Presented in ECCV'02

Learning a Sparse Representation for Object Detection

Shivani Agarwal and Dan Roth

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{sagarwal,danr}@cs.uiuc.edu

Abstract. We present an approach for learning to detect objects in still gray images, that is based on a sparse, part-based representation of objects. A vocabulary of information-rich object parts is automatically constructed from a set of sample images of the object class of interest. Images are then represented using parts from this vocabulary, along with spatial relations observed among them. Based on this representation, a feature-efficient learning algorithm is used to learn to detect instances of the object class. The framework developed can be applied to any object with distinguishable parts in a relatively fixed spatial configuration. We report experiments on images of side views of cars. Our experiments show that the method achieves high detection accuracy on a difficult test set of real-world images, and is highly robust to partial occlusion and background variation.
In addition, we discuss and offer solutions to several methodological issues that are significant for the research community to be able to evaluate object detection approaches.

1 Introduction

This paper describes an approach for learning to detect instances of object classes in images. The development of reliable object detection systems is important for a wide variety of problems, such as image classification, content-based image retrieval, tracking and surveillance. Much research has been done in this direction. However, the problem remains a largely unsolved one.

The approach presented is based on the belief that the key to finding a solution to this problem lies in finding the right representation. Specifically, we suggest that in order to extract high-level, conceptual information such as the presence of an object in an image, it is essential to transform the raw, low-level input (in this case, the pixel grayscale values) to a higher-level, more "meaningful" representation that can support the detection process.

One theory of biological vision explains object detection on the basis of decomposition of objects into constituent parts [1–3]. According to this theory, the representation used by humans for identifying an object consists of the parts that constitute the object, together with structural relations over these parts that define the global geometry of the object. Such a representation also forms the basis of some computational theories of object detection [4].

In this paper, we describe an approach that uses an automatically acquired, sparse, part-based representation of objects to learn a classifier that can be used to accurately detect occurrences of a category of objects in natural scenes. As shown in our experiments, the method is highly robust to cluttered backgrounds and partial occlusion.

1.1 Related Work

Most work that has used learning in object detection has been done at the pixel level (e.g., [5–7]). Here we mention a small number of recent studies that have deviated from this, and relate them to our work.

Several approaches represent objects using low-level features. For example, in [8, 9], the object parts are local groupings of common primitive features such as edge fragments, while in [10], rectangle features are used to capture the presence of edges, bars and other simple structures. We believe that the expressivity of a part-based representation can be enhanced by considering distinctive, higher-level parts that are rich in information content and more specific to the object class of interest. Such an approach has been followed in [11], in which separate classifiers are used to detect heads, arms and legs of people in an image, and a final classifier is then used to decide whether or not a person is present. However, the work in [11] requires the object parts to be manually defined and separated for training the individual part classifiers. In order to build a system that is easily extensible to deal with different objects, it is important that the part selection procedure be automated. Our method for automatically selecting information-rich parts builds on a technique described in [12], in which interest points are used to collect distinctive parts. In [12], a generative probabilistic model is learned over these parts. Our method, on the other hand, does not need to assume any probabilistic model; we simply collect parts that could be of interest and then directly learn a classifier over them. In addition, the model learned in [12] relies on a very small number of fixed parts, making it potentially sensitive to large variations across images. By learning over a large feature space, we are able to learn a more expressive model that is robust to such variations.

We use a feature-efficient learning algorithm that has been used in a similar task in [13]. However, [13] uses a pixel-based representation, whereas in our approach, images are first transformed (automatically) and represented using a higher-level and more sparse representation. This has implications both in terms of detection accuracy and robustness, and in terms of computational efficiency: the sparse representation of the image allows us to perform operations (e.g., computing relations) that would be prohibitive in a pixel-based representation.

1.2 Problem Specification

We assume some object class of interest. Our goal is to develop a system which, given an image, can detect instances of this object class in the image, and return their locations. In the process of finding the locations of these object instances, we also require the system to be able to output the *number* of object instances it finds in the image. This may appear to be a trivial requirement for an object detection system, but as we discuss later in Section 2.4, it imposes a non-trivial condition that the system must satisfy.

1.3 Overview of the Approach

This section outlines our conceptual approach, which consists broadly of four stages.

