Unsupervised Sub-categorization for Object Detection: Finding Cars from a Driving Vehicle

Rob G.J. Wijnhoven[1,2]
[1] ViNotion
Eindhoven, The Netherlands
rob.wijnhoven@vinotion.nl

Peter H.N. de With[2,3]
[2] University of Technology, Eindhoven, NL
[3] CycloMedia Technology,
Waardenburg, NL

Abstract

We present a novel algorithm for unsupervised sub-categorization of an object class, in the context of object detection. Dividing the detection problem into smaller sub-problems simplifies the object vs. background classification.

The algorithm uses an iterative split-and-merge procedure and uses both object and non-object information. Sub-classes are automatically split into two new classes if the visual variation is too large, while two classes are merged if they are visually similar. After each iteration, samples are relabeled to the most preferred subclasses.

In contrast to existing literature on unsupervised sub-categorization, our approach does not fix the number of final subclasses and determines this number using a visual similarity measure. Because we use a fast stochastic learning algorithm, full retraining and relabeling can be applied at each iteration. We show that the algorithm significantly outperforms state-of-the-art multi-class algorithms for a car detection problem using standard HOG features and simple linear classification, while significantly decreasing training time to a few minutes. Additionally, for our car detection problem, the identified subclasses by the algorithm were semantically meaningful and reveal the viewpoint of the object without the use of any motion information.

1. Introduction

Detection of an object-class is complicated because of the large variation in visual appearances of different object instances. Besides variations in illumination and contrast, strong visual variations are caused by differences in the instances of the object class (*e.g.* pickup truck vs. station wagon) and the changes in viewpoint.

Models for object detection use appearance and spatial information of the objects. Implicit Shape Models [10] store spatial information for a dictionary of visual features. Sliding window classifiers encode appearance information on a regular grid, thereby implicitly modeling spatial information. As an example of a sliding window detector, Dalal and Triggs proposed *Histogram of Oriented Gradients* (HOG) [3]. This feature transformation step (descriptor) adds invariance to illumination and contrast changes, but only partially to the intra-class and viewpoint variations. This means that the classification of the features into objects vs. background needs to take care of these variations.

The use of kernel machines like the Support Vector Machine (SVM), tackles this problem, but at the expense of a higher computational cost. This cost is linear to the number of support vectors, which is typically very high ($\gg 100$). Another solution to solve the variation is to split the object samples into multiple subsets, and use a separate classifier for each subset. Each subset is responsible for a specific range of viewpoints or intra-class variations.

Many algorithms solve the multi-view problem, but require a manual division of the object training samples into subclasses. Multi-view face detection is proposed by Schneiderman and Kanade [16], Jones and Viola [6], Wu *et al*. [21], Lin and Liu [13] and Huang *et al*. [5].

Other work embed 3D viewpoint knowledge in the learning phase, which enables estimation of the object viewpoint. Yan *et al*. [23] map multiple 2D views onto a 3D model using a homographic framework. *Partial Surface Models* (PSM) by Kushal *et al*. [9] link viewpoint-specific parts of objects with a graph. Liebelt *et al*. [12] apply a voting space using local features, extracted from object images, generated from several 3D models. Su *et al*. [18] learn a 3D model of objects and automatically improve this model using unannotated object images. Ozuysal *et al*. [14] estimate the bounding box and viewpoint of objects and perform verification with viewpoint-specific classifiers. Liebelt and Schmid [11] learn object appearance from viewpoint-annotated 2D images and learn the 3D relations from 3D models.

Manual annotation into subclasses is very time-consuming and can result in a non-optimal division of the samples with respect to detection performance. Further-

more, 3D models are not always available. Therefore, unsupervised sub-categorization is preferred. k-means clustering could be used to divide the training set, but includes background information in the clustering process. Approaches that do use this background information in the subcategorization process are *e.g*. *Cluster Boosted Tree* [22] and the algorithm by Kuo and Nevatia [8]. Most algorithms assume a fixed number of final subclasses and/or do not consider merging or splitting of classes. The performance of the resulting detection system is highly dependent on the number of subclasses, where the optimal number of subclasses is typically found by searching for the optimum in the range of classes.

