adaboost select the best ones. We repeat this process for all positive templates t_a.

4.5. Mask learning for binary templates on specific object detection

The proposed masks can also be applied to template based detectors for specific object detection, without using a strong classifier, but only a look up table as in the original baseline. For each template t_a of an image a trained with homogeneous background as in [4], we can learn efficiently an optimization mask. Since negative images are very easy to obtain, we pre-compute a set of 10,000 negative samples taken randomly from 100 images not containing the objects we want to train.

Then, when we learn a template t_a, we propose 3 different ways of optimizing it using the negative samples as a validation set. We use this set to define parameters independently of each template because using the same parameters for all templates is suboptimal: some of them tend to be less discriminative, others produce more false positive. The three optimization methods we propose are:

(i) *DOT Opt:* Using a variable number of regions $R \pm r$. This optimization does not use SVM. To select the strongest regions, we use the gradients magnitudes as defined in the baseline. We compute the $R + r$ strongest dominant regions $\mathcal{R}$ (as defined in Section 4.4). Then, using the template t_a, we calculate the similarity with Eq. (2) for each negative sample. If similarity is bigger than a threshold value (65%), we count it as an error. We decrease the number of regions (eliminating first the regions with smaller gradient magnitudes) until we reach a target error, or the minimum allowed regions $R - r$.

(ii) *DOT SVM-Region:* This is using a variable number of regions $R \pm r$ and selecting the dominant regions based on values obtained with Eq. (6) (trained using a linear SVM). We follow the same procedure as in (i), but here we eliminate first the regions with smaller C_j value (the value computed with Eq. (6)). In order to train the SVM, we use as positive elements only the template t_a, and we AND it to a random set of negatives (in the order of hundreds, in experiments we use 500) to create the negative elements. In this way, we use the w weights to focus on the most important regions that t_a has.

(iii) *DOT SVM-Bit:* Starting from R regions provided by the original DOT, we train the SVM in a way similar to those above. Using the validation set, we determine which bits will be converted into zeros. This is done using again the negatives validation set and in a way similar to those above, we eliminate the elements corresponding to the w weights with lowest values in a bitwise manner. When calibrating a template, we define a small target error bigger than zero. We observed that setting zero as target, might cause some templates to eliminate several bits due to the hard negative samples, producing less generalization.

We use the same set of negative images for all the trained objects. We found out experimentally that the DOT SVM-Bit optimization consistently outperforms all the others. See Section 5.6.

5. Experimental results

The experimental results section is structured as follow: In Section 5.1, we show the effect of different types of masks. In Section 5.2 we evaluate DOT for category level detection. In Section 5.3, we

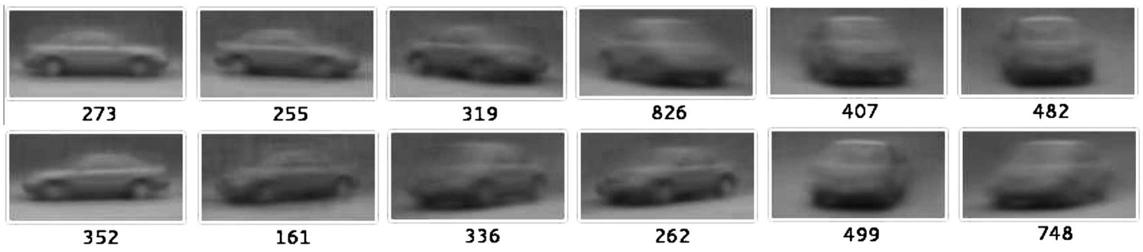

Fig. 4. Average images of 12 clusters (with size of the cluster indicated below each image, 4924 images in total).

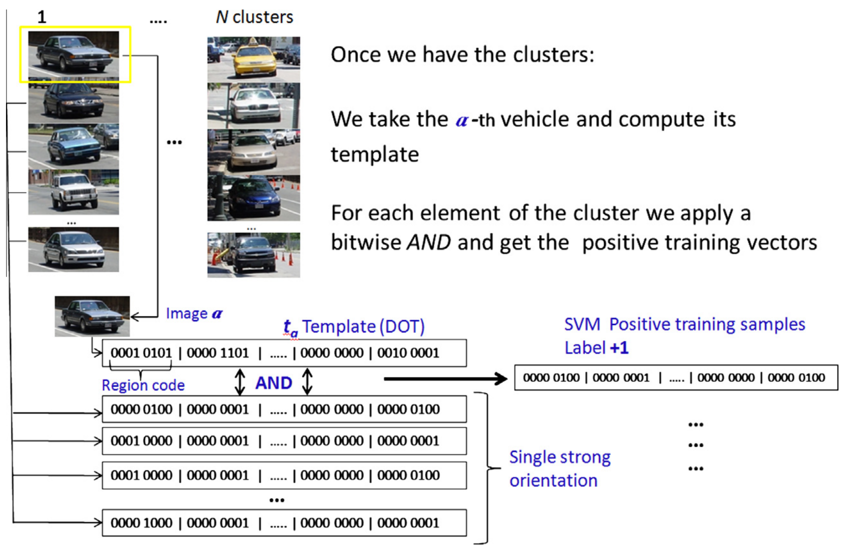

Fig. 5. Using clusters to compute the masks. When processing $a-th$ element, for each image of the cluster, we apply bitwise AND in order to get the positive training samples.

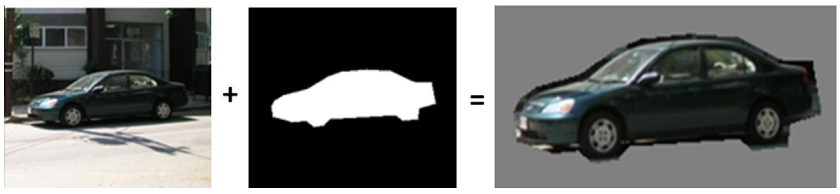

Fig. 6. An image of a car, plus its corresponding binary segmentation mask in a normalized bounding box.

apply our method to the detection of different shape based categories. Section 5.4 shows results of our system in the context of multi-view vehicle detection. In all cases, we report results that are competitive with the state-of-the-art while being much faster. Section 5.5 shows how to efficiently transfer metadata from annotations into detections. Finally in Section 5.6, we present results on improving baseline of [4] on specific object detection.

