Generic Object Class Detection using Feature Maps

Oscar Danielsson and Stefan Carlsson

CVAP/CSC, KTH,
Teknikringen 14, S-100 44 Stockholm, Sweden
{osda02,stefanc}@csc.kth.se

Abstract. In this paper we describe an object class model and a detection scheme based on feature maps, i.e. binary images indicating occurrences of various local features. Any type of local feature and any number of features can be used to generate feature maps. The choice of which features to use can thus be adapted to the task at hand, without changing the general framework. An object class is represented by a boosted decision tree classifier (which may be cascaded) based on normalized distances to feature occurrences. The resulting object class model is essentially a linear combination of a set of flexible configurations of the features used. Within this framework we present an efficient detection scheme that uses a hierarchical search strategy. We demonstrate experimentally that this detection scheme yields a significant speedup compared to sliding window search. We evaluate the detection performance on a standard dataset [7], showing state of the art results. Features used in this paper include edges, corners, blobs and interest points.

Keywords: detector, AdaBoost, decision tree, distance transform, SIFT

1 Introduction and Related Works

Object class modeling and detection is a difficult problem. Often the intra-class variation is significant and the background class is extremely large (i.e. all image patches not containing an object of the target class), so the decision boundary required to separate the positive and negative classes in feature space will generally be complex. To represent a complex decision boundary, we need a powerful classifier/model. However, such classifiers are in general expensive to evaluate. This is a problem because at the detection stage we will need to evaluate the classifier/model for a very large number of subregions of the test image.

To solve this problem Viola and Jones proposed using a cascade of increasingly complex AdaBoost classifiers [20]. The complex classifiers at the upper stages of the cascade are then only evaluated on a small subset of the patches in the test image. In addition they proposed integral images to make the computation of Haar features extremely efficient. They combined these two techniques to build an accurate, real-time face detector. However their method has a limited ability to handle intra-class variation, mainly due to the Haar features not being

robust to intra-class variation. Therefore Laptev exchange the Haar features for histogram features (which can be efficiently computed using integral histograms) and can thus handle more difficult classes than faces [14]. The reason being that the histogram features are more robust to intra-class variation and therefore the target class gets a more compact distribution in feature space. Felzenszwalb et. al. then take one step further and introduce deformable part models into the cascade to get an even more flexible classifier, which has shown good performance in the popular Pascal challenge [5, 4].

We see that these methods handle increasingly difficult target classes by using features that are increasingly robust to intra-class variation, while maintaining computational efficiency. In the present paper we continue this line of research and propose a generic framework that takes feature maps as input. The number of feature maps and the methods used to generate them is not specified in the framework and can thus be adapted to the task at hand. We use an AdaBoost classifier (with decision trees as weak classifiers) that we cascade to minimize computations on obvious negatives. The basic image measurements used by our classifier are distances to feature occurrences. Therefore we can define an efficient hierarchical search that gives a significant speedup compared to the sliding window approach.

Hierarchical search schemes have been used previously minimize the Chamfer distance between a search template and a test image [1]. A very large number of templates are needed to represent an object class with significant intra-class variation. Gavrila has devised a search scheme that is hierarchical in both search space and in template space [11]. While the hierarchical search is a desirable property of the Chamfer matching methods, the template-based representation of an object class is not. The problem is that there is a big risk that even a very large set of templates does not represent the whole target class (overfitting). The use of a strong classifier is better in this sense, since most classifiers have been designed to have a good ability to generalize beyond the training set. For example, if weak classifiers can perform better than chance on every distribution over the training set, AdaBoost can provably achieve arbitrarily good generalization bound [8].

In summary our method (1) has good generalization properties (inherited from the AdaBoost procedure), (2) allows for a very fast hierarchical search and (3) allows the user to adapt the choice of image features to the task at hand.

The rest of the paper is organized as follows. In section 2 we describe how our method represents the object category and how this representation is learnt. In section 3 we describe the detection algorithm in detail. In section 4 we present experiments evaluating the detection performance and computational efficiency of our method. Finally, we conclude in section 5.

2 Object Class Model

In this section we describe how an object class is modeled and how the parameters of that model are learnt. The target object class is represented by a

boosted decision tree classifier based on normalized distances to feature occurrences. The classifier can be visualized as a linear combination of flexible feature configurations, described in a normalized coordinate system.

