Robust Multi-View Car Detection using Unsupervised Sub-Categorization

Cheng-Hao Kuo and Ramakant Nevatia
University of Southern California, Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089, USA
{chenghak|nevatia}@usc.edu

Abstract

This paper presents a novel approach for multi-view car detection using unsupervised sub-categorization instead of manual labeling. Cars have large variability of models and the view-point makes the appearance change dramatically. For object classes with a large intra-class variation like cars, a divide-and-conquer strategy may be applied. Instead of using manually predefined intra-class sub-categorization, we examine several non-linear dimension reduction methods and group samples in the low-dimension embedding in an unsupervised way. The clustered samples have strong view-point similarities internally. A boosting-based cascade tree classifier is trained based on these sub-categorizations. To demonstrate the capability of our multi-view car detector, we create a more challenging test set with annotations. Compared to the UIUC side-view car data set, our test set contains a large range of car models, view points, and complex backgrounds. We compare our approach with previous methods and the result shows that ours outperforms the state-of-the-art methods.

1. Introduction

Object detection and localization is one of the important problems in computer vision research. Here we focus on a specific case: cars. Cars have a large intra-class variation and the appearance changes dramatically as the view-point changes. Occlusion, varying illumination, and a complex background make the problem difficult as usual. For an object class with small intra-class variation, e.g. frontal faces, a single cascade classifier proposed by Viola and Jones [20] has been viewed to be a successful and efficient method. However, for an object class with large intra-class variation, e.g. cars with different viewpoints, as in figure 1, we need a more complex method to solve the problem. One intuitive solution is to adopt a divide-and-conquer strategy, where the training samples are partitioned into several sub-categorizations via a high-level knowledge. In other words, when the original problem is too challenging to solve, divide them and solve the easier tasks individually.

Mostly the manual categorization is ambiguous to humans, prone to error, and time-consuming. Instead of using a manual intra-class sub-categorization, we propose a novel combination of image descriptors, non-linear dimension reduction, and clustering to group car samples in an unsupervised way. We also build a robust tree-structured detector based on the sub-categories. Besides, a new car test set is created to demonstrate the capability of our multi-view car detector. This new test set is more challenging than previous ones, e.g. the UIUC car data set, in many aspects. Our system is evaluated on several car data sets and experimental results show that our detector outperforms state-of-art methods.

1.1. Related Work

Multi-view object detection and localization have been an active research area in computer vision. In [15], Schneiderman and Kanade adopted view-based detectors in parallel using Bayesian rule classifiers with application to faces and cars. Viola and Jones [8] extended their famous work in [20] to handle multi-view faces by a pose estimator and decision tree. In [11], Li et al. described a pyramid-structured multi-view face detector. Huang et al. [7] de-
veloped a rotation invariant multi-view face detector using a Vector Boosting Tree (VBT). However, these methods require manual labeling to divide the object samples into several categories. Since the number of training examples is often large, manual categorization is expensive and it is easy to make mistakes which make learning difficult.

Some strategies have been developed to achieve automatic intra-class sub-categorization, or to learn a detector with implied unsupervised clustering. In [16], Seemann et al. extract object silhouettes or shapes from the training images and cluster them with an agglomerative clustering scheme. However, the features used for clustering are different from the features used for detection. In [17], Shan et al. use two nested AdaBoost loops; the outer loop is used to pick representative exemplars and the inner AdaBoost is used to select discriminative features on the selected exemplars; Their performance is highly affected by choosing the initial candidate exemplar set. Wu et al. [21] presented a Cluster Boosted Tree (CBT) to automatically construct tree structured object detectors without using predefined intra-class sub-categorization. Although CBT shows good results on several data sets, there is no semantic meaning to the learned sub-categories. The features used to split training samples are learned between objects and non-objects, not between intra-class objects. In [23], Yuan et al. proposed a multiplicative form of two kernel functions to learn the similarity for foreground-background and within-class variation jointly. The detectors associated with each foreground training sample are clustered to form the final set of detectors. However, it requires a large number of detectors and the experiment is conducted only on some positive and negative samples, not on the test images as the real localization task. Zhang et al. [24] clustered extensive human poses by embedding the human silhouettes in a low-dimensional manifold and built a detector for each cluster. Their method still requires manual labeling of 13 points for one person.

