Generic Object Class Detection using Boosted Configurations of Oriented Edges

Oscar Danielsson and Stefan Carlsson
{osda02,stefanc}@csc.kth.se

School of Computer Science and Communications
Royal Inst. of Technology, Stockholm, Sweden

Abstract. In this paper we introduce a new representation for shape-based object class detection. This representation is based on very sparse and slightly flexible configurations of oriented edges. An ensemble of such configurations is learnt in a boosting framework. Each edge configuration can capture some local *or global* shape property of the target class and the representation is thus *not limited* to representing and detecting visual classes that have distinctive local structures. The representation is also able to handle significant intra-class variation. The representation allows for very efficient detection and can be learnt automatically from weakly labelled training images of the target class. The main drawback of the method is that, since its inductive bias is rather weak, it needs a comparatively large training set. We evaluate on a standard database [1] and when using a slightly extended training set, our method outperforms state of the art [2] on four out of five classes.

1 Introduction

Generic shape-based detection of object classes is a difficult problem because there is often significant intra-class variation. This means that the class cannot be well described by a simple representation, such as a template or some sort of mean shape. If templates are used, typically a very large number of them is required [3]. Another way of handling intra-class variation is to use more meaningful distance measures than the Euclidean or Chamfer distances typically used to match templates to images. Then, the category can potentially be represented by a substantially smaller number of templates/prototypes [4]. A meaningful distance measure, for example based on thin-plate spline deformation, generally requires correspondences between the model and the image [4–6]. Although many good papers have been devoted to computing shape based correspondences [7, 4], this is still a very difficult problem under realistic conditions. Existing methods are known to be sensitive to clutter and are typically computed using an iterative process of high computational complexity.

Another common way of achieving a compact description of an object category is to decompose the class into a set of parts and then model each part independently. For shape classes, the parts are typically contour segments [8,

9, 5], making such methods practical mainly for object classes with locally discriminative contours. However, many object classes lack such contour features. These classes might be better described by global contour properties or by image gradient structures that are not on the contour at all, as illustrated in figure 1.

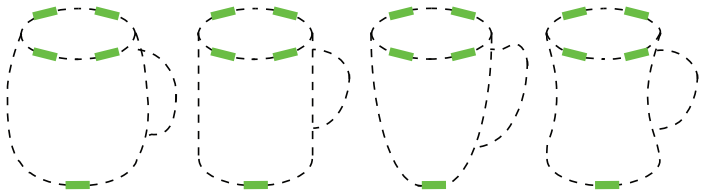


Fig. 1. *Intra-class shape variation too large for efficient template-based representation. No discriminative local features are shared among the exemplars of the class. However, exemplars share discriminative edge configurations.*

In this paper we present an object representation that achieves a compact description of the target class without assuming the presence of discriminative local shape or appearance features. The main components of this representation are sparse and slightly flexible configurations of oriented edge features, built incrementally in a greedy error minimization scheme. We will refer to these configurations as weak classifiers. The final strong classifier is a linear combination of weak classifiers learnt using AdaBoost [10]. We also stress that the method can be trivially extended to include other types of features (for example corners or appearance features) when applicable. We use the attentional cascade of Viola and Jones for selection of relevant negative training examples and for fast rejection of negative test examples [11]. For detection we evaluate the sliding-window approach and we also point out the possibility of a hierarchical search strategy. Feature values are in both cases computed efficiently using the distance transform [12]. We achieve detection performance comparable to state-of-the-art methods.

Our main contributions are: (i) a generic, shape-based object class representation that can be learnt automatically in a discriminative framework from weakly labelled training examples, (ii) an efficient detection algorithm that combines an attentional cascade with efficient feature value computation and (iii) an evaluation showing very promising results on a standard dataset.

In the following section we review other methods that use similar object representations. The rest of the paper is organized as follows. In section 3 we describe how our method represents the object category. In section 4 we describe the edge features and their computation. In sections 5 and 6 we describe the algorithms for learning and detection in detail. In sections 7 and 8 we present an experimental evaluation. Finally, in sections 9, 10 and 11, we analyze the results, suggest directions for future study and draw conclusions.

2 Related Work

In the proposed approach the target object category is represented by sparse, flexible configurations of oriented edges. Several other works have relied on similar representations. For example, Fleuret and Geman learn a disjunction of oriented edge configurations, where all oriented edges of at least one configuration are required to be present within a user-specified tolerance in order for detection to occur [13]. Wu et al. learn a configuration of Gabor basis functions, where each basis function is allowed to move within a user-specified distance from its preferred position in order to maximize its response [14]. The most similar representation is probably from Danielsson et al. [15], who use a combination of sparse configurations of oriented edges in a voting scheme for object detection. In their work, however, the number of configurations used to represent a class, the number of edges in each configuration and the flexibility tolerance of edges are all defined by the user and not learned automatically. They also use different methods for learning and detection.