5.1. Effect of learning the mask

For the first set of experiments we use the vehicles dataset of Leibe et al. [31]. It contains multiview vehicles for 7 different orientations grouped in 7 folders, and it comes with a manually annotated pixelwise segmentation mask for each vehicle. The dataset contains 1262 vehicles, not including the ones labeled as incomplete masks. We use 50 more vehicles from there to increase the training samples, resulting in a total of 1312 samples, of which 700 are used for training and 612 for testing. Additionally, we collect 8427 negative examples plus 1300 bootstrapped hard negatives for training, all sampled from Pascal VOC2007. For testing we collect 0.5 million negative samples from 1043 Pascal VOC2006 images not containing cars.

Implementation technical details. For this experiment, we use the following settings. First, all training samples are normalized to 150×65 pixels. We use a grid of 30×13 regions (5×5 pixels per region, 390 regions, jittering as described in Section 3) and a threshold on gradients of 20. The number of valid regions $\mathcal{R}$ used (see [4]) is set to $R = 256$. For this experiment, we did not cluster the training images ourselves, but instead use the ground truth orientations provided by the dataset.

The masks are then learnt using SVM. We use LIBLINEAR [32], because of its speed and the sparsity of our vectors. We use $C = 0.05$, with compensation of weights for training unbalanced data. We use $n_0 = 7$ orientations per byte plus the bit for homogeneous regions (no gradient). To create the mask we follow the procedure described in Section 4.4. Training all SVMs takes less than 3 min for $M = 700$ templates when using 4 cores in an Intel i7 @2.8 GHz computer.

The advantage of using a mask. In this experiment, we compare four different settings: (i) using the whole image as template (No-Mask), (ii) using the manual masks as seen in Fig. 6, provided by [31] (Manual Mask) (iii), using a learnt mask at region-level (SVM-Region Mask) and (iv) using a learnt mask at bit-level (SVM-Bit Mask). Fig. 7 shows the results (miss rate vs. false positives per window), obtained with a strong classifier consisting of 100 boosted templates (left) and another using 250 templates (right). The positive effect of using masks is evident. While the use of manual masks gives only a limited improvement over the baseline without masks (up to 5%), the learnt masks clearly perform much better, with a drop in the miss rate at 10^{-4} FPPW up to 15% for the region-level masks and about 20% for the bit-level masks.

The strong improvement over the manual masks may seem surprising at first, but can be explained by the fact that the learnt masks also include context information where useful, while at the same time ignoring unstable or irrelevant gradients within the object. We can conclude that learning masks seems a critical ingredient if one wants to obtain good results with our template based Adaboost scheme.

To analyze possible overfitting, we used 2 sets, one divided into training and validation, the second used for testing. Varying sets changes less than 1% at 10^{-4} FPPW in several configurations. If a weak classifier overfits on its subset, Adaboost will not select it as a good feature. Adaboost is robust against overfitting.

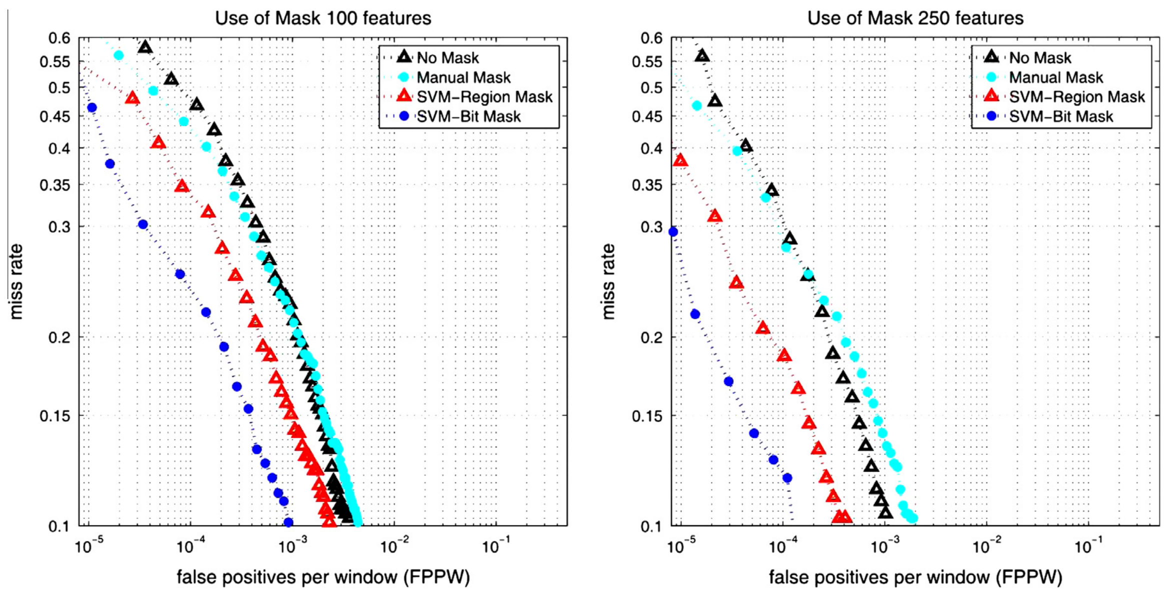

Fig. 7. Analysis of different masks. Left: Using 100 DOT Classifiers for: no mask, manual mask, SVM-region mask and SVM-bit mask. Right: Using 250 DOT classifiers.

5.2. Comparison with DOT baseline: category level

In an attempt to compare our method with the DOT baseline of [4] (developed for the detection of specific objects), we have implemented a version of our method as equivalent as possible to the baseline proposed in [4]. We use for this purpose, the vehicles data set of Leibe et al. [31]. Since [4] assumes uniform backgrounds for training, we use the provided manual masks in the dataset to filter out the background. Then we run all templates independently followed by normalized non-maximum suppression (NMS) and a global normalized threshold. This is similar to the look up table used in [4].