1.2. Outline of our approach

We focus on the problem of multi-view car detection. Our approach contains two main elements: Unsupervised hierarchical sub-categorization and a boosting-based tree-structured detector. For the former part, we first adopt Locally Linear Embedding (LLE) [14] to represent high dimensional car samples in a 2D space and further normalize these 2D points onto a circle. Within this compact space, car samples are grouped into several sub-categories by the k-means clustering method. These sub-categories, as leaf nodes, are progressively merged together by the agglomerative clustering method to construct a tree from bottom up top.

For the second part, we construct a tree-structured classifier based on the sub-categorization obtained from the previous part. At each node a rejection cascade is learned. In each level of cascade of a certain node, we use the Gentle AdaBoost algorithm [5], a variant of the AdaBoost method, to select discriminative features and form a strong classifier. Whenever the classifier of the node is not strong enough to distinguish between the object/non-object samples, we divide the object samples into two groups according to the hierarchical set of sub-categories and form two nodes. The number of leaf nodes is determined by the number of sub-categories we set in the first part. The training process stops when the desired target training accuracy is reached.

The paper is organized as follows: In section 2, we present our method for unsupervised sub-categorization. In section 3, we describe our framework including the search tree structure, the features, and learning algorithm. Section 4 shows experimental results on multi-view car detection. Some conclusions and discussions are in section 5.

2. Unsupervised Sub-categorization

Suppose we have car samples with different models and different orientations. Instead of manually labeling the view points, we present a novel combination to accomplish unsupervised sub-categorization such that each category has a strong view-point similarity visually. Due to the 360° rotation of different cars, we construct the lower dimensional embedding of car samples to represent the variation on the view-points. We examine two classical dimension reduction methods, Isometric Mapping (ISOMAP) [18] and Locally Linear Embedding (LLE) [14], which discover low-dimensional representations of high-dimensional samples. ISOMAP uses geodesic distances to reflect the true low-dimensional geometry of the manifold. LLE recovers a global nonlinear structure from locally linear fits. For the features, instead of using the intensity of image pixels, we

![Two dimensional embedding of 2,462 cars using LLE. A set of car images is superimposed on all the data points. Note that the cars change view-points smoothly around the ring.](image-url)
adopt the Histogram of Oriented Gradients (HOG) [2] descriptor to form a feature vector for each car sample. The HOG features are selected since cars have strong edges in their appearances. Compared to ISOMAP, the computation of LLE is stable and cheap since it only requires sparse matrix diagonalization; the visual result in 2D scatter is also better. Hence we only present our unsupervised sub-categorization result via LLE. Figure 2 shows the two dimensional embedding of car training samples via LLE. The two dimensional embedding discovered by LLE represents the cars with different view-points as a ring. Traveling along the trajectory of this ring, the view-points change smoothly.

By selecting a center point of all sample points in the 2D embedding, we are able to normalize every point onto a full circle based on the angle to the center point. Then an unsupervised clustering algorithm, e.g. k-means, can be applied on this circle. As in Figure 3, the clustered samples which form an arc have strong similarities in orientation.

One intuitive method for unsupervised sub-categorization is simply using k-means on the space of the high dimensional features, e.g. the HOG features. However, the clustered samples contain a large variation of view-points and the result is sensitive to the number of clusters. Our explanation is that the HOG feature space mainly describes shapes of different cars and somewhat ignores their textures and colors. Imposing a 2D embedding on such a feature space makes the LLE method annihilates the variance from different car models and simply preserves the variance due to view point changes in the low dimensional space since the latter variance is more significant than the former one. Our method achieves very good clustering of view-points of cars and the result is robust to the number of clusters.

Once k clusters are constructed, we are able to build the classifier based on the learned sub-category, e.g. view-point classifiers in parallel. In order to share features in a more efficient way, we propose a hierarchical set of sub-category which can be established by agglomerative clustering. We choose the distance between any two clusters to be the mean distance between elements of each cluster, as in (1).

$$d(A, B) = \frac{1}{|A||B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$$  (1)

The result of agglomerative hierarchical clustering is shown in Figure 4. By visual observation, the algorithm merges two clusters with similar view-points gradually. It provides an excellent setup for the multi-view detection problem using the divide-and-conquer strategy. Being different from the top-down splitting of the training samples without semantic meanings in CBT [21], we use the bottom-up merging of training samples with close view points to build the tree-structured classifier.