Whereas many approaches for unsupervised subcategorization preset the number of subclasses as a fixed number, our algorithm automatically determines the number of subclasses depending on visual similarity. We show results on a multi-view car dataset, and show that the detectors trained by our approach significantly outperform Cluster Boosted Tree [22] and the approach of Kuo and Nevatia [8]. Training time was reduced from a week for CBT [22] and two days for [8] to only a few minutes.

The remainder of the paper is as follows. Related work is described is Section 2. Section 3 introduces the proposed algorithm. We evaluate the algorithm on the car-detection problem described in Section 4. Results are described in Section 5 and conclusions are drawn in Section 6.

2. Related work

In the class of unsupservised sub-categorization, we have found few publications closely related to our problem statement. Kuo and Nevatia [8] propose a two-stage approach by first clustering positive object samples into subcategories and then training a multi-class boosting classifier. The method outperforms Cluster Boosted Tree (CBT) on a multi-view car detection dataset. Shan *et al*. [17] select one so-called exemplar in each iteration, and then select discriminative features for this exemplar using boosting. The final classifier is a weighted sum of the per-exemplar classifiers, learned with a boosting framework. The training set is not directly divided into subclasses and the number of exemplars is fixed by the user. *MCBoost* by Kim and Cipolla [7] start from a (random) initialization in k classes. At each boosting iteration, weak classifiers are selected and the data samples are assigned to the classifier with highest response. Zhang and Zhang's *Winner-Take-All* WTA-McBoost [24] is conceptually simlar to MCBoost. Starting point is an initial division in k classes. Boosting is used to learn classifiers for the k subclasses. After P iterations of learning weak classifiers, samples are relabeled into the classifier with highest response. Yuan *et al*. [15] solve the problem with standard SVM training, using two kernel functions; one measuring similarity for object-vs-background and one taking care of intra-class variation. This slightly outperforms CBT but uses 560 mode detectors, resulting in a high computational cost during testing.

Conceptually most similar to the proposed algorithm are Tu's *Probabilistic Boosted Tree* [19] (PBT) and *Cluster Boosted Tree* (CBT) by Wu and Nevatia [22]. PBT trains a hierarchical detection tree using boosting. The tree is grown by binary splitting at nodes using a discriminative feature as the splitting criterion. Samples are moved either to the left or the right child, implicitly clustering the samples (both positive and negative samples). Each new node of the tree is then trained again into a strong classifier. CBT also generates a detection tree. The number of subclasses in CBT is fixed. CBT starts from a single node with all positive samples. Iteratively, nodes are split using the response of discriminative features selected for the samples in the current node. Unsupervised clustering using k-means is applied on the response of the samples on all gathered weak classifier until now.

Instead of manually specifying the number of output subclasses, our proposed algorithm generates the subclasses automatically. As in MCBoost and McBoost-WTA, we apply sample relabeling to redistribute samples to the most suitable subclass. MCBoost-WTA relabels samples after every *P* iterations of weak classifier learning (typically 32). In contrast, we apply sample relabeling after each iteration of our algorithm, because we learn a strong classifier in each learning iteration. This enables more rapid convergence to the final solution.

Instead of using a hierarchical binary tree (as in PBT), we define a single layer with multiple subclasses. In addition, we only consider subdivision of object/positive samples and use a fixed set of negative/background samples for each subclass. Whereas PBT splits at each level of the tree by considering a discriminative feature, we employ the classifier margin as a measure of visual variation in the current subclass.

In contrast to all the approaches that use boosting to select features, we do not apply the computationally expensive feature selection steps, but learn a linear classifier on all features, making our algorithm significantly faster. Furthermore, our system is memoryless, i.e. we do not keep the selected (weak) features from the previous iteration. This is advantageous, since features selected in the beginning of the division process may not be optimal for the final class division.