We argue that our representation essentially generalizes these representations. For example, by constraining the weak classifiers to use only a single edge feature we get a representation similar to that of Wu et al [14]. The proposed approach also extends the mentioned methods because the tolerance in edge location is learnt rather than defined by the user. Furthermore, being discriminative rather than generative, it can naturally learn tolerance thresholds with negative as well as positive parity, ie. it can also encode that an edge of a certain orientation is *un*likely to be within a certain region.

The use of a cascade of increasingly complex classifiers to quickly reject obvious negatives was pioneered by Viola and Jones [16, 11]. The cascade also provides a means of focusing the negative training set on relevant examples during training. Each stage in the cascade contains an AdaBoost classifier, which is a linear combination of weak classifiers. Viola and Jones constructed weak classifiers by thresholding single Haar features, whereas our weak classifiers are *flexible configurations of* oriented edge features.

3 Object Class Representation

The key component of the object representation is the weak classifiers. A weak classifier can be regarded as a conjunction of a set of single feature classifiers, where a single feature classifier is defined by an edge feature (a location and orientation) along with a tolerance threshold and its parity. A single feature classifier returns true if the distance from the specified location to the closest edge with the specified orientation is within tolerance (i.e. it should be sufficiently small if the parity is positive and sufficiently large if the parity is negative). A weak classifier returns true if all its constituent single feature classifiers return true. A strong classifier is then formed by boosting the weak classifiers. Figure 2 illustrates how the output of the strong classifier is computed for two different examples of the target class. The output of the strong classifier is thresholded

to determine class membership. The edge features will be described in the next section and the weak classifiers in section 5.1.

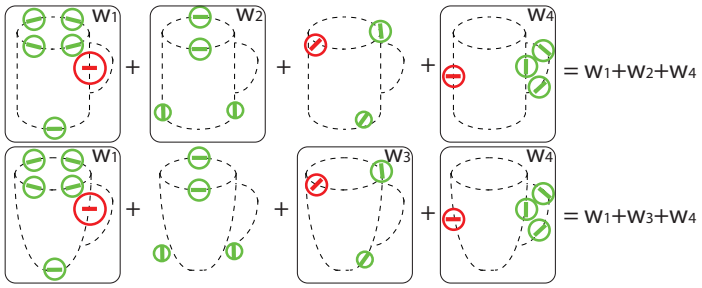


Fig. 2. *An object class is represented by a strong classifier, which is a linear combination of weak classifiers. Each weak classifier is a conjunction of single feature classifiers. Each single feature classifier is described by a feature (bars), a distance threshold (circles) and a parity of the threshold (green = pos. parity, red = neg. parity). The output of the strong classifier is the sum of the weights corresponding to the "active" weak classifiers, as illustrated for two different examples in the figure.*

4 Edge Features

An edge feature defines a single feature classifier together with a threshold and its parity. In this section we define the edge feature and the feature value. We also describe how to compute feature values efficiently.

An edge feature $F_k = (\mathbf{x_k}, \theta_k)$ is defined by a location $\mathbf{x_k}$ and orientation θ_k in a normalized coordinate system. The feature value, f_k, is the (normalized) distance to the closest image edge with similar orientation (two orientations are defined as similar if they differ by less than a threshold, t_θ). In order to emphasize that feature values are computed by aligning the normalized feature coordinate system with an image, I, using translation $\mathbf{t}$ and scaling s, we write $f_k\left(I, \mathbf{t}, s\right)$. See figure 3 for an illustration.

In order to define $f_k\left(I, \mathbf{t}, s\right)$, let $\mathcal{E}(I) = \{\ldots, (\mathbf{p}', \theta'), \ldots\}$ be the set of oriented edge elements (or edgels) in image I ($\mathcal{E}(I)$ is simply computed by taking the output of any edge detector and appending the edge orientation at each edge point) and let $\mathcal{E}_\theta(I) = \left\{(\mathbf{p}', \theta') \in \mathcal{E}(I) \middle| \left|\cos(\theta' - \theta)\right| \geq \cos(t_\theta)\right\}$ be the set of edgels with orientation similar to θ. We can then define $f_k\left(I, \mathbf{t}, s\right)$ as:

$$f_k\left(I, \mathbf{t}, s\right) = \frac{1}{s} \cdot \min_{(\mathbf{p}', \theta') \in \mathcal{E}_{\theta_k}(I)} ||s \cdot \mathbf{x_k} + \mathbf{t} - \mathbf{p}'|| \tag{1}$$

Typically we define a set of features $\mathcal{F}$ by constructing a uniformly spaced grid in the normalized coordinate system, i.e. $\mathcal{F} = \mathcal{X} \times \mathcal{Y} \times \Theta$, where $\mathcal{X}$, $\mathcal{Y}$ and Θ are uniformly spaced points.