We start by defining some notation. We then describe the classifier and how it is learnt in sections 2.1 to 2.4. Finally, we mention variations to the learning algorithm in section 2.5.

We assume that we have a set of features $\mathcal{K}$, for which we can compute feature maps, $\Phi_k(I, \mathbf{x}) \in \{0, 1\}$ that returns 1 if feature k occurs at location $\mathbf{x}$ in image I and 0 otherwise. We will also make use of distance transforms of feature maps: $d_k(I, \mathbf{x}) = \min_{\{\mathbf{x}' | \Phi_k(I, \mathbf{x}')=1\}} ||\mathbf{x} - \mathbf{x}'||$. Distance transforms can be computed efficiently [2].

The basic building blocks of the classifier are localized features $F = (k, \mathbf{p})$, defined by the feature index k and the location $\mathbf{p}$ in a normalized coordinate system. We also define the feature value, $f(I, \mathbf{t}, s) = d_k(I, s \cdot \mathbf{p} + \mathbf{t})/s$, which is obtained by translating ($\mathbf{t}$) and scaling (s) the normalized coordinate system into an image (I) and computing the normalized distance to the closest occurrence of the feature in the image. Note that the computation of the feature value essentially only involves a lookup into the distance transform table. All feature values are nonnegative.

We define a dictionary $\mathcal{F} = \{F_n | n = 1 \dots N\} = \mathcal{K} \times \mathcal{P}$ of localized features, where $\mathcal{P}$ is a uniformly spaced grid in the normalized coordinate system. By concatenating the corresponding feature values, we get a feature vector $\mathbf{f}(I, \mathbf{t}, s) = [f_1(I, \mathbf{t}, s) \dots f_N(I, \mathbf{t}, s)]^T$.

The training data consists of a set of images $\{I_j | j \in \mathcal{J}\}$ and a set of annotations $\{(\mathbf{t}_i, s_i, j_i) | i \in \mathcal{I}\}$, specifying the location, scale and image number of each instance of the target class in the image set.

2.1 Cascade

The cascade is not really a part of the object model, but rather a sequence of object models of increasing detail and specificity. However, it serves two important functions: (1) it minimizes the number of computations spent on obvious negatives at the detection stage and (2) it provides a mechanism for selecting hard negative examples at the training stage. The cascade is learnt according to Viola and Jones [20]. Each stage of the cascade contains an object model, which is learnt using all annotated instances of the target class as positive examples. We gather negative examples by running the current cascade on all training images and sample false positive detections. We then compute feature vectors for all (positive and negative) training examples and pass that to the strong learner, which will be described in the next section. The strong learner outputs a classification function H, that is thresholded to determined class membership. The threshold is typically selected to give a specific true positive rate on a validation set.

2.2 Strong Classifier

In this section we describe the strong classifier. The strong classifier is a linear combination of weak classifiers learnt using a variant of AdaBoost. We give a generalized description of the boosting algorithm, closely following Schapire and Singer [18], in listing 1. The input to the learner is a set of feature vectors $\{\mathbf{f}^m\}$, with target class $c_m \in \{-1, 1\}$. The classification function of the strong classifier, $H(\mathbf{f}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{f})$, is a linear combination of the classification functions of the weak classifiers. The classification function is thresholded to determine class membership. We mention different alternatives for initializing and updating the weight distribution and for choosing the αs in section 2.5. In the next section, we describe the weak classifier.

Algorithm 1 Boosting

Require: $\{\mathbf{f}^m\}$, $c_m \in \{-1, 1\}$, T
  $\mathbf{d_1} \leftarrow$ initialize weight distribution
  for $t = 1$ to T **do**
    Train weak classifier $h_t : \mathbb{R}^{*N} \rightarrow \mathbb{R}$ using distribution $\mathbf{d_t}$
    Choose $\alpha_t \in \mathbb{R}$
    $\mathbf{d_{t+1}} \leftarrow$ update weight distribution
  end for
  return $\{\alpha_1, \ldots, \alpha_T\}$, $\{h_1, \ldots, h_T\}$

2.3 Weak Classifier

The weak classifier is a binary decision tree. The leaf nodes contain the outputs of the classifier and the internal nodes contain binary classifiers, which we will refer to as single feature classifiers (described in the next section). At the detection stage the output of the single feature classifier determines whether to visit the left or right subtree next; when a leaf node is reached, its output is returned.