3. Our Approach

The algorithm aims at dividing object samples into subclasses in an unsupervised fashion. We start from a single class containing all object samples. In an iterative process, we divide the positive object samples into subclasses and merge visually similar classes. The optimization proce-

dure tries to obtain the optimal subclass division that results in the highest detection performance for the overall object class. The optimization is driven by a parameter that describes the amount of visual variation covered by a subclass. Each subclass is directly linked to a classifier trained on that subset of training samples. Although the positive samples are divided over the subclasses, each subclass uses the total set of negative samples for training the classifier.

To maintain a reasonable complexity, we use a linear classifier for each subclass and classify using all features. This makes the computational complexity of testing linear to the number of subclasses, which is significantly below the computational cost of using an SVM with many support vectors ($\gg 100$). After having generated the subclasses using our algorithm, time-complexity for actual detection can be decreased by training the subclasses using cascaded multi-class algorithms, but this is outside the scope of this paper and can be explored for future work.

To train the linear classifiers, we use Stochastic Gradient Descent (SGD). SGD has been widely used for the training of neural networks. Bottou and Bousquet [1] showed that the max-margin constraint as used in SVMs can be directly embedded into the gradient descent framework. Wijnhoven and De With [20] used this stochastic algorithm to train a classifier for object detection and have shown that SGD quickly converges to the optimal classifier. Because SGD considers each sample only once during optimization, the time complexity is linear to the number of training samples and is therefore significantly lower than other optimization algorithms and thus well suited for our problem.

Let us now first discuss a single iteration of the algorithm, which can be iteratively expanded later. For each subclass, we train a classifier using the object samples that belong to this subclass, together with all negative samples.

When a subclass is found to contain too much visual variation (larger than threshold T_V), we split the class into two subclasses, thereby simplifying the combined classification problem. Splitting of a subclass with high visual variation is done using *k*-means clustering ($k = 2$). Although *k*-means not necessarily results in the optimal division, it reduces the variation of the subclass. To measure the visual variation V_i that is covered by a classifier C_i, we use the $L2$-norm of the normal of the hyperplane of the linear classifier. This normal is represented by the weight vector ω. The vector length of ω is inversely proportional to the size of the margin [2]. Since we want to obtain subclasses that distinguish well their classification features, a large margin is preferred, coupled with a small vector length.

To measure the visual similarity between classifiers of the different subclasses, we compare the inter-class distance. This distance is calculated using the Euclidean distance between the ($L2$-normalized) classification weights of two classifiers. If the distance between two classifiers is smaller than a threshold T_M, we remove the least distinctive classifier. This is the classifier that has the highest similarity (smallest distance) to any other classifier. We allow the removal of multiple similar classifiers in a single iteration. Note that similar classifier weights does not mean that the respective subclasses are similar, it means that the information to distinguish each subclass from the background (object detection) is similar, making one of the two classes redundant.

After merging or splitting, all samples are relabeled and we continue with the next iteration. If samples are outliers in the current subclass (*e.g.* side-view sample in a front-view subclass), they can deteriorate the classification performance of the current class and can improve the performance of the other class. As discussed in WTA-Boost [24], relabeling of the samples to the most similar subclass increases overall performance. We apply this relabeling using the classification score. To avoid overtraining when a subclass contains only few training samples (*e.g.* a single sample), we impose a lower bound on the number of samples and remove subclasses with fewer samples. Note that we only allow either a merging or a splitting operation in each iteration. This allows the samples to move to the new subclass distributions.

This procedure is repeated until the algorithm has converged, which is based on three criteria. First, only a few samples are relabeled to a different subclass. We use a percentage of the positive samples as the threshold (T_C). Second, no subclasses are allowed to have a classifier with variance above the variation threshold (T_V). Third, there should be no subclasses that are visually similar, which means the similarity is below the threshold (T_M). The total algorithm is outlined in Algorithm 1.