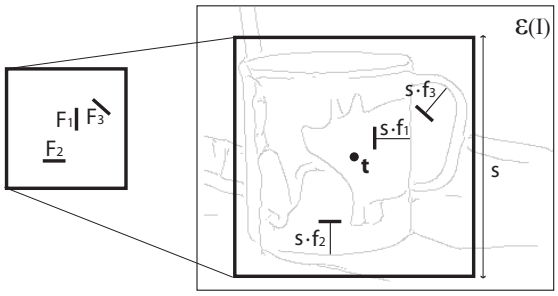


Fig. 3. *Feature values $f_k(I, \mathbf{t}, s)$ are computed by translating ($\mathbf{t}$) and scaling (s) features $F_k = (\mathbf{x_k}, \theta_k)$ and taking the (normalized) distances to the closest edges with similar orientations in the image.*

4.1 Efficient Computation of Feature Values

The feature value $f_k(I, \mathbf{t}, s)$ defined in the previous section can be computed using only a single array reference and a few elementary operations. The required preprocessing is as follows.

Start by extracting edges $\mathcal{E}(I)$ from the input image I. This involves running an edge detector on the image and then computing the orientation of each edge point. The orientation is orthogonal to the image gradient and unsigned, i.e. defined modulo π. For edge points belonging to an edge segment, we compute the orientation by fitting a line to nearby points on the segment. We call the elements of $\mathcal{E}(I)$ edgels (edge elements).

The second step of preprocessing involves splitting the edgels into several overlapping subsets $\mathcal{E}_\theta(I)$, one subset for each orientation $\theta \in \Theta$ (as defined in the previous section). Then compute the distance transform $d_\theta(I, \mathbf{p})$ on each subset [12]:

$$d_\theta(I, \mathbf{p}) = \min_{(\mathbf{p}', \theta') \in \mathcal{E}_\theta(I)} ||\mathbf{p} - \mathbf{p}'|| \tag{2}$$

This step is illustrated in figure 4; in the first column of the figure the subsets corresponding to horizontal and vertical orientations are displayed as feature maps and in the last column the corresponding distance transforms $d_\theta(I, \mathbf{p})$ are shown. Feature values can be efficiently computed as: $f_k(I, \mathbf{t}, s) = d_{\theta_k}(I, s \cdot \mathbf{x_k} + \mathbf{t})/s$. This requires only one array reference and a few elementary operations.

5 Learning

The basic component of our object representation is the weak classifier (section 5.1). It is boosted to form a strong classifier (section 5.2). Finally, a cascade of strong classifiers is built (section 5.3).

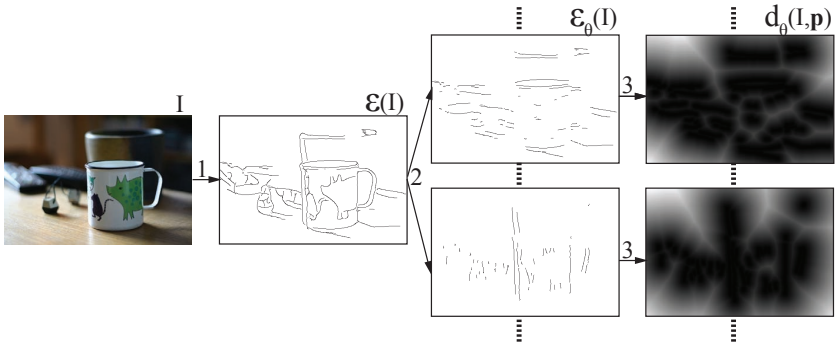


Fig. 4. *Images are preprocessed as follows: (1) extract a set of oriented edges $\mathcal{E}(I)$, (2) split $\mathcal{E}(I)$ into (overlapping) subsets $\mathcal{E}_\theta(I)$ and (3) compute the distance transforms $d_\theta(I, \mathbf{p})$.*

5.1 Learning the Weak Classifiers

In this section we describe how to learn a weak classifier given a weighted set of training examples. The goal of the learner is to find a weak classifier with minimal (weighted) classification error on the training set.

The input to the training algorithm is a set of training examples $\{I_j | j \in \mathcal{J}\}$ with target class $c_j \in \{0, 1\}$ and a weight distribution, $\{d_j | j \in \mathcal{J}\}$, $\sum_{j \in \mathcal{J}} d_j = 1$, over these examples. Each training image is annotated with the centroid $\mathbf{t}_j$ and scale s_j of the object in the image.