A generalized description of the weak learner is given in listing 2. The input to the weak learner is a set of feature vectors $\{\mathbf{f}^m\}$, with target class $\{c_m\}$, and a weight distribution $\mathbf{d}$. The weak learner then computes the output of the current node and possibly constructs left and right subtrees recursively. We will mention different alternatives for computing the output of a node and for validating the split induced by a single feature classifier in section 2.5. In the next section we describe the single feature classifier.

2.4 Single Feature Classifier

A single feature classifier g consists of a single localized feature (selected from the dictionary) $F_n \in \mathcal{F}$, along with a distance threshold $t \in \mathbb{R}^+$ and its parity $p \in \{-1, 1\}$. The output of the single feature classifier is 1 if $p \cdot f_n \leq p \cdot t$ and -1 otherwise.

Algorithm 2 Weak learner

Require: $\{\mathbf{f}^m\}$, $\{c_m\}$, $\mathbf{d}$
  Node.output $\leftarrow$ compute output using $\{c_m\}$ and $\mathbf{d}$
  Train single feature classifier $g : \mathbb{R}^N \rightarrow \{-1, 1\}$ using $\{\mathbf{f}^m\}$, $\{c_m\}$ and $\mathbf{d}$
  Compute split $\mathcal{M}_- = \{m|g(\mathbf{f}^m) = -1\}$ and $\mathcal{M}_+ = \{m|g(\mathbf{f}^m) = 1\}$
  Stop $\leftarrow$ validate split using $\mathcal{M}_-$, $\mathcal{M}_+$,$\{c_m\}$ and $\mathbf{d}$
  if Stop **then**
    return Node
  end if
  Node.left = Weak learner($\{\mathbf{f}^m|m \in \mathcal{M}_-\}, \{c_m|m \in \mathcal{M}_-\}, \mathbf{d}$)
  Node.right = Weak learner($\{\mathbf{f}^m|m \in \mathcal{M}_+\}, \{c_m|m \in \mathcal{M}_+\}, \mathbf{d}$)
  return Node

Learning a single feature classifier involves selecting a feature n, a threshold t and a parity p. A generalized procedure for learning a single feature classifier is given in listing 3. 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 (the parity is 1 if $\mu^+ \leq \mu^-$ and -1 otherwise):

$$t = \ln\left(\frac{\mu^-}{\mu^+}\right) \cdot \frac{\mu^-\mu^+}{\mu^- - \mu^+} \tag{1}$$

Thus each localized feature in the dictionary yields a single threshold and parity. The remaining task is to select the feature that minimizes the error function. We will mention different error functions in the following section.

Algorithm 3 Learn single feature classifier

Require: $\{\mathbf{f}^m\}$, $\{c_m\}$, $\mathbf{d}$
  for $n = 1$ to N **do**
    $(t_n, p_n) \leftarrow$ select threshold and polarity using $\{f_n^m\}$, $\{c_m\}$ and $\mathbf{d}$
    $e_n \leftarrow$ compute error using $\{f_n^m\}$, $\{c_m\}$, $\mathbf{d}$ and (t_n, p_n)
  end for
  $n^* \leftarrow \arg\min_n e_n$
  return (n^*, t_{n^*}, p_{n^*})

2.5 Variations

In the previous sections we have given a generalized description of the classifier and how to learn it. However, there are several ways in which this general scheme can be varied and in this section we mention the most interesting variations, which will also be compared experimentally in section 4.

Firstly, we have the choice of whether or not to use asymmetric weighting, as described in [19]. This choice affects the initialization and update of the weight distribution in the strong learner. Using asymmetric weighting 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 in this case.

Secondly, we have the choice of whether to let the weak classifiers output binary or confidence rated predictions. This choice affects (1) the computation of the αs in the strong learner, (2) the computation of the output of a node in the weak learner and (3) the error that is minimized by the single feature learner. In the case of binary predictions we use the original AdaBoost algorithm of Freund and Schapire [8] to compute the αs. The output of a node is simply the weighted majority of the training examples and the error is the weighted training error. In the case of confidence rated predictions we follow Schapire and Singer's recipe for domain-partitioning hypotheses [18]. The αs are set to 1 in this case.