The main computationally intensive component of the algorithm is the training of the classifiers. This is done at each iteration for each subclass and thus takes more time when growing into more subclasses. Splitting a subclass is performed using *k*-means ($k = 2$) with its own computational requirement. Per classifier, the computational complexity is linear to the number of training samples times the number of optimization epochs [20]. Since there are only a few positive samples compared to the number of negatives (hundreds vs. thousands, resp.), the computational complexity of a single training epoch is linear to the fixed number of negatives. Therefore, the time complexity C_t of a single iteration t is $N_i \times epochs \times |S^-|$.

4. Experiment: Multi-View Car Detection

For the experiment we use the following settings of the proposed algorithm. We converge when 1% of the positive samples switches label ($T_C = 0.01 \times |S_0^+|$). We remove subclasses using similarity threshold $T_M = 1$. In each iteration, we run 5 epochs of SGD optimization using $\lambda = 0.001$

Algorithm 1 Unsupervised sub-categorization algorithm.

Let S_0^+ be the set of all positive training samples
Let S^- be the set of negative training samples
Let N be the number of subclasses, set $N = 1$
converged = false
while not converged **do**
  for all subclasses i **do**
    Train classifier C_i with samples S_i^+ and S^-
    Calculate classifier variance V_i
  end for
  for classifiers C_i with $V_i > T_V$ **do**
    Split subclass S_i^+ into two sets, S_{N+1}^+ and S_{N+2}^+
    Remove subclass S_i^+
    $N++$
  end for
  Create classifier similarity matrix $M(i, j)$
  for classifiers C_i, C_j with $M(i, j) > T_M$ **do**
    Remove least distinctive classifier, C_i or C_j
  end for
  for all classes i **do**
    for all positive samples p of S_i^+ **do**
      Find classifier C_j with highest score for p
      Move sample p to S_j^+
    end for
  end for
  if number samples moved $> T_C$ AND
    All classifiers C_i have $V_i < T_V$ AND
    All classifiers C_i, C_j have $M(i, j) < T_M$ **then**
    converged = true
  end if
end while

using a subset of the negatives (30k samples). After convergence, we apply more accurate training by training with negative samples (100k samples). We investigate the effect of the classifier variation threshold T_V on the number of resulting subclasses and the related detection performance.

We evaluate the performance of our system on a car detection problem using the USC multi-view car dataset [1] and compare it with the work of Kuo and Nevatia [8]. The appearance variation between the cars is substantial. The main visual variations in appearance are caused by viewpoint (azimuth) and intra-class variation.

We use a sliding detection window of 128×64 pixels and extract HOG features [3] (8×8 cells, 18 orientations using sign, 1×1 blocks, $L2$ normalization). Our image data is exactly the same as in [8]. The 2,462 positive training samples are mirrored to obtain 4,924 samples. We apply bootstrapping using a single detector to obtain negatives from the PASCAL 2007 non-car samples (8,427 images). Detection is applied by downsampling in steps of 1.05 and starts at scale 0.7 to detect small cars. If multiple sub-classes result in a detection, we return the score of the best-matching subclass and the respective class label. Window-level detections are merged using a mean-shift procedure. For evaluation, we use the PASCAL VOC criteria [4] with 50% overlap. For the DET curves, we analyze 1,043 car-free images from the PASCAL 2006 set, as used in [8].

The objects are sometimes much smaller than the detection window (*e.g.* full-frontal). Therefore, we automatically adjust the bounding box for all subclass detectors by calculating the energy E of the positive classification weights ω over the full window. We grow a new bounding box until we reach 80% of the energy E, starting from the center of the window. This is done first horizontally and then vertically.