A weak classifier is the conjunction of a set of single feature classifiers. A single feature classifier is defined by a feature, F_k, a threshold, t, and a parity, $p \in \{-1, 1\}$. The single feature classifier has the classification function $g(I, \mathbf{t}, s) = p \cdot f_k(I, \mathbf{t}, s) \leq p \cdot t$ and the weak classifier thus has the classification function $h(I, \mathbf{t}, s) = \bigwedge_{i=1}^{N} g_i(I, \mathbf{t}, s) = \bigwedge_{i=1}^{N} (p_i \cdot f_{k_i}(I, \mathbf{t}, s) \leq p_i \cdot t_i)$. The number of single feature classifiers, N, is learnt automatically.

We want to find the weak classifier that minimizes the classification error, e, wrt. the current importance distribution:

$$e = P_{j \sim \mathbf{d}}(h(I_j, \mathbf{t}_j, s_j) \neq c_j) = \sum_{\{j \in \mathcal{J} | h(I_j, \mathbf{t_j}, s_j) \neq c_j\}} d_j \tag{3}$$

We use a greedy algorithm to incrementally construct a weak classifier. In order to define this algorithm, let $h_n(I, \mathbf{t}, s) = \bigwedge_{i=1}^{n} g_i(I, \mathbf{t}, s)$ be the conjunction of the first n single feature classifiers. Then let $\mathcal{J}_n = \{j \in \mathcal{J} | h_n(I_j, \mathbf{t_j}, s_j) = 1\}$ be the indices of all training examples classified as positive by h_n and let e_n be the classification error of h_n:

$$e_n = \sum_{\{j \in \mathcal{J} | h_n(I_j, \mathbf{t_j}, s_j) \neq c_j\}} d_j \tag{4}$$

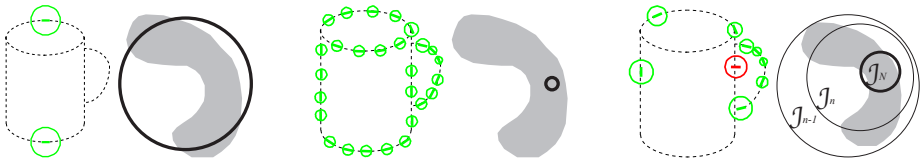
We can write e_n in terms of e_{n-1} as:

$$e_n = e_{n-1} + \sum_{\{j \in \mathcal{J}_{n-1} | g_n(I_j, \mathbf{t_j}, s_j) \neq c_j\}} d_j - \sum_{\{j \in \mathcal{J}_{n-1} | c_j = 0\}} d_j \tag{5}$$

This gives us a recipe for learning the nth single feature classifier:

$$g_n = \arg\min_g \sum_{\{j \in \mathcal{J}_{n-1} | g(I_j, \mathbf{t_j}, s_j) \neq c_j\}} d_j \tag{6}$$

We keep g_n if $e_n < e_{n-1}$, which can be evaluated easily using equation 5, otherwise we let $N = n - 1$ and stop. The process is illustrated in figure 5: in a) the weak classifier is not sufficiently specific to delineate the class manifold well, in b) the the weak classifier is too specific and thus inefficient and prone to overfitting, whereas in c) the specificity of the weak classifier is tuned to the task at hand.



(a) Too generic (b) Too specific (c) Optimal

Fig. 5. *The number of single feature classifiers controls the specificity of a weak classifier. a) Using too few yields weak classifiers that are too generic to delineate the class manifold (gray). b) Using too many yields weak classifiers that are prone to overfitting and that are inefficient at representing the class. c) We learn the optimal number of single feature classifiers for each weak classifier (the notation is defined in the text).*

Learning a Single Feature Classifier Learning a single feature classifier involves selecting a feature F_k from a "dictionary" $\mathcal{F}$, a threshold t from $\mathbb{R}^+$ and a parity p from $\{-1, 1\}$. An optimal single feature classifier, that minimizes equation (6), can be found by evaluating a finite set of candidate classifiers [17]. However, typically a large number of candidate classifiers have to be evaluated, leading to a time consuming learning process. Selecting the unconstrained optimum might also lead to overfitting.