However, the baseline results using NMS were very poor compared to our method. To make this simple approach cope with the within-class variability, several low performing templates were needed. These templates are needed to cover some type of vehicles, but create lot of false positives. We tried improving the baseline in various ways e.g. by adapting the thresholds for each template individually and selecting the best templates, but still results were very poor and no match against the boosted version. At 10^{-3} FPPW, the baseline reaches only 7% detection accuracy. This is no match against the boosted No-Mask (79%), and Bit-Mask (91%), both using 100 templates. At 10^{-4} FPPW, the baseline drops further to 0%. Conclusion: [4] by itself is not suited for category-level detection. This might be because in boosting (1) we use a combination of several weighted templates to decide if a detection is correct or not, (2) we can also include negative templates in the strong classifier, and (3) we learn the optimal weights and thresholds.

5.3. Learning different categories

Next, we evaluate our scheme on the challenging ETHZ Shapes dataset of [21]. This dataset contains five diverse, mostly shape-based object categories, with 255 images in total. All categories have significant scale changes and intra-class variation.

Implementation details. We follow the same evaluation protocol of [21] (detection rates at 0.3 and 0.4 FPPI, with a detection considered correct when the intersection over union of bounding boxes is $\geqslant 0.5$). We report the average detection rate over 5 splits with half of the images for training and half for testing. Note that, as shown in [21], learning from bounding boxes is in this context a significantly harder problem than learning from a manually drawn model. Our method uses 1000 random negative samples obtained from PASCAL VOC 2007 dataset, plus about 100 hard samples obtained via bootstrapping. We normalize the training samples of each category to a training size with aspect ratio adapted to the class.

Results. The results are summarized in Table 1. For the apples category, a single SVM-Bit mask classified all training samples correctly (both positives and negatives), indicating overfitting. So for this category we switched to the SVM-Region masks, which generalized better. Lowering the overlap criterion slightly (0.4 instead of 0.5) we achieve 100% detection at 0.3/0.4 FPPI, using only 40 templates for each trial. In order to select a better bounding box fit, we retrieve the template (s) that contribute most to the detection, and select the best. Based on that we recover the original training aspect ratio and obtain a more accurate bounding box. Looking at the results for the other categories, we can conclude our method performs very well when the category is rigid (although, admit-

Table 1
Comparing different methods: (a, b) [22], (c) [20], (d) [23], (e) [24], (f) [21], Detection rates are measured using the PASCAL criterion (50% overlap).

Category	(a) Ommer 1	(b) Ommer 2	(c) Ferrari KAS	(d) M2 HT + IKSVM	(e) Fritz	(f) Ferrari full system	(g) Our method
Evaluation FPPI = 0.3/0.4							
Apples	95.0/95.0	95.8/96.6	50.0/60.0	95.0/95.0	–/89.9	77.7/83.2	98.0/98.0
Bottles	89.3/89.3	89.3/89.3	92.9/92.9	92.9/96.4	–/76.8	79.8/81.6	87.3/90.8
Giraffes	70.5/75.4	73.9/77.3	49.0/51.1	89.6/89.6	–/90.5	39.9/44.5	73.2/74.2
Mugs	87.3/90.3	91.0/91.8	67.8/77.4	93.6/96.7	–/82.7	75.1/80.0	92.6/93.6
Swans	94.1/94.1	94.8/95.7	47.1/52.4	88.2/88.2	–/84.0	63.2/70.5	87.5/87.5
Average	87.2/88.8	88.9/90.1	61.4/66.8	91.9/93.2	–/84.8	67.2/72.0	87.7/88.8

tedly, not as good as [23], which, however, is not as fast as our scheme, and uses several iterations to perform detections). The case of giraffes, being a very flexible category, shows the limitations of our method. Fig. 8 shows some detection examples.

We show that our method can compete with other shape based methods. Even though we get similar/slightly lower results, our method is designed to run in realtime. As far as we know, none of the other methods was designed to be fast, and no computation time is reported in the other papers. Our method performs relatively high on rigid categories: apples (best), bottles and mugs.

Discussion. An advantage of shape based methods such as [20,21,23] is that they can approximate the object boundaries. However, they usually cannot include context information into the training, and it is more difficult to include also negative samples, as in discriminative approaches. In our proposal, in order to approximate the object boundaries, we retrieve the template (s) that contributed most to the detection, and select the best. Based on that we can recover the original training aspect ratio and obtain a more accurate bounding box. This also gives us the possibility to transfer meta data to the specific sample, as described in Section 5.5.

5.4. A vehicle detector

Next, we evaluate our method on the USC vehicles dataset of [15]. We use the 2462 vehicles of the data set and their mirrored versions to generate positive templates as a pool for training Adaboost. For negative data, we use 8427 samples plus about 2500 hard samples obtained in a bootstrapping retraining process. The negative images are taken from PASCAL VOC2007 train/val data set, as in [15]. Using as first step an automatic clustering we create 12 clusters, as shown in Fig. 4. Based on these, we learn our masks, and then learn a 500-DOT strong classifier.

The data set provided in [15] contains data for two types of evaluation: in terms of FPPW (false positives per window) and in terms of precision-versus-recall. First we perform a per window evaluation (see Fig. 9, left). Our method outperforms Clustered Boosted Tree [33] and can compete with Robust Multiview [15]. In a second experiment, we also evaluate our detector per image, using a sliding window and Non-Maximum Suppression. We annotated each of the 12 clusters to an average adjusted bounding box. This annotation was retrieved using the top positive templates during testing time, and then averaged. This results in the precision recall curve of Fig. 9 (Right). Again our results outperform Clustered Boosted Tree [33] on this data set, and Robust Multiview Car detector [15], especially in the low recall - high precision regime. Fig. 11 shows detection examples. Recently, a new vehicle detector applied on this data set was presented using Stochastic Gradient Descent (SGD) and a splitting/clustering method over HOG templates [10]. This method is an equivalent implementation of HOG + SVM when using a single cluster, but faster. They show impressive results of almost 100% at 10^{-4} FPPW when using 9 views, and substantially outperform all other methods. In the precision recall comparison, they achieved AuC = 95.2, slightly better than ours (AuC = 93.7). However, even though their method is faster than HOG + SVM, they still need to compute HOG, and they need also a separate strong classifier for each view.