3. Framework of Tree-Structured Classifier

Next, we describe the framework of our tree-structured classifier. Each node contains a cascade of rejecters. We apply varied-block-sized HOG features and the Gentle AdaBoost algorithm to form a strong classifier in each level of cascade. The learning algorithm of the tree and the nodes are presented.
3.1. The structure of Cascade Tree Classifier

For the implementation of our classifier, we present a method called cascade tree classifier. It is suitable for detection task since it can make a rejection decision at both leaf and non-leaf nodes. When a query comes, it is tested by the root node detector first. If the query passes the root node detector, it is sent to the child nodes and tested by the detectors of the child nodes. The possible path of each node to its child nodes is non-exclusive, unlike in Jones and Viola’s decision tree in [8]. In our cascade search tree, an input is classified as an object if it passes at least one route from the root node to any leaf node. It is similar in structure to the Vector Boosting Tree (VBT) detector proposed by Huang et al. [7]; however, we do not use vector boosting to learn the output vectors jointly. We also keep the flexibility of using both width-first-search and depth-first-search. Since each category has strong similarity internally, we are able to make view-point estimation according to the output leaf node.

3.2. Learning the Cascade Tree

After unsupervised sub-categorization and hierarchical clustering are established, we are in a position to learn our cascade search tree classifier. Suppose we have tree nodes \(d_i, i = 1...N_d\), and car samples \(S^+_i\) associated with \(d_i\), where \(N_d\) is the number of sub-categories. Node \(d_1\) is the root node and \(S^+_1\) consists of all the car samples. The leaf nodes contain the car samples in each sub-category.

Our goal is to train a classifier \(T_i\) for every node and then combine them to form a tree-structured detector. In the beginning of the learning process, we train the classifier of the root node, treating all the car samples as the same category. The algorithm for training a classifier for a certain node is described in Section 3.4. Once the classifier for the root node \(T_1\) is learned, we visit the child nodes of \(d_1\), e.g. \(d_2\) and \(d_3\). At this time, \(S^+_1\) is split into two categories, e.g. \(S^+_2\) and \(S^+_3\). The child classifiers \(T_2\) and \(T_3\) are learned individually using their associated car samples with their ancestor classifier \(T_1\). The learning process stops when all the leaf nodes have been visited and all classifiers \(T_i\) have been learned. The pseudo code of learning a cascade search tree classifier is presented in Algorithm 1.

3.3. Features

There are a large number of image features developed in the literature, such as the wavelet coefficients [15], Haar-like features [20], SIFT-like features [12], grids of HOG descriptor [2], Edgelet features [22], covariance descriptor [19], and biologically-inspired sparse, localized feature [13]. In this paper, we adopt the varied-block-sized HOG feature [25] for our implementation since it encodes more information than the fixed-block-sized HOG and it is suitable for integration into the cascade framework.

Algorithm 1 Learning cascade search tree

<table>
<thead>
<tr>
<th>Input:</th>
<th>({d_i}): Tree nodes ({S^+_i}): Positive samples in node (d_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>initialize (L = d_1), where (d_1) is the root node</td>
</tr>
<tr>
<td>2:</td>
<td>while (L) is not empty do</td>
</tr>
<tr>
<td>3:</td>
<td>Pop the first node (d) from (L)</td>
</tr>
<tr>
<td>4:</td>
<td>Collect negative sample (S^-<em>d) using ({T</em>{a(d)}}), where</td>
</tr>
<tr>
<td></td>
<td>(a(d)) are all the ancestor nodes of (d)</td>
</tr>
<tr>
<td>5:</td>
<td>Train a cascade classifier (T_d = F(S^+_d, S^-<em>d, {T</em>{a(d)}}))</td>
</tr>
<tr>
<td>6:</td>
<td>if (d) is not leaf node then</td>
</tr>
<tr>
<td>7:</td>
<td>for (j = 1) to (n) do</td>
</tr>
<tr>
<td>8:</td>
<td>Find the (j)-th child node (c_j) of node (d)</td>
</tr>
<tr>
<td>9:</td>
<td>Push (c_j) into (L)</td>
</tr>
<tr>
<td>10:</td>
<td>end for</td>
</tr>
<tr>
<td>11:</td>
<td>end if</td>
</tr>
<tr>
<td>12:</td>
<td>end while</td>
</tr>
</tbody>
</table>

Output: \(\{T_i\}\): Detectors for nodes \(d_i\)

3.4. Learning a node classifier

Here we describe a method to learn a node classifier in the cascade tree. It takes the form of a cascade rejecter. Each level in the cascade is learned using a 36D HOG feature with varied block sizes and Gentle AdaBoost [5]. A 36D HOG feature is formed by concatenating the 9 orientations bins in \(2 \times 2\) cells.