We have observed empirically that our feature values (being nonnegative) tend to be exponentially distributed. This suggests selecting the threshold t for a particular feature as the intersection of two exponential pdfs, where μ^+ is the (weighted) average of the feature values from the positive examples and μ^- is the (weighted) average from the negative examples.

$$t = \ln\left(\frac{\mu^-}{\mu^+}\right) \cdot \frac{\mu^- \mu^+}{\mu^- - \mu^+} \tag{7}$$

The parity is 1 if $\mu^+ \leq \mu^-$ and -1 otherwise. We thus have only one possible classifier for each feature and can simply loop over all features and select the classifier yielding the smallest value for equation (6). This method for single feature classifier selection yielded good results in evaluation and we used it in our experiments.

5.2 Learning the Strong Classifier

We use asymmetric AdaBoost for learning the strong classifiers at each stage of the cascade [16]. This requires setting a parameter k specifying that false negatives cost k times more than false positives. We empirically found $k = 3n_-/n_+$ to be a reasonable choice. Finally the detection threshold of the strong classifier is adjusted to yield a true positive rate (tpr) close to one.

5.3 Learning the Cascade

The cascade is learnt according to Viola and Jones [11]. After the construction of each stage we run the current cascade on a database of randomly selected negative images to acquire a negative training set for the next stage. It is important to use a large set of negative images and in our experiments we used about 300.

6 Detection

6.1 Sliding Window

The simplest and most direct search strategy is the sliding window approach in scale-space. Since the feature values are computed quickly and we are using a cascaded classifier, this approach yields an efficient detector. Typically we use a multiplicative step-size of 1.25 in scale, so that $s_{i+1} = 1.25 \cdot s_i$, and an additive step-size of $0.01 \cdot s$ in position, so that $x_{i+1} = x_i + 0.01 \cdot s$ and $y_{i+1} = y_i + 0.01 \cdot s$. Non-maximum suppression is used to remove overlapping detections.

There are some drawbacks with the sliding window approach: (i) all possible windows have to be evaluated, (ii) the user is required to select a step-size for each search space dimension, (iii) the detector has to be made invariant to translations and scalings of at least half a step-size (typically by translating and scaling all training exemplars) and (iv) the output detections have an error of about half a step-size in each dimension of the search space (if the step-sizes are sufficiently large to yield an efficient detector these errors typically translate to significant errors in the estimated bounding box).

Since the feature values f_k represent distances, we can easily compute bounds $f_k^{(u)}$ and $f_k^{(l)}$, given a region $\mathcal{S}$ in search space, such that $f_k^{(l)} \leq f_k(I, \mathbf{t}, s) \leq f_k^{(u)} \forall (\mathbf{t}, s) \in \mathcal{S}$. This can be used to for efficient search space culling and should enable detectors with better precision at the same computational cost. We will explore the issue of hierarchical search in future work.

6.2 Estimating the Aspect Ratio

The detector scans the image over position and scale, but in order to produce a good estimate of the bounding box of a detected object we also need the aspect ratio (which typically varies significantly within an object class). We estimate the aspect ratio of a detected object by finding one or a few similar training exemplars and taking the average aspect ratio of these exemplars as the estimate. We retrieve similar training exemplars using the weak classifiers that are turned "on" by the detected object. Each weak classifier, h_k, is "on" for a subset of the positive training exemplars, $\mathcal{T}_k = \{x|h_k(x) = 1\}$ and by requiring several weak classifiers to be "on" we get the intersection of the corresponding subsets: $\mathcal{T}_{k_1} \cap \mathcal{T}_{k_2} = \{x|h_{k_1}(x) = 1 \wedge h_{k_2}(x) = 1\}$. Thus we focus on a successively smaller subset of training exemplars by selecting weak classifiers that fire on the detected object, as illustrated in figure 6. We are searching for a non-empty subset of minimal cardinality and the greedy approach at each iteration is to specify the weak classifier that gives maximal reduction in the cardinality of the "active" subset (under the constraint that the new "active" subset is non-empty).

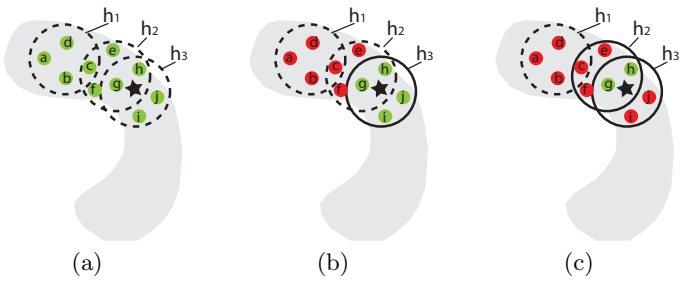


Fig. 6. *Retrieving similar training exemplars for a detected object (shown as a star). a) Weak classifiers h_2 and h_3 fire on the detection. b) Weak classifier h_3 is selected, since it gives a smaller "active" subset than h_2. c) Weak classifier h_2 is selected. Since no more weak classifiers can be selected, training exemplars "g" and "h" are returned.*

7 Experiments and Dataset

We have evaluated object detection performance on the ETHZ Shape Classes dataset [1]. This dataset is challenging due to large intra-class variation, clutter and varying scales. Several other authors have evaluated their methods on this dataset: in [5] a deformable shape model learnt from images is evaluated, while hand drawn models are used in [18, 19, 1, 20]. More recent results are [21, 22, 2], where [2] reports the best performance.