Speed analysis. In terms of training time, our detector needs about 15 min to compute 4800 masks, and another 10 min for learning the strong classifier in a non-optimized implementation. This compares favorably with [15] that needs two days. At test time, the method in [15], needs 1.5 s to parse an image in 1.0 scale. In [10], the computational complexity scales linearly with the number of sub-classes (views). The detection stage at scale 1.0, takes 220 ms to calculate the HOG features and 110 ms to apply the detector for a single class and they claim, it takes 730 ms per image (the arithmetics do not sum up for the 9 views, they argue this is due to cache memory), using a i7-2600 k processor at 3.4 GHz with a single-threaded non-optimized implementation. Our method, on the other hand, takes 15 ms for calculating the dominant orientations and 80 ms to parse a single image in an Intel i7 @2.8 GHz computer, used single-threaded. When adding a (non-optimized) cascade structure [16], we obtain a further speedup by a factor 5 (without sacrificing performance). This brings our total processing time for parsing an image at scale 1.0 down to 31 ms

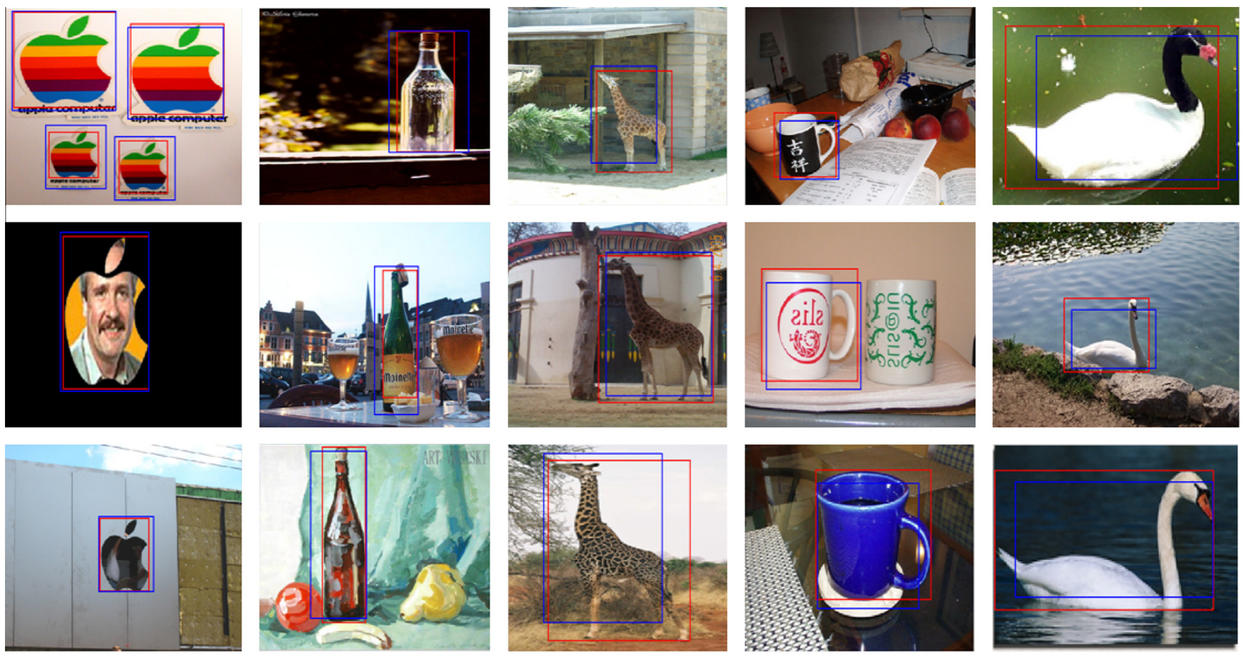
Fig. 8. Examples of detections in the ETH-shapes dataset (in blue) and the corresponding ground truth annotations (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

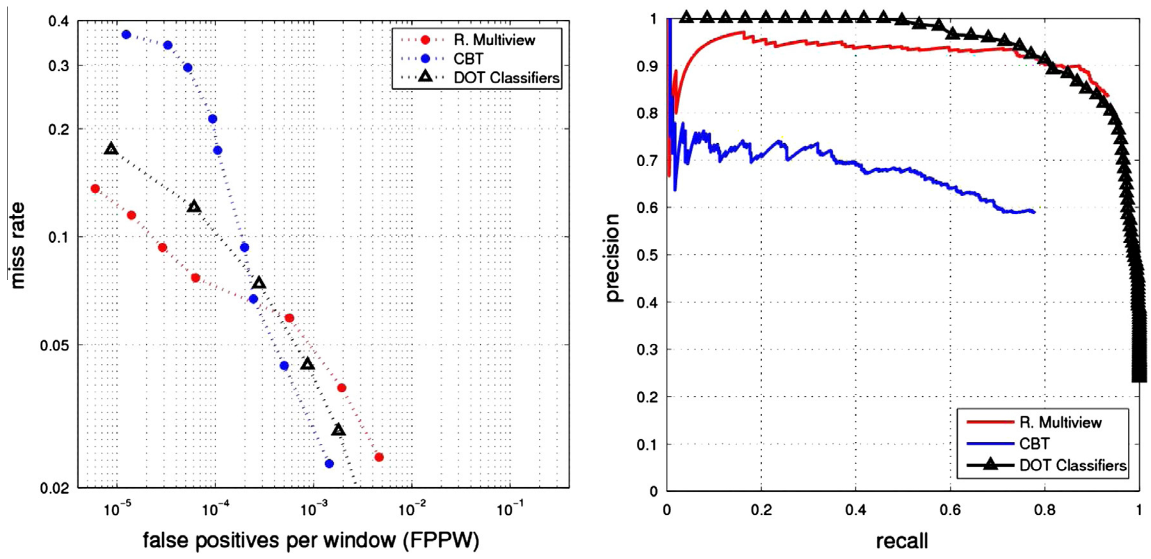

Fig. 9. (a) Detection curves on USC vehicles dataset. (b) Precision/recall curve on USC vehicles dataset. Kuo et al. [15] (R. Multiview) gets AuC = 86.5, while our method achieves AuC = 93.7.