5. Results

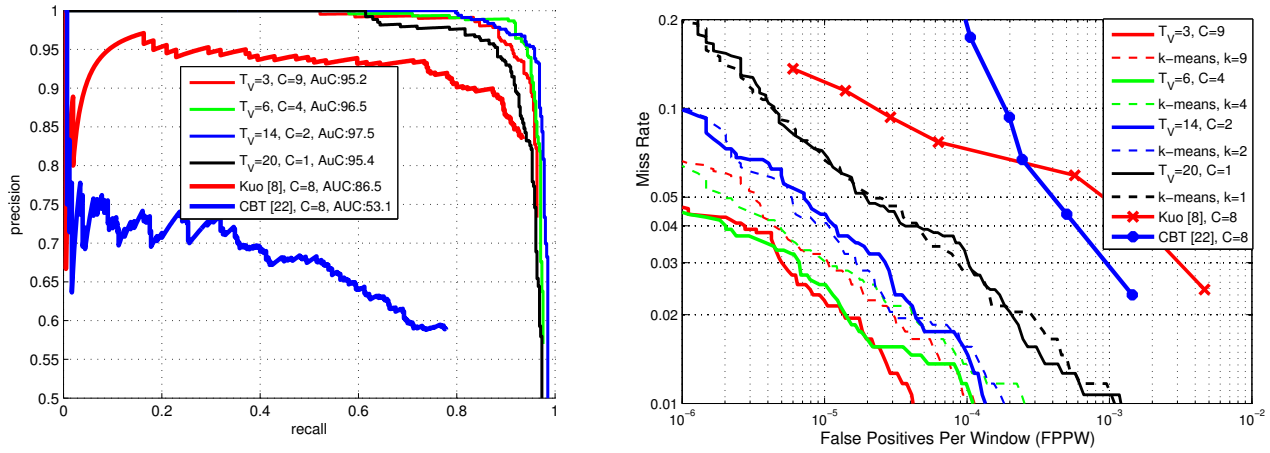
The detection performance is shown in Figure 1. The overall detection performance is shown in the recall-precision (RP) curves in Figure 1(a). It can be seen that a single linear classifier using HOG features already gives good detection performance. Evaluating the performance of our algorithm for more subclasses shows an increase in performance. Although the values of the Area-under-Curve (AuC) measure seem quite constant, evaluating the performance at typical working points in the graph shows an improvement (consider precision at *e.g.* 90% recall).

The authors expect that the large difference with the results reported in [8] could be partially explained by two different reasons. First, the image might not have been searched at small scales to find the small cars (the smallest car has size 61×41 pixels). Second, there might not have been any compensation for the fact that cars for some subclasses (e.g. frontal) are often much smaller than the size of the detection window. Note that the ground-truth contains tight bounding boxes around the cars and the evaluation criterion strongly penalizes incorrectly sized results. Even for our single class detector, the window defined by 80% positive classification energy was 104×56 pixels, which positively influences the evaluation measure.

Visual inspection of the detection results for the different number of subclasses shows that the drop in performance in RP is partially caused by misdetections, introduced by the incorrect merging of window-level detections. During merging, the subclass label is ignored and detections from all subclasses are merged, which can cause detections from one viewpoint to be merged with detections caused by a different viewpoint.

Since the algorithm aims at improving the classification performance, we now omit the merging step and evaluate the window-level classification performance using a DET Curve (see Figure 1(b)). The performance improves when creating more subclasses, at FPPW $1e-5$, the miss-rate

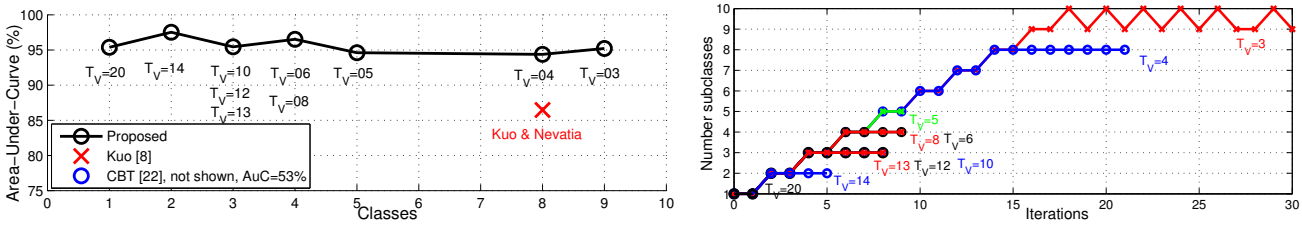
[1] http://iris.usc.edu/people/chenghak/download/README.pdf



(a) Recall-precision after merging window-level detections.