Finally, we can pose various constraints on the weak classifier. For example we can limit the depth of the decision tree to reduce the risk of over-fitting.

3 Detection

In this section we describe the detection procedure. It consists of three parts: (1) preprocessing, (2) scale space search and (3) aspect ratio estimation. The preprocessing is illustrated in figure 1 and entails computing feature maps and distance transforms for each feature. The scale space search can be done using a sliding window approach, however the features used in this paper allow a hierarchical search scheme with efficient search space culling to be defined. This is described in the next section. The scale space search yields the position and scale of detected objects. However, we want the bounding box, which also requires an aspect ratio. In section 3.2 we describe how to estimate the aspect ratio.

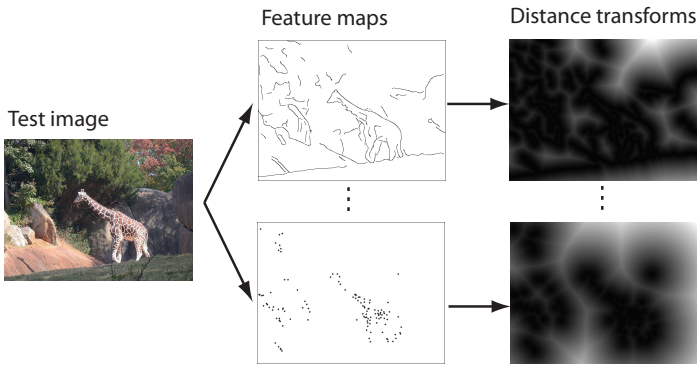


Fig. 1. *Images are preprocessed by computing feature maps and the corresponding distance transforms for each feature.*

3.1 Hierarchical Search

In this section we describe the hierarchical search in scale space. The idea is that, given a region in search space, we can compute bounds on the value of all localized features and if none of the possible values would yield a detection, we can discard the whole region.

We have previously defined the value of a localized feature $F = (k, \mathbf{p})$ to be $f(I, \mathbf{t}, s) = d_k(I, s \cdot \mathbf{p} + \mathbf{t})/s$. Now, if we have a cuboid region, S, in search space, we can compute upper and lower bounds for the feature value; i.e. we can compute $f^{(u)}$ and $f^{(l)}$ such that $f^{(l)} \leq f(I, \mathbf{t}, s) \leq f^{(u)} \forall (\mathbf{t}, s) \in S$.

Let B contain the 8 corner points of S and let $(\mathbf{t}_0, s_0)$ be any point in S (for example the centroid). Then let $P' = \{s \cdot \mathbf{p} + \mathbf{t} | (\mathbf{t}, s) \in B\}$, $\mathbf{p}'_0 = s_0 \cdot \mathbf{p} + \mathbf{t}_0$ and $d_{\max} = \max_{\mathbf{p}' \in P'} \|\mathbf{p}'_0 - \mathbf{p}'\|$. We can now compute upper and lower bounds as follows: $f^{(u)} = (d_k(I, \mathbf{p}'_0) + d_{\max})/s_1$ and $f^{(l)} = \max((d_k(I, \mathbf{p}'_0) - d_{\max})/s_2, 0)$, where s_1 and s_2 are the minimum and maximum scales in S respectively.

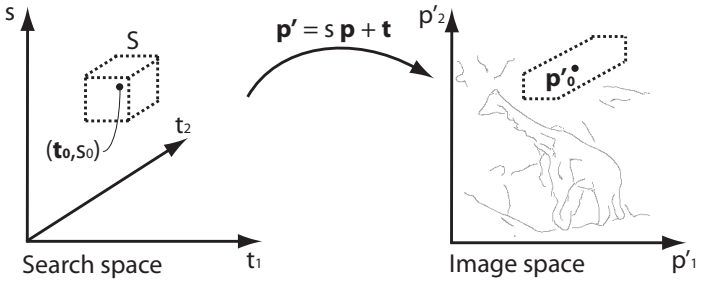


Fig. 2. *If a localized feature has position* $\mathbf{p}$ *in normalized coordinates and the normalized frame is aligned with an image by translation* $\mathbf{t_0}$ *and scaling* s_0*, the position of the feature in the image is* $\mathbf{p}' = s_0 \cdot \mathbf{p} + \mathbf{t_0}$*. However, if we have a whole range* S *of possible translations and scalings, the position of the localized feature in the image can be anywhere in the dashed region in image space. We can easily compute bounds for the feature value given that the position of the localized feature is within that region.*

The uncertainty in the feature value may yield an ambiguity in the output of the single feature classifier (i.e. it could be either 1 or -1). When evaluating the weak classifier we are then unable to decide whether to visit the left or right child node next. In such cases we pursue both paths and the output of the weak classifier is defined as the maximum of all leaf nodes that were reached. Thus we get an optimistic strong classifier that returns 1 if (but not only if) any point in the region, S, is a detection.