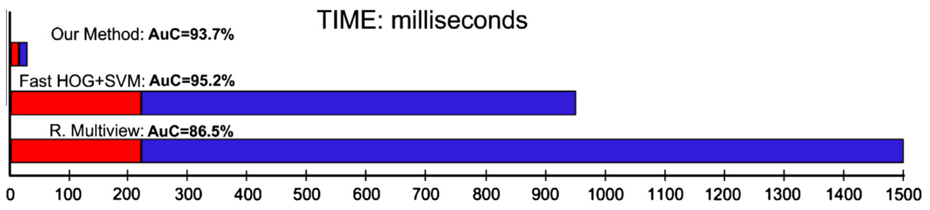

Fig. 10. Processing testing time for an image in 1.0 scale, comparing our method vs. Fast HOG + SVM [10], R. Multiview [15]. Red: time of pre-procesing (Dominant gradients, or HOG). Blue: time parsing the image. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

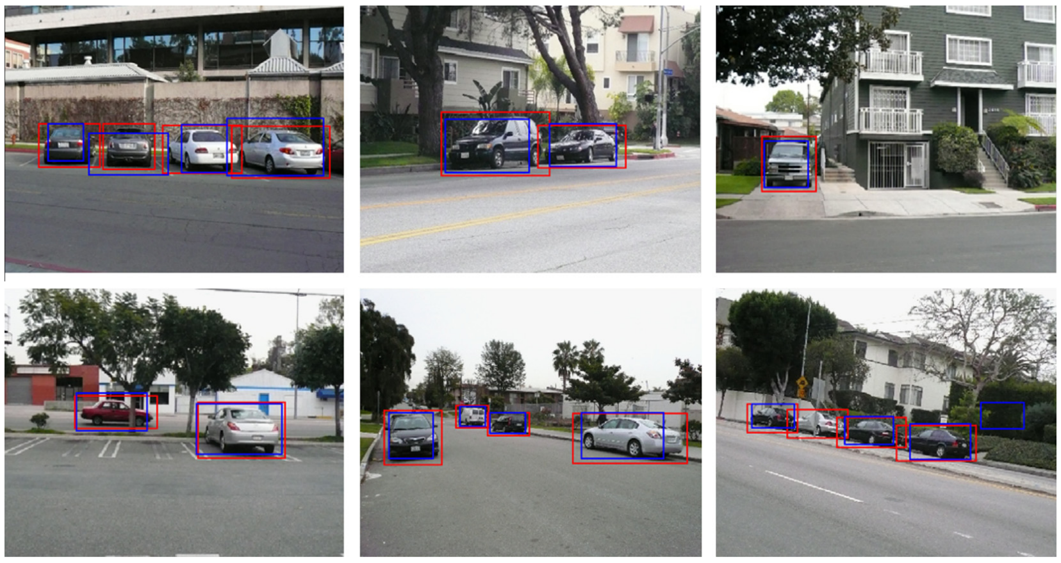

Fig. 11. Detected vehicles (in blue) and the corresponding ground truth annotations (red). The last image shows a false positive and a false negative where our method fails. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

or about 24 times faster on a slower machine. See Fig. 10 for a graphical comparison.

5.5. Transfering meta data

In this section, we show how we can transfer the segmentation mask of the training sample to a test sample. With the same method, more metadata can be defined offline and transferred at testing time, e.g. material types.

During detection, several templates are applied to evaluate a window. Using the activated top 5 most similar positive training-templates, we can retrieve a segmentation mask for the detected vehicle from annotated meta data. To perform this experiment, we used the dataset provided by [31], using the provided manual

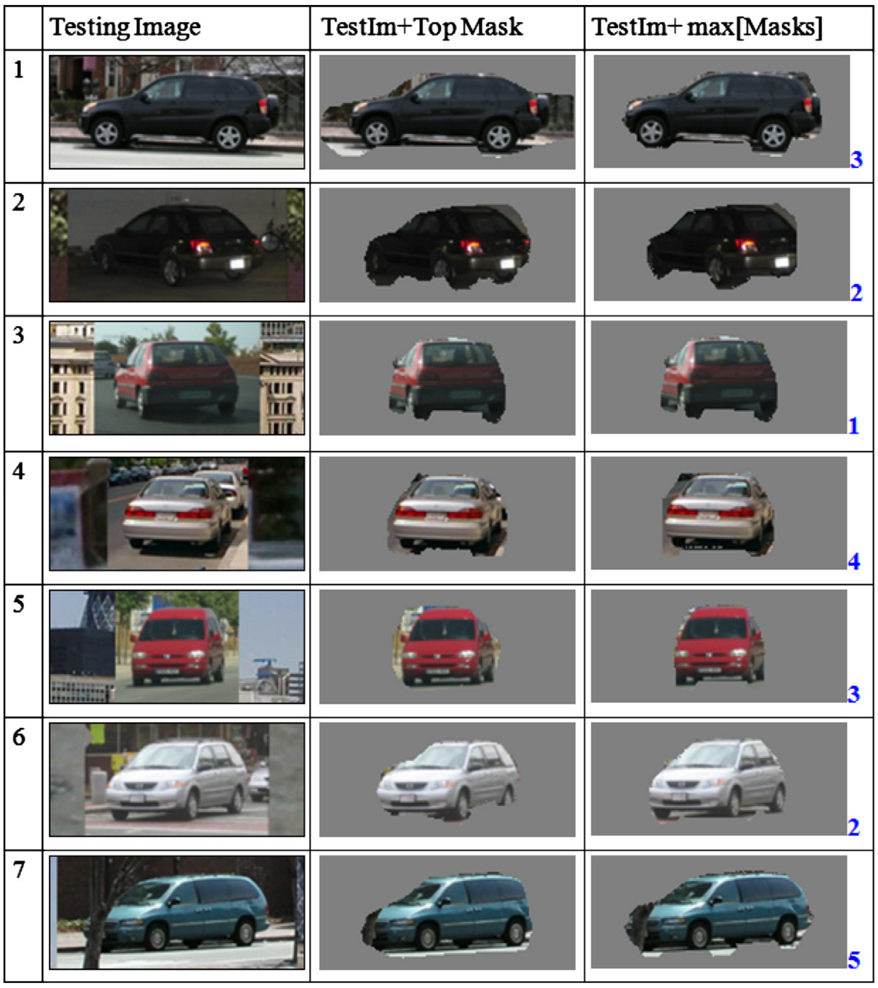

Fig. 12. Metadata transfer using the top 1 template (middle column) and the top 5 templates masked maximum (right). Numbering on the right is the rank of the final selected template.

masks as in Fig. 6 for the 700 training/612 testing vehicles. We perform two experiments:

(i) For the first experiment, we only use the single top most similar template. Then we perform a pixelwise comparison (intersection/union) of the segmented pixels using the selected training template mask and the manually annotated mask. This results in an overlapping accuracy of 85.3% as average over all 612 testing vehicles.