Gentle AdaBoost, a gentler version of Real AdaBoost, uses Newton stepping rather than exact optimization at each step. Consider a binary classification problem: let \(\{x_i, y_i\}\) be training samples where \(x_i\) is the feature vector and \(y_i \in \{1, -1\}\) is the label. In each boosting round, Gentle AdaBoost fits a regression function, \(f(x)\), by weighted least-squares of \(y_i\) to \(x_i\) with weights \(w_i\) for minimizing the object function:

\[
E[e^{-yF(x)}] = \sum w_i e^{y_i f(x_i)}
\]

The Gentle AdaBoost is described in Algorithm 2.

In our framework, Gentle AdaBoost selects HOG features and combines them to form strong classifiers in each level of cascade. The 36D HOG feature of a sample \(x_i\) can be represented as \(h_i \in \mathbb{R}^{36}\). If we adopt a linear regression function, then \(f(h_i) = \langle v, h_i \rangle \in \mathbb{R}^1\) is the inner product of \(v\) and \(h_i\), where \(v\) contains the regression coefficients. Assume we have \(N\) training samples with 36D HOG features, the weighted least squares problem can be represented by

\[
Av = y
\]
The vector \( v \) which minimized \( \varepsilon \) can be computed by linear algebra technique:

\[
v = (A^TWA)^{-1}A^TWy
\]  

where \( A \in \mathbb{R}^{N \times 36} \) is the data matrix and \( y \) is the vector of labels. The weighted least square error \( \varepsilon \) can be written as

\[
\varepsilon = \sum_{i=1}^{N} w_i[y_i - f(h_i)]^2 = \sum_{i=1}^{N} w_i[y_i - \langle v, h_i \rangle]^2  
\]  

(4)

The performance and speed of our method are compared with the state-of-art methods.

4.1. UIUC car data set

Since the UIUC car data set has been used in many previous reports, we evaluate our detector on this set. The training samples contain 550 cars of side-view with the same size 100 \times 40 pixels. There are two test sets: the first for the single-scale case and the second is for the multi-scale case. The single-scale test set contains 200 cars of fixed size which is close to training samples in 170 images. The multi-scale test set contains 139 cars with varied sizes between 89 \times 36 and 212 \times 85 pixels in 107 images. The performance of the detection task is usually evaluated by precision and recall curves.

We compare our methods with previous ones in Table 1. It can be shown that our detector has competitive performance to the state-of-the-art methods.
Figure 5. Our novel multi-view car test images. Compared to UIUC side-view car test set, our new test set includes extensive view-point change and large intra-class variation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Single-scale</th>
<th>Multi-scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agarwal et al. [1]</td>
<td>76.5%</td>
<td>39.6%</td>
</tr>
<tr>
<td>Fergus et al. [4]</td>
<td>88.5%</td>
<td>-</td>
</tr>
<tr>
<td>Fritz et al. [6]</td>
<td>88.6%</td>
<td>87.8%</td>
</tr>
<tr>
<td>Kapoor/Winn [9]</td>
<td>94.0%</td>
<td>-</td>
</tr>
<tr>
<td>Mutch/Lowe [13]</td>
<td>99.94%</td>
<td>90.6%</td>
</tr>
<tr>
<td>Wu/Nevatia [21]</td>
<td>97.5%</td>
<td>93.5%</td>
</tr>
<tr>
<td>Leibe et al. [10]</td>
<td>97.5%</td>
<td>95.0%</td>
</tr>
<tr>
<td>Ours</td>
<td>98.5%</td>
<td>95.0%</td>
</tr>
</tbody>
</table>

Table 1. Detection rates at the point of equal precision and recall in UIUC car test set.