(b) DET curve, window-level performance.

Figure 1. Recall-precision and DET curves (USC mult-view car dataset). Curves for CBT and Kuo & Nevatia reproduced from [8].



(a) Number of subclasses vs. detection performance, measured as Area-Under-Curve of recall-precision curves (Figure 1(a)).

(b) Iterative convergence behaviour of subclasses over time. Different T_V values in different colors.

Figure 2. Convergence behaviour over time and recall-precision vs. number of subclasses.

decreases from 7.2% for a single subclass, to 4.4%, 2.5% and 2.1%, for 2, 4 and 9 classes, respectively. Note that the difference between 4 and 9 subclasses is marginal, indicating that the division in 4 subclasses is close to optimal and splitting in more subclasses will not significantly improve the detection performance.

Since the dataset contains accurately annotated object samples with a uniform distribution over the view-sphere, we also evaluate the performance of sub-categorization using standard *k*-means (Figure 1(b)). Although *k*-means gives decent subclass divisions, it is outperformed by our algorithm. At FPPW $1e-5$, the miss-rate goes from 3.0% to 2.5% and from 3.0% to 2.1% for 4 and 9 classes, respectively. This shows that the use of non-object (background) information improves the performance.

The relationship to the number of resulting classes is shown in Figure 2(a). As already discussed, although the AuC measure shows a relatively constant performance over the number of classes, the actual window-level performance significantly improves.

To see how fast the algorithm converges, we have plotted the number of subclasses over time (Figure 2(b)). Allowing a high visual variation T_V, results in a single subclass. Lowering T_V makes the algorithm more critical, resulting in a higher number of subclasses. We can see that the algorithm quickly splits into the final number of subclasses and only needs a few iterations to converge. When T_V becomes too strict, the algorithm never converges, because subclasses are always split and then removed again because of high visual similarity with another subclass.

In addition to an increase in classification performance, our algorithm generates semantically meaningful subclasses that are nicely distributed over the viewpoint circle. Examples of the generated subclasses for 8 classes are shown in Figure 3(a), together with the HOG features with positive classification weights. Some detection examples are shown in Figure 3(b).

We want to note that varying the number of positive training samples would not influence the working of the algorithm. As long as sufficient samples are available for each variation (*e.g.* viewpoint), similar subclasses will be found. Sub-categorization will only fail if too few samples are available to train decent classifiers (overtraining).

Annotation accuracy of the training samples is important. Because the used HOG features are explicitly spatially defined, variation in the object annotation box generates a

different description, resulting in a different subclass. The algorithm was evaluated on the PASCAL VOC datasets, but the annotations were found too noisy to generate proper subclasses. Typically, only one subclass was found, or very many subclasses because of overtraining.

Although the computational complexity scales linear to the number of of sub-classes, in practice the computational requirements are less. The detection stage at scale 1.0 takes 220 msec to calculate the HOG features and 110 msec to apply the detector for a single class. Linear extrapolation for 8 sub-classes would result in a time-requirement of 1100 msec, but in practice this takes only 730 msec because of data caching in the processor. Experiments were done on a i7-2600k processor at 3.4 GHz using a single-threaded non-optimized implementation.