(ii) In a second experiment, we apply first each of the top 5 recalled template masks on the testing image and then evaluate a bitwise similarity using Eq. (2) to all 5 top templates. Then we select the maximum response. With this approach we obtain 86.6%, which is 1.3% better than when using directly the most similar before applying masks as in experiment (i). Fig. 12 shows some examples for both experiments (i) and (ii). Fig. 13, shows the top 5 templates recalled in the testing process. When using the top 10 templates accuracy increases only slightly to 86.7%. It is important to notice that, while the re-called mask are the binary manual annotations, in the detection process, the SVM-bit mask is used to compute the *DOT Classifiers*. The obtained segmentation masks can be refined further by integrating a bottom up segmentation algorithm.

Another scheme for transferring metadata has been proposed by [25], based on exemplar SVMs. They use global appearance similar to a template. Since a single template/exemplar cannot generalize in both categories and viewpoints, they use all positive samples to train the same amount of SVMs (thousands). Each SVM is focused only on a specific sample/similar appearance vs all negatives. When testing, they compute all SVMs, followed by a normalized non-maximum suppression. This is very inefficient. Our method is substantially different. We use SVM only to create binary masks to later be boosted, and we show how to make metadata transfer efficient.

5.6. Improving DOT baseline: specific object detection

In this set of experiments, we compare the baseline implementation that we call DOT Original, and the three methods for optimization that we propose in Section 4.5: DOT Opt, DOT SVM-Region, DOT SVM-Bit, applied to specific objects. In particular, we trained 4 objects: Cup, Camera, Hole-Punch and Toy-Gun.

The testing was done on 1000 heavily cluttered images (500 annotated with each object, plus 500 images without object).

The mask training is able to run at online learning speed. For the DOT SVM-Region, DOT SVM-Bit, the optimization of a given template takes about 100 ms in an Intel i7 @2.8 GHz computer on a single thread. This time includes the whole optimization process: (1) training the SVM with 1 positive sample (the template), plus 500 random negatives from a set of 10,000, and (2) the tuning using the whole validation set.

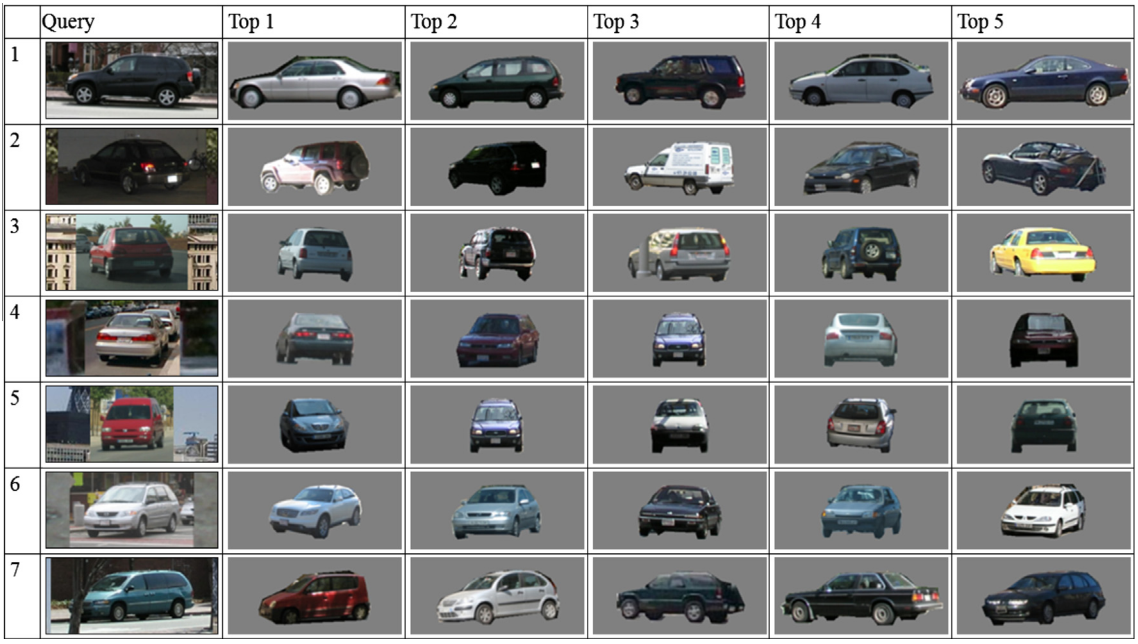

Fig. 13. Top 5 templates for testing images of Fig. 12, using the annotated masks. First column shows the query/testing image. Columns 2–6 show the ranked retrieved template images.