4.2. New data set

Although the UIUC car test set is well-established, it contains only side-view cars and most researchers obtain nearly perfect results. In order to demonstrate the capability of our multi-view car detector, we created a new and more challenging test set. This test set contains 196 images with 410 cars in sizes varying between 61 × 41 and 304 × 165 pixels. The images are taken mainly in streets and highways without heavy traffic. The cars in these images include various models such as sedans, pickups, vans, and SUVs. Compared to the UIUC car test set, ours is more difficult in the following aspects: 1) cars with a larger range of view-points; 2) more occlusions on cars; 3) more complex backgrounds; 4) larger intra-class variations; 5) larger image size; 6) more cars in one image. In our test images, we manually annotated each car sample using a bounding box. Cars with a height smaller than 40 pixels or percentage of occlusion greater than 50% are not included in our annotation. For the training samples, we collected 2,462 multi-view car images from the MIT street scene image set and normalized the sample size to 128 × 64 pixels. For negative training images, we extracted 8,427 images not containing any cars from the PASCAL VOC2007 Challenge data.

There are several metrics used for the evaluation of classification/detection task, e.g. the Precision/Recall (PR) curve, the Detection Error Trade-off (DET) curve, and the Receiver Operator Characteristic (ROC) curve. Summary of the measures includes Average Precision (AP), Equal Error Rate (ERR), and Area Under Curve (AUC).

We follow the definition of true detection from PASCAL VOC Challenge. A detection is considered true or false positive according to the overlap ratio between the predicted bounding box $B_p$ and ground truth bounding box $B_{gt}$. We follow the formula:

$$\frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} > 50\%$$

Multiple detections of the same object in an image are considered false detections. A post-processing, e.g. merging multiple detections for the same object, is required to

---

1 http://iris.usc.edu/

2 http://cbcl.mit.edu/software-datasets/streetscenes/
achieve a higher precision rate. We prefer the PR curve over the DET and ROC curves since our problem is a detection/localization task. However, we also provide the DET curve on a log-log scale, \( \text{i.e.} \) miss rate versus False Positives Per Window tested (FPPW) as in [2], without considering different post-processing techniques. To generate the DET curve, another 514 car samples (1,028 with reflections) are collected from the MIT street scene image set and 1,043 car free images are selected from PASCAL VOC2006 train/val data set.

We apply our multi-view detector and compare with Wu et al.'s Clustering Boosted Tree (CBT) [21]. We selected CBT to compare, as it is designed for multi-view object detection and the result applied to several data sets are competitive with state-of-art. Like our detector, CBT does not require manual sub-categorization of objects. The major difference is that CBT divides the sample space by unsupervised clustering based on discriminative images features. We used the author provided CBT code to train the detector on the same training samples and test on our own multi-view car test set. Our method dominates CBT in the PR curve in Figure 6. The DET curve is also shown in Figure 7. Our method still has better performance when FPPW is less than \( 10^{-4} \) which we consider is the minimum required false positive rate to build a useful detector. These two curves show that our method outperforms CBT. Some detection results are shown in Figure 8. In terms of training time, it takes about two days for training our detector, as opposed to more than one week for training CBT.

4.3. Computational Time

Given a test image, our method can search around 40,000 detection windows per second, which corresponds to 1.5 second for a dense scan of a \( 640 \times 480 \) image. The training time is about two days, which is efficient for training a cascade tree-structured classifier.

5. Conclusions and Discussions

We have described an approach for multi-view object detection using unsupervised sub-categorization and cascade search tree classifier. Instead of using manually predefined intra-class sub-categorization, we propose a novel combination of image descriptors, non-linear dimension reduction, and clustering to group car samples in an unsupervised way. The clustered samples have significant semantic meanings. A cascade search tree classifier is trained based on these sub-categorizations. To demonstrate our capability of multi-view car detector, we create a more challenging test set with a large range of car models, view points, and complex background. The experimental results show that our method outperforms the state-of-the-art.

6. Acknowledgements

This research was funded, in part, by the U.S. government VACE program.
Figure 8. Some detection results on our new car test set. The true detections are represented by green rectangles. The false detection is shown in red rectangle. The missed car is shown in yellow rectangle from the annotations. This figure is best viewed in color.

References