We are now ready to define the hierarchical search algorithm. The algorithm recursively partitions the search space into smaller regions, evaluating the classifier at each new region. If the classifier returns -1 for any region, that region is discarded. When the classifier returns 1, subdivision continues until the current

region is small enough; then the classifier is evaluated at the centroid of that region. A more detailed description is given in listing 4.

Algorithm 4 Hierarchical search

Require: Classifier c, Search region S
  if S is sufficiently small **then**
    $(\mathbf{t}, s) \leftarrow$ centroid of S
    result $\leftarrow$ evaluate classifier on $(\mathbf{t}, s)$
    if result = 1 **then**
      return $(\mathbf{t}, s)$
    else
      return $\emptyset$
    end if
  end if
  result $\leftarrow$ evaluate classifier on S
  if result = -1 **then**
    return $\emptyset$
  end if
  $[R_1, \ldots, R_l] \leftarrow$ split S into subregions
  initialize $D \leftarrow \emptyset$
  for all R_i **do**
    $D \leftarrow D \cup$ HierarchicalSearch(c, R_i)
  end for
  return D

3.2 Aspect Ratio Estimation

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 use regression to estimate the aspect ratio of a detected object. Specifically, we use gradient boosted regression trees [9]. The regressor is trained using set of feature vectors $\{\mathbf{f}^m\}$, with target aspect ratio a_m. We use the same training set for the aspect ratio estimator as for the detector (albeit the aspect ratio estimator only uses the positive examples). Each regression tree recursively splits the training examples in two and finally one estimate of the aspect ratio is assigned to each leaf node by optimizing some target function. Typically the target of the ensemble is to minimize the square norm of the residual and the target of each new regression tree is to correct the errors of the current ensemble.

At the detection stage the boosted regression trees are applied to the feature vector of each detected object to estimate its aspect ratio.

4 Experiments and Results

We have performed experiments on the ETHZ Shape Classes dataset [7]. This dataset is challenging due to large intra-class variation, clutter and varying scales. We used all images from the ETHZ dataset for testing only. Training images were downloaded from Google Images. These images contained a total of 106 applelogos, 128 bottles, 270 giraffes, 233 mugs and 165 swans. As in [6], 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. Several other authors have evaluated their methods on this dataset and we let [17] represent state-of-the-art.

The goal of our first experiment was to compare the different variants of the algorithm, as described in section 2.5. We use the giraffes as the test class, because it is the class with the most intra-class variation and thus the most difficult and realistic class. The results are given in figure 3. We vary one property at a time, starting with the choice of boosting algorithm. We compare discrete AdaBoost [8], real AdaBoost [18] and gentle AdaBoost [10]. The results, shown in figure 3(a), indicate that real AdaBoost is the best choice. We then experiment with the depth setting of the decision tree weak learner (figure 3(b)). We compare different set depths and an automatic version, where we stop growing the tree when further growth does not improve the classification error on the training set. We see that we should either set the depth to some small value, like one or two, or use the automatic version (which typically outputs very shallow trees). Then we experiment with different image features, first using only oriented edges [3] and then using also corners [12], blobs [15] and interest points (figure 3(c)). Interest points were detected using the Kadir-Brady detector [13]. For each interest point we compute the SIFT descriptor [16] and assign it to one out of eight different clusters which were computed using k-means on a set of interest points extracted from random background images. The interest points thus generate eight different feature maps - one for each cluster. We see that using more features improves the result. We finally tested the asymmetric weighting scheme [19], concluding that it improves the results (figure 3(d)).