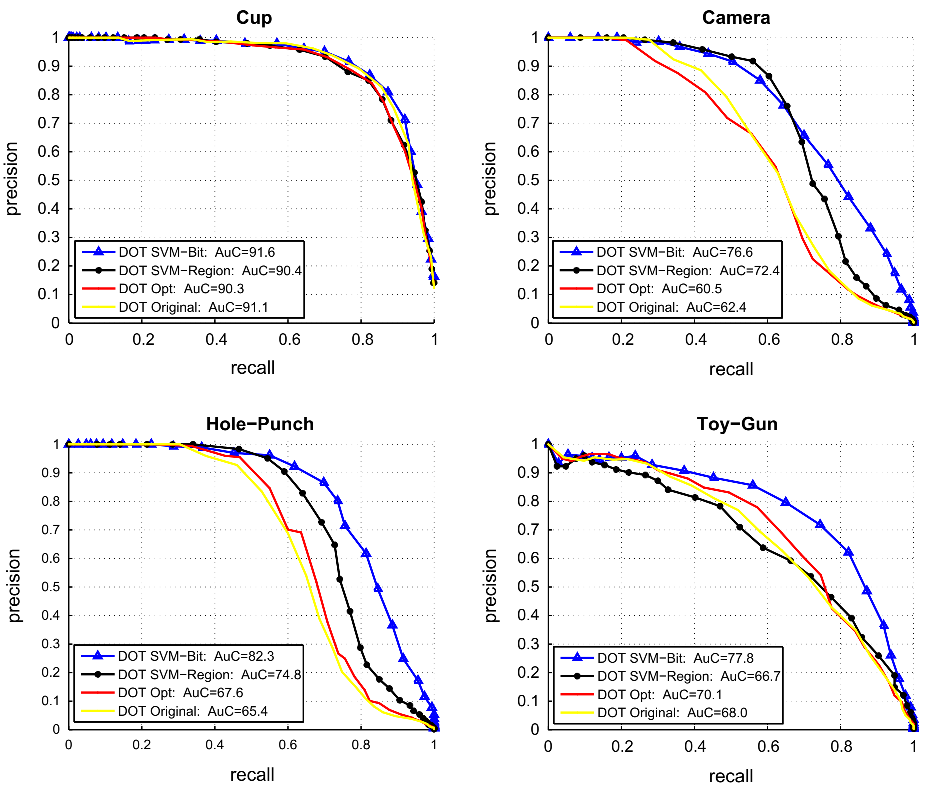

Fig. 14. Precision-recall results for Cup, Camera, Hole-Punch and Toy-Gun. We compare the original DOT vs. Our methods. DOT SVM-Bit outperforms all other methods.

To train the SVM, we use LIBLINEAR [32], and we modified it to read/write the samples from shared memory instead of files. We used $C = 0.05$, with compensation of weights for training unbalanced data. Also, when copying the training elements to the SVM, we only use the non-zero elements of the template and the corresponding regions of the negative samples modified with the AND operation, this speeds up training. To train/calibrate each template for the 4 different objects, we use the same set of 10,000 negative samples.

We trained 200 templates per object. In order to use exactly the same templates on each experiment, we first train the 4 objects using DOT and save the training images. Then we use these images

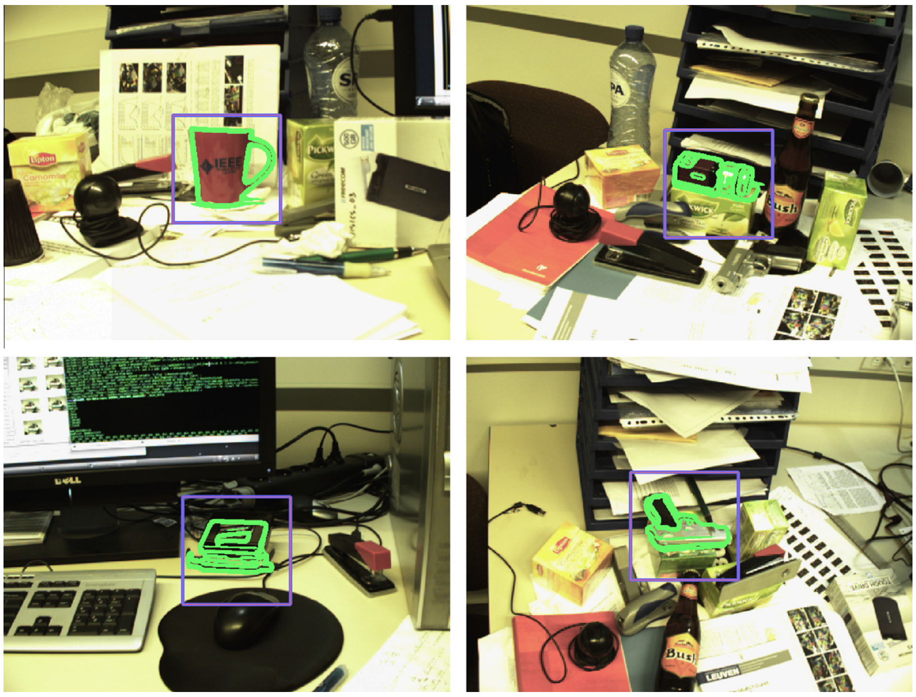

Fig. 15. Detection examples for Cup, Camera, Hole-Punch and Toy-Gun.

for the other methods. Since optimizing the template takes only about 100msec, we can use it in online training. On testing time, the speed is exactly the same as DOT Original, since once the template is optimized, we AND the masks only once. We use $R = 120$ regions for all experiments, and $r = 20$ for the DOT Opt, and DOT SVM-Region optimization (which means: $R = 120 \pm r = 20$), having a maximum of 140 active regions, and a minimum of 100. We use for DOT Original and DOT SVM-Bit, the same regions. DOT SVM-Bit suppresses those bits damaging most the template discriminative power, and maintains the most useful bits to reject false positive.

We computed the precision-recall curves for each experiment. Fig. 14, shows results on the 4 objects. For the Cup, the accuracy reached by DOT is already high, so the improvement of optimization is very small (0.5%). For the other objects, comparing DOT original with the DOT SVM-bit, we obtain an increment of: 14.2%, 16.9%, 9.8% for the Camera, Hole Punch and Toy-Gun respectively. This is consistent with the improvement of bitmask demonstrated in Section 5.1. DOT SVM-bit improves over the original implementation on all 4 objects, using the same regions than DOT, but optimized with masks. Fig. 15 show an example detection frame for each object.

6. Future work

Before we conclude our paper, we discuss some research lines that we consider as interesting directions for future research:

DOT, LINE2D and LINEMOD [5] are similar methods based on efficient bitwise operations. Our template optimization and category-level proposals can be applied to the other methods as well leading to an improvement of performance.

Optimizing for speed. So far, we did not optimize our detector yet. Still, the inherent use of bit-wise operations already makes it fast. In [5], speeds of 10 fps were reported on a VGA image, using 3000 templates for an implementation combining the binary LINE2D representation and dense depth map. Our scheme uses almost an order of magnitude less templates (highest number we tried was 500 templates, for the USC vehicles data set), and therefore should be able to run significantly faster. Moreover, our method can easily be parallelized.