6. Conclusions

We have presented a novel algorithm for the unsupervised sub-categorization of object classes for the purpose of object detection. Both object and non-object information (background) is considered in the optimization process. The number of desired subclasses does not need to be manually specified, and results explicitly from the output of the algorithm. An iterative split-and-merge procedure automatically divides a subclass into two new classes if the visual variation is too large, while it merges two classes if they are visually similar. To guide the optimization of the subclass division process, we employ a visual variation and similarity metric, based on the size of the classifier margin, i.e. the length of the normal of the hyperplane. After each iteration, samples are relabeled to the most preferred subclasses and because of the use of a fast stochastic learning algorithm, full retraining is applied. We evaluate the performance on a car detection problem and show that detection performance is significantly improved over a range of subclasses, using standard HOG features and linear classification. The algorithm converges in a few iterations, limiting total training time to a few minutes. In addition to the improvement in classification performance, the found subclasses are semantically meaningful.

References

[1] L. Bottou and O. Bousquet. Learning using large datasets. In *Mining Massive DataSets for Security*. 2008. 3

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. 3

[3] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In *CVPR*, 2005. 1, 4

[4] M. Everingham, A. Zisserman, C. K. I. Williams, and L. van Gool. PASCAL VOC2006 Results, http://pascallin.ecs.soton.ac.uk/challenges/voc/voc2006/results.pdf, 2006. 4

[5] C. Huang, H. Ai, Y. Li, and S. Lao. Vector Boosting for rotation invariant multi-view face detection. In *ICCV*, 2005. 1

[6] M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, July 2003. 1

[7] T.-K. Kim and R. Cipolla. Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *NIPS*, 2008. 2

[8] C.-H. Kuo and R. Nevatia. Robust multi-view car detection using unsupervised sub-categorization. In *Workshop on Applications of Computer Vision (WACV)*, 2009. 2, 4, 5

[9] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. In *CVPR*, 2007. 1

[10] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1):259–289, May 2008. 1

[11] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *Proc. CVPR*, 2010. 1

[12] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-Independent Object Class Detection using 3D Feature Maps. In *CVPR*, 2008. 1

[13] Y.-Y. Lin and T.-L. Liu. Robust Face Detection with Multi-Class Boosting. In *CVPR*, 2005. 1

[14] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009. 1

[15] Y. Quan, A. Thangali, V. Ablavsky, and S. Sclaroff. Multiplicative kernels: Object detection, segmentation and pose estimation. In *CVPR*, 2008. 2

[16] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *CVPR*, pages 746–751, 2000. 1

[17] Y. Shan, F. Han, H. S. Sawhney, and R. Kumar. Learning exemplar-based categorization for the detection of multi-view multi-pose objects. In *CVPR*, 2006. 2

[18] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*, 2009. 1

[19] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *ICCV*, 2005. 2

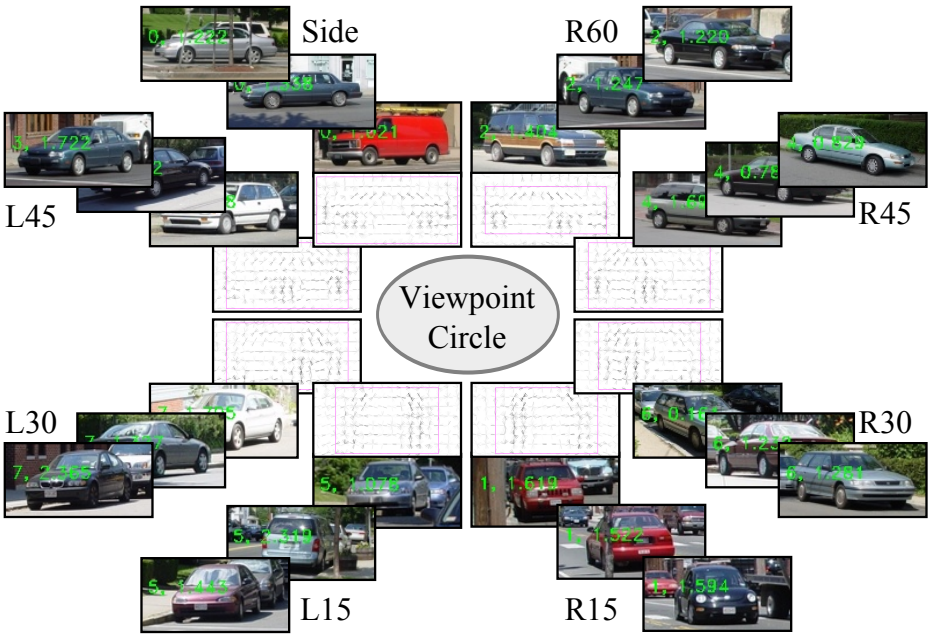
[20] R. G. J. Wijnhoven and P. H. N. de With. Fast training of object detection using stochastic gradient descent. In *ICPR*, 2010. 3