Experiments were performed on all classes in the dataset and results are produced as in [5], using 5-fold cross-validation. We build 5 different detectors

for each class by randomly sampling 5 subsets of half the images of that class. All other images in the dataset are used for testing. The dataset contains a total of 255 images and the number of class images varies from 32 to 87. Thus, the number of training images will vary from 16 to 43 and the test set will consist of about 200 background images and 16 to 44 images containing occurrences of the target object.

Quantitative results are presented as the detection rate (the number of true positives divided by the number of occurrences of the target object in the test set) versus the number of false positives per image (FPPI). We prefer using FPPI, instead of precision, as a measure of error rate, since it is not biased by the number of positive and negative examples in the test set. Precision would for example be unsuitable for comparison to methods evaluated using a different cross-validation scheme, since this might affect the number of positive test examples. As in [5], a detection is counted as correct if the detected bounding box overlaps more than 20 % with the ground truth bounding box. Bounding box overlap is defined as the area of intersection divided by the area of union of the bounding boxes.

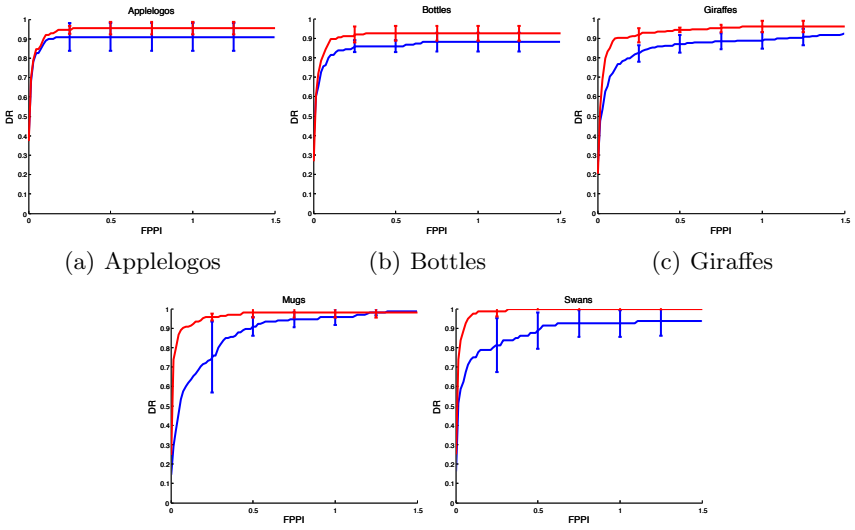
8 Results

Quantitative results are plotted in figure 7. The results of the initial experiment are shown by the blue (lower) curves. The problem is that there are very few training examples (from 16 to 43). Since the presented method is statistical in nature it is not expected to perform well when training on a too small training set. Therefore we downloaded a set of extra training images from Google Images. These images contained a total of 40 applelogos, 65 bottles, 160 giraffes, 67 mugs and 103 swans, which were used to extend the training sets. The red (upper) curves are the results using the extended training set and the detection performance is now almost perfect.

In table 1 we compare the detection performance of our system to previously presented methods. Figure 8 shows some sample detections (true and false positives).

Table 1. Comparison of detection performance. We state the average detection rate at 0.3 and 0.4 FPPI and its standard deviation in parentheses. We compare to the systems of [5, 2].

	A. logos	Bottles	Giraffes	Mugs	Swans
ours@0.3 FPPI:	95.5(3.2)	91.9(4.1)	92.9(1.9)	96.4(1.4)	98.8(2.8)
ours@0.4 FPPI:	95.5(3.2)	92.6(3.7)	93.3(1.6)	97.0(2.1)	100(0)
[5]@0.4 FPPI:	83.2(1.7)	83.2(7.5)	58.6(14.6)	83.6(8.6)	75.4(13.4)
[2]@0.3 FPPI:	95.0	92.9	89.6	93.6	88.2
[2]@0.4 FPPI:	95.0	96.4	89.6	96.7	88.2