We also evaluated the performance of the detector, using the settings from the previous experiment (i.e. real AdaBoost, automatic depth determination, all features and asymmetric weighting), on all other classes in the ETHZ dataset. The results are plotted in figure 4 and in table 1 we compare our results to some previous methods. We also show some example detections in figure 5.

Finally, we compare the runtime of the hierarchical search with the sliding window approach. Here we again use the giraffe class. Each test image is represented by a point in the scatter plot shown in figure 6, with the sliding window runtime on the x-axis and the hierarchical runtime on the y-axis. We see that on average the hierarchical search yields a 70-fold speed-up. Both algorithms were implemented in MATLAB/mex and executed on a 2.8 GHz Pentium D desktop computer (using a single core).

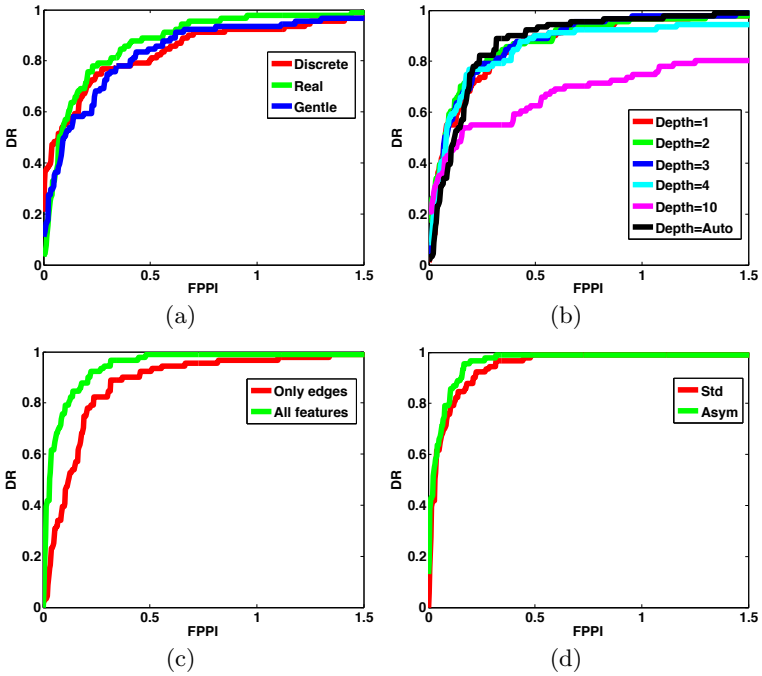


Fig. 3. *Comparison of different variants of the algorithm. See text for details. Best viewed in color.*

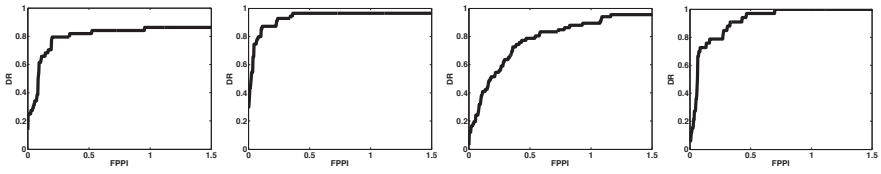
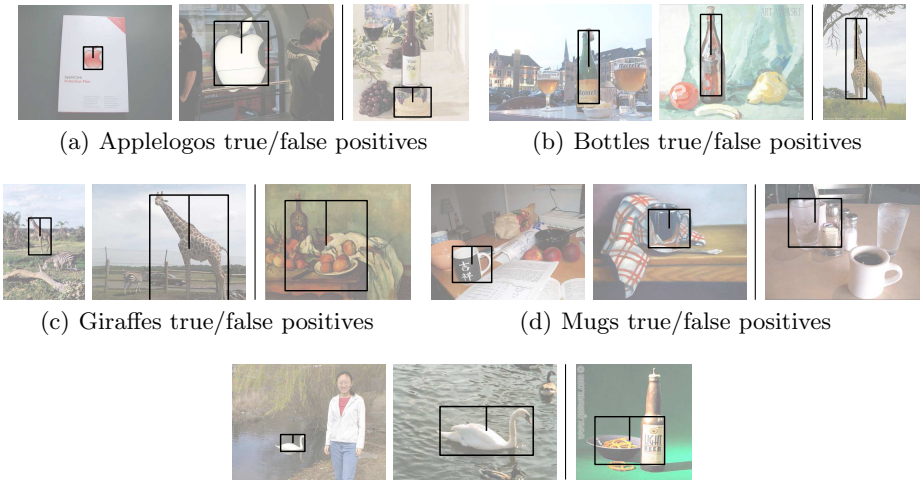


Fig. 4. *Detection rate (DR) plotted versus false positives per image (FPPI) for the remaining classes of the ETHZ dataset.*

Table 1. Comparison of detection performance. We state the detection rate at 0.4 FPPI. We compare to the systems of [6, 17].