Handling multiscale. In [5], the handling of multiscale is done by training different sizes of templates. This solution works only if the number of scales is small (say within one octave) because it trains all sizes in the same table and many scales would lead to a slow system. As seen in [8], the key to handling multiscale images efficiently is to use the feature responses computed at a single scale to approximate feature responses at nearby scales. In our approach, we use the strongest gradients for detection, and different sizes of templates can be handled by resizing the images in training time and computing different masks for different scales. Doing so, we can train N cascades and speed up tremendously the detection process. Note that this is different from creating a cascade for each view, which escalates linearly in time with the number of viewpoints.

7. Conclusions

We have presented a new method for (multiview) category-level object detection. It is template-based, fast and highly accurate. We show how to combine different *DOT classifiers* to successfully generalize and learn to detect a class rather than an *a priori* known object. We demonstrate how to eliminate background noise that damages the detection and, at the same time, to integrate useful context information. This results in a system that obtains results competitive with the state-of-the-art, yet needs only 31msec to analyze a VGA image at scale 1.0.

Additionally, we showed how to optimize a template based detector for specific object detection, at online learning speed. We demonstrated empirically how to obtain up to 17% performance increment without sacrificing testing speed.

The efficiency of our method relies on the image representation rather than on speed up tricks, GPUs, or geometrical constrains. Ideas of methods such as [8,26] can be integrated with our method since they are complementary.

Acknowledgements

Part of this work was funded through the ERC grant COGNIMUND and PARIS Project.

References

[1] D. Gavrila, V. Philomin, Real-time object detection for smart vehicles, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 1, 1999, pp. 87–93.
[2] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Stud. Comput. Intell. – Comput. Vis. (2004) 91–110.
[3] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.
[4] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, N. Navab, Dominant orientation templates for real-time detection of texture-less objects, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2257–2264.
[5] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, V. Lepetit, Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes, in: IEEE Int. Conf. on Computer Vision (ICCV), 2011.
[6] Y. Freund, R. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: Computational Learning Theory, 1995, pp. 23–37.
[7] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2001, pp. 511–518.
[8] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Trans. Pattern Anal. Machine Intell. 32 (2010) 1627–1645.
[9] R. Wijnhoven, P.H.N. de With, Unsupervised sub-categorization for object detection: finding cars from a driving vehicle, in: Proc. IEEE ICCV Workshops, 2011, pp. 2077–2083.
[10] F. Tang, H. Tao, Fast multi-scale template matching using binary features, in: Workshop on Applications of Computer Vision IEEE WACV, 2007.
[11] L.T.-K.K. Danhang, Tang Yang, Fast pedestrian detection by cascaded random forest with dominant orientation templates, in: British Machine Vision Conference (BMVC), 2012.
[12] S. Savarese, L. Fei-Fei, Multi-view object categorization and pose estimation, Stud. Comput. Intell. – Comput. Vis. (2010) 205–231.
[13] X. Perrotton, M. Sturzel, M. Roux, Implicit hierarchical boosting for multi-view object detection, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 958–965.
[14] C. Kuo, R. Nevatia, Robust multi-view car detection using unsupervised sub-categorization, in: Workshop on Applications of Computer Vision (WACV), 2009, pp. 1–8.
[15] C. Zhang, P.A. Viola, Multiple-instance pruning for learning efficient cascade detectors, in: Conf. on Neural Information Processing Systems (NIPS), 2007.
[16] B. Yao, G. Bradski, L. Fei-Fei, A codebook-free and annotation-free approach for fine-grained image categorization, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 2012.
[17] Q. Zhu, M. Yeh, K. Cheng, S. Avidan, Fast human detection using a cascade of histograms of oriented gradients, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2006, pp. 1491–1498.
[18] I. Laptev, Improvements of object detection using boosted histograms, in: British Machine Vision Conference (BMVC), 2006.
[19] F.J.V. Ferrari, L. Fevrier, C. Schmid, Groups of adjacent contour segments for object detection, PAMI 30 (1) (2008) 36–51.
[20] F.J.V. Ferrari, L. Fevrier, C. Schmid, From images to shape models for object detection, in: IJCV, 2009.
[21] J.M. Bjorn Ommer, Multi-scale object detection by clustering lines, in: IEEE Int. Conf. on Computer Vision (ICCV), 2009.
[22] S. Maji, J. Malik, Object detection using a max-margin hough transform, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2009.
[23] B. Fritz, M. Schiele, Decomposition, discovery and detection of visual categories using topic models, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2008.
[24] T. Malisiewicz, A. Gupta, A.A. Efros, Ensemble of exemplar-svms for object detection and beyond, in: IEEE Int. Conf. on Computer Vision (ICCV), 2011.
[25] P. Dollar, S. Belongie, P. Perona, The fastest pedestrian detector in the west., in: British Machine Vision Conference (BMVC), 2010, pp. 1–11.
[26] R. Benenson, M. Mathias, R. Timofte, L. Van Gool, Pedestrian detection at 100 frames per second, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2012.
[27] S. Taylor, T. Drummond, Multiple target localisation at over 100 fps., in: British Machine Vision Conference (BMVC), 2009.
[28] J. Uijlings, A. Smeulders, R. Scha, What is the spatial extent of an object? in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 770–777.
[29] D. Mladenić, J. Brank, M. Grobelnik, N. Milic-Frayling, Feature selection using linear classifier weights: interaction with classification models, in: ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2004, pp. 234–241.
[30] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323.
[31] B. Leibe, N. Cornelis, K. Cornelis, L. Van Gool, Dynamic 3d scene analysis from a moving vehicle, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.
[32] R. Fan, K. Chang, C. Hsieh, X. Wang, C. Lin, Liblinear: a library for large linear classification, J. Machine Learn. Res. 9 (2008) 1871–1874.
[33] B. Wu, R. Nevatia, Cluster boosted tree classifier for multi-view, multi-pose object detection, in: IEEE Int. Conf. on Computer Vision (ICCV), 2007, pp. 1–8.