(a) Applelogos (b) Bottles (c) Giraffes

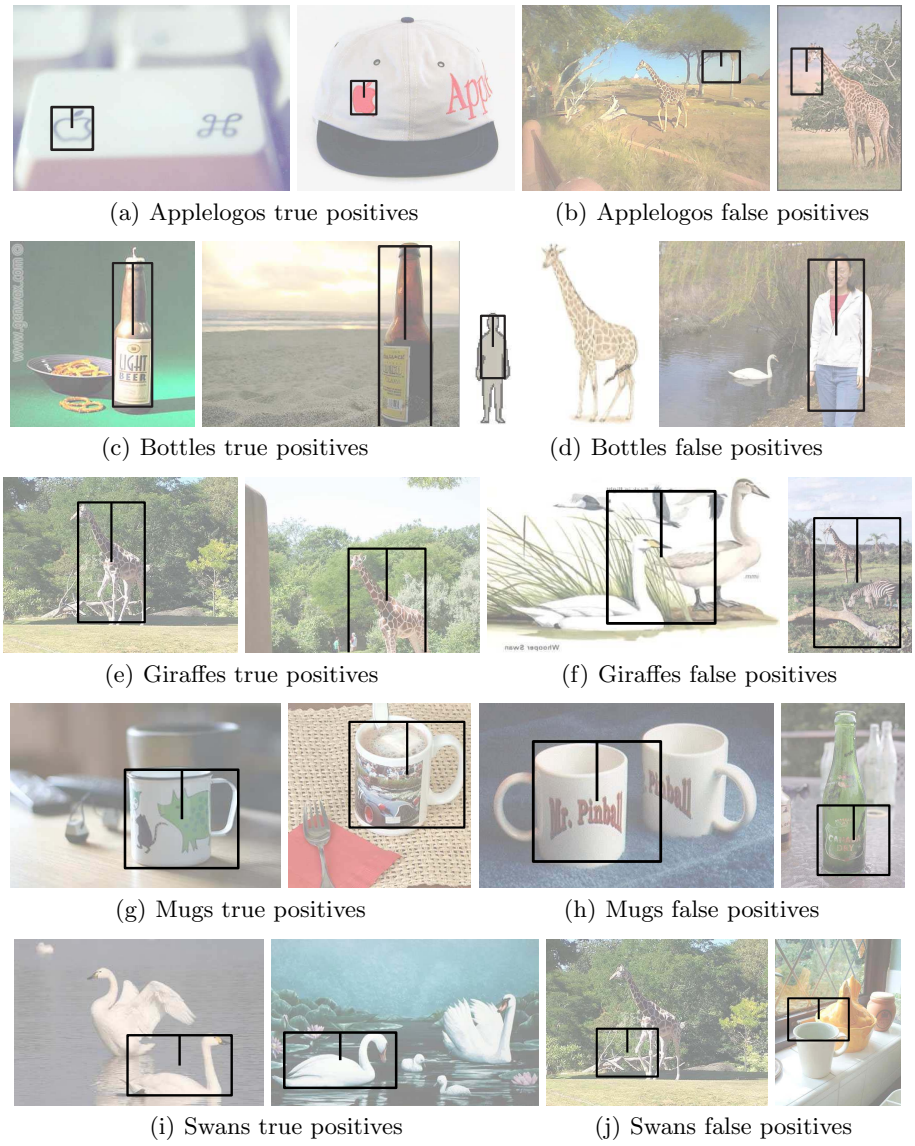
(d) Mugs (e) Swans

Fig. 7. *Detection rate (DR) plotted versus false positives per image (FPPI). The results from the initial experiment are shown by the blue (lower) curves and the results from the experiment using extended training sets are shown by the red (upper) curves.*

Our currently unoptimized MATLAB/mex implementation of the sliding window detector, running (a single thread) on a 2.8 GHz Pentium D desktop computer, requires on the order of 10 seconds for scanning a 640 x 480 image (excluding the edge detection step). We believe that this could be greatly improved by making a more efficient implementation. This algorithm is also very simple to parallelize, since the computations done for one sliding window position are independent of the computations for other positions.

9 Discussion

Viola and Jones constructed their weak classifiers by thresholding single Haar-features in order to minimize the computational cost of weak classifier evaluation [16]. In our case, it is also a good idea to put an upper limit on the number of features used by the weak classifiers in the first few stages of the cascade for two reasons: (1) it reduces the risk of overfitting and (2) it improves the speed of the learnt detector. However, a strong classifier built using only single-feature weak classifiers should only be able to achieve low false positive rates for object classes with small intra-class variability. Therefore later stages of the cascade should not limit the number of features used by each weak classifier (rather, the learner should be unconstrained).