	A. logos	Bottles	Giraffes	Mugs	Swans
ours@0.4 FPPI:	81.8	**96.4**	**98.9**	74.2	**90.9**
[6]@0.4 FPPI:	83.2	83.2	58.6	83.6	75.4
[17]@0.4 FPPI:	**95.0**	**96.4**	89.6	**96.7**	88.2



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

(c) Giraffes true/false positives (d) Mugs true/false positives

(e) Swans true/false positives

Fig. 5. *Example detections (true and false positives) for each class.*

5 Conclusion

In this paper we presented a framework for modeling and detecting visual object classes. The method is based on feature maps, which are computed by some external routine that is defined by the user. The learnt model of the object class is essentially a linear combination of a set of flexible spatial configurations of the input features. The advantages of the method is that it (1) has good generalization properties (inherited from the AdaBoost procedure), (2) allows for a very fast hierarchical search and (3) allows the user to adapt the choice of image features to the task at hand. We demonstrated these properties experimentally.

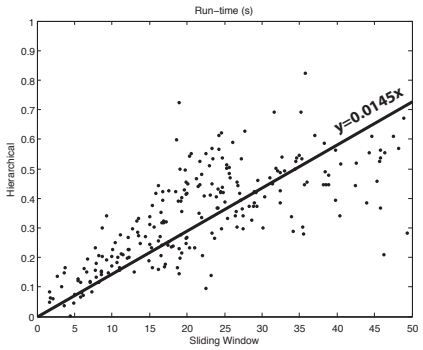


Fig. 6. *Comparison of the runtimes of the hierarchical search (y-axis) and the sliding window search (x-axis). Each point represents one test image.*

Acknowledgements This work was supported by The Swedish Foundation for Strategic Research in the project "Wearable Visual Information Systems".

References

1. Borgefors, G.: Hierarchical chamfer matching: A parametric edge matching algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 10(6), 849–865 (1988)
2. Breu, H., Gil, J., Kirkpatrick, D., Werman, M.: Linear time euclidean distance transform algorithms. IEEE Trans. Pattern Anal. and Mach. Intell. 17(5), 529–533 (1995)
3. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8, 679–714 (1986)
4. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html
5. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade object detection with deformable part models. IEEE Computer Vision and Pattern Recognition (2010)
6. Ferrari, V., Jurie, F., Schmid, C.: Accurate object detection with deformable shape models learnt from images. IEEE Computer Vision and Pattern Recognition (2007)
7. Ferrari, V., Tuytelaars, T., Gool, L.V.: Object detection by contour segment networks. Proc. of the European Conference on Computer Vision (2006)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer Systems and Sciences 55, 119–139 (1997)
9. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29(5), 1189–1232 (2001)
10. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics 28(2), 337–374 (2000)
11. Gavrila, D.M.: A bayesian, exemplar-based approach to hierarchical shape matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(8) (2007)
12. Harris, C., Stephens, M.: A combined corner and edge detector. Proc. of the 4th Alvey Vision Conference pp. 147–151 (1988)
13. Kadir, T., Zisserman, A., Brady, M.: An affine invariant salient region detector. Proc. of the European Conference on Computer Vision (2004)
14. Laptev, I.: Improving object detection with boosted histograms. Image and Vision Computing 27, 535–544 (2009)
15. Lindeberg, T.: Feature detection with automatic scale selection. International Journal of Computer Vision 30(2), 77–116 (1998)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
17. Maji, S., Malik, J.: Object detection using a max-margin hough transform. Proc. of the IEEE Computer Vision and Pattern Recognition (2009)
18. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, 297–336 (1999)
19. Viola, P.A., Jones, M.J.: Fast and robust classification using asymmetric adaboost and a detector cascade. Proc. of Neural Information Processing Systems (2001)
20. Viola, P.A., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)