[21] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Proc. IEEE Intl. Conf. on Automatic Face and Gesture Recognition*, pages 79–84, May 2004. 1

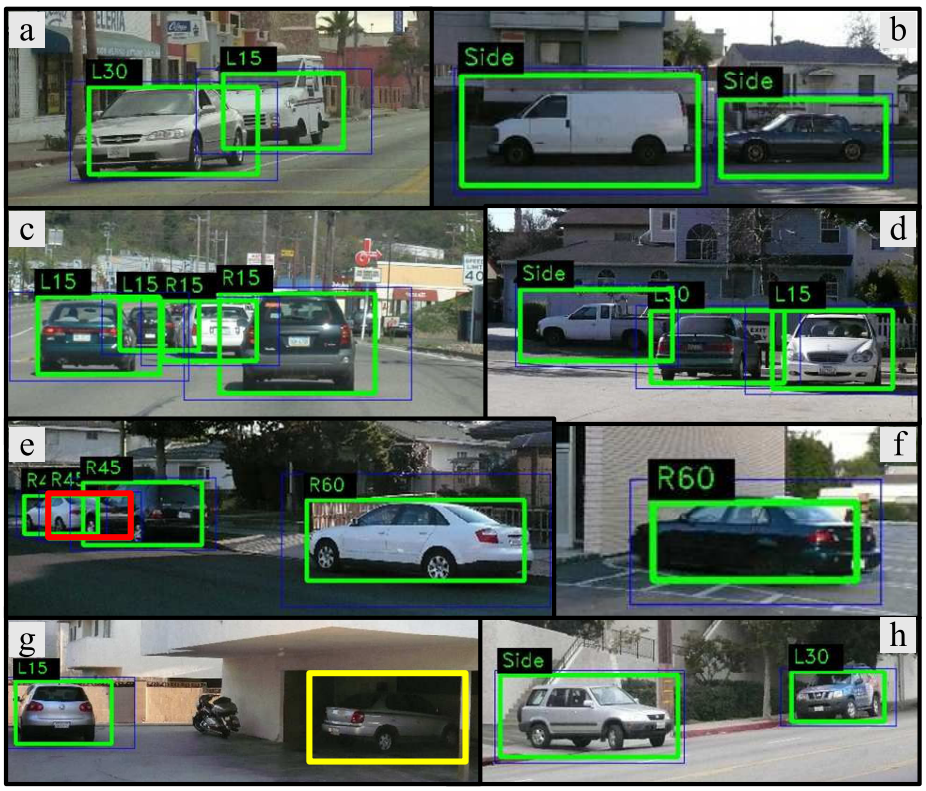
[22] B. Wu and N. Ram. Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In *ICCV*, pages 1303–1310, 2007. 2

[23] P. Yan, S. M. Khan, and M. Shah. 3D model based object class detection in an arbitrary view. In *ICCV*, 2007. 1

[24] C. Zhang and Z. Zhang. Winner-take-all multiple category boosting for multi-view face detection. Technical Report MSR-TR-2009-190, November 2009. 2, 3



(a) Subclasses and HOG features with positive classification weights for 8 subclasses. Corresponding labels are manually annotated per subclass.



(b) Example detections using 8 subclasses. Rectangles: blue: original classification window, green: 80% energy, red: false detections, yellow: missed detections.

Figure 3. Subclasses as generated by the algorithm and detection results (USC multi-view car dataset).