(a) Applelogos true positives (b) Applelogos false positives

(c) Bottles true positives (d) Bottles false positives

(e) Giraffes true positives (f) Giraffes false positives

(g) Mugs true positives (h) Mugs false positives

(i) Swans true positives (j) Swans false positives

Fig. 8. *Example detections (true and false positives) for each class.*

10 Future Work

As mentioned, search space culling and hierarchical search strategies will be explored, ideally with improvements in precision and speed as a result.

Another straight forward extension of the current work is to use other features in addition to oriented edges. We only need to be able to produce a binary feature map, indicating the occurrences of the feature in an image. For example we could use corners, blobs or other interest points. We could also use the visual words in a visual code book (bag-of-words) as features. This would allow modeling of object appearance and shape in the same framework, but would only be practical for small code books.

We will investigate thoroughly how the performance of the algorithm is affected by constraining the number of features per weak classifier and we will also investigate the parameter estimation method presented in section 6.2 and the possibility of using it for estimating other parameters of a detected object, like the pose or viewpoint.

11 Conclusions

In this paper we have presented a novel shape-based object class representation. We have shown experimentally that this representation yields very good detection performance on a commonly used database. The advantages of the presented method compared to its competitors are that (1) it is generic and does not make any strong assumptions about for example the existence of discriminative parts or the detection of connected edge segments, (2) it is fast and (3) it is able to represent categories with significant intra-class variation. The method can also be extended to use other types of features in addition to the oriented edge features. The main drawback of the method is that, since its inductive bias is rather weak, it needs a comparatively large training set (at least about 100 training exemplars for the investigated dataset).

Acknowledgement

This work was supported by the Swedish Foundation for Strategic Research (SSF) project VINST.

References

1. Ferrari, V., Tuytelaars, T., Gool, L.V.: Object detection by contour segment networks. Proc. of the European Conference on Computer Vision (2006)
2. Maji, S., Malik, J.: Object detection using a max-margin. Proc. of the IEEE Computer Vision and Pattern Recognition (2009)
3. Gavrila, D.M.: A bayesian, exemplar-based approach to hierarchical shape matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **29** (2007)

4. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 509–522
5. Ferrari, V., Jurie, F., Schmid, C.: Accurate object detection with deformable shape models learnt from images. Proc. of the IEEE Computer Vision and Pattern Recognition (2007)
6. Thuresson, J., Carlsson, S.: Finding object categories in cluttered images using minimal shape prototypes. Proc. of Scandinavian Conference on Image Analysis (2003)
7. Carlsson, S.: Order structure, correspondence and shape based categories. International Workshop on Shape, Contour and Grouping (1999)
8. Shotton, J., Blake, A., Cipolla, R.: Contour-based learning for object detection. Proc. of the International Conference of Computer Vision (2005)
9. Opelt, A., Pinz, A., Zisserman, A.: A boundary-fragment model for object detection. Proc. of the European Conference of Computer Vision (2006)
10. Freund, Y., Schapire, R.E.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence **14** (1999) 771–780
11. Viola, P.A., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision **57** (2004) 137–154
12. Breu, H., Gil, J., Kirkpatrick, D., Werman, M.: Linear time euclidean distance transform algorithms. IEEE Trans. Pattern Analysis and Machine Intelligence **17** (1995) 529–533
13. Fleuret, F., Geman, D.: Coarse-to-fine face detection. International Journal of Computer Vision **4** (2001) 85 – 107
14. Wu, Y.N., Si, Z., Fleming, C., Zhu, S.C.: Deformable template as active basis. Proc. of the International Conference on Computer Vision (2007)
15. Danielsson, O., Carlsson, S., Sullivan, J.: Automatic learning and extraction of multi-local features. Proc. of the International Conference on Computer Vision (2009)
16. Viola, P.A., Jones, M.J.: Fast and robust classification using asymmetric adaboost and a detector cascade. Proc. of Neural Information Processing Systems (2001) 1311–1318
17. Fayyad, U.M.: On the Induction of Decision Trees for Multiple Concept Learning. PhD thesis, The University of Michigan (1991)
18. Ravishankar, S., Jain, A., Mittal, A.: Multi-stage contour based detection of deformable objects. Proc. of the European Conference on Computer Vision (2008)
19. Zhu, Q., Wang, L., Wu, Y., Shi, J.: Contour context selection for object detection: A set-to-set contour matching approach. Proc. of the European Conference on Computer Vision (2008)
20. Schindler, K., Suter, D.: Object detection by global contour shape. Pattern Recognition **41** (2008) 3736–3748
21. Stark, M., Goesele, M., Schiele, B.: A shape-based object class model for knowledge transfer. Proc. of International Conference on Computer Vision (2009)
22. Ommer, B., Malik, J.: Multi-scale object detection by clustering lines. Proc. of International Conference on Computer Vision (2009)