1. *Vocabulary Construction*
   The first stage consists of building a "vocabulary" of parts that can be used to represent objects in the target class. This is done automatically by using an interest operator to extract information-rich parts from sample images of the object of interest. Similar parts thus obtained are grouped together and treated as a single part.
2. *Image Representation*
   Each input image is transformed and represented in terms of parts from the vocabulary obtained in the first stage. This requires determining which parts from the vocabulary are present in the image; a similarity-based measure is used for this purpose. Each image is then represented as a binary feature vector based on the vocabulary parts present in it and the spatial relations among them.
3. *Learning a Classifier*
   Given a set of training images labeled as positive (object) or negative (non-object), each image is re-represented as a binary feature vector as described above. These feature vectors are then fed as input to a supervised learning algorithm that learns to classify an image as a member or non-member of the object class, with some associated confidence. As shown in our experiments, the part-based representation captured by the feature vectors enables a relatively simple learning algorithm to learn a good classifier.
4. *Detection Hypothesis using the Learned Classifier*
   The final stage consists of using the learned classifier to form a reliable detector. We introduce the notion of a *classifier activation map*, which is generated by applying the classifier to various windows in a test image. We then present a method for producing a good detection hypothesis using this activation map.

This framework can be applied to any object that consists of distinguishable parts arranged in a relatively fixed spatial configuration. Our experiments are performed on images of side views of cars; therefore, we will use this object class as a running example throughout the paper to illustrate the ideas and techniques we introduce.

We describe each stage of our approach in detail in Section 2. Section 3 presents an evaluation of our method. In this section, we first discuss several methodological issues, including evaluation and performance measurement techniques, that are important for the research community to be able to evaluate object detection approaches. We then present our experimental results. Section 4 concludes with a summary and possible future directions.

2 Approach

This section describes each stage of our approach in detail.

Fig. 1. Left: A sample object image used in the vocabulary construction stage. **Center**: Interest points detected by the Förstner operator. Crosses denote intersection points; circles denote centers of circular patterns. **Right**: Patches extracted around the interest points.

2.1 Vocabulary Construction

To obtain an expressive representation for the object class of interest, we require distinctive parts that are specifi c to the object class but can also capture the variation across different instances of the object class. Our method for automatically selecting such parts is based on the extraction of interest points from a set of representative images of the target object. A similar method has been used in [12].

Interest points in an image are points that have high information content in terms of the local change in signal. They have been used in a variety of problems in computer vision, including stereo matching [14], object recognition and image retrieval [15]. Interest points have typically been designed and used for properties such as rotation and viewpoint invariance, which are useful in recognizing different views of the same object, and not for the "perceptual" or "conceptual" quality that is required for reliably detecting different instances of an object class. However, by using interest points in conjunction with a redundant representation that is described below, we are able to capture a degree of conceptual invariance that turns out to be suffi cient for this task.

We apply the Förstner interest operator [16, 12] to a set of representative images of the object class. This detects intersection points of lines and centers of circular patterns. A vocabulary of representative parts is then constructed by extracting small image patches around the interest points obtained. The goal of extracting a large vocabulary from different instances of the object class is to be able to "cover" new object instances, i.e. to be able to represent new instances using a subset of this vocabulary.

In our experiments, the Förstner operator was applied to a set of 50 representative images of cars, each 100×40 pixels in size. Figure 1 shows an example of this process. Patches of size 13×13 pixels were extracted around each such interest point, producing a vocabulary of 400 parts from the 50 images. This vocabulary is shown in Figure 2.

As seen in Figure 2, several of the parts extracted by this procedure are visually very similar to each other. To facilitate learning, it is important to abstract over these parts by mapping similar parts to the same feature id (and distinct parts to different feature ids). This is achieved via a bottom-up clustering procedure. Initially, each part is assigned to a separate cluster. Similar clusters are then successively merged together until no similar clusters remain. In merging clusters, the similarity between two clusters C_1 and C_2 is measured by the average similarity between their respective parts:

$$\text{similarity}(C_1, C_2) = \frac{\sum_{p_1 \in C_1} \sum_{p_2 \in C_2} \text{similarity}(p_1, p_2)}{|C_1| \times |C_2|}$$

where the similarity between two parts is measured by normalized correlation, allowing for small shifts of upto 2 pixels. Using this technique, the 400 parts were grouped into

Fig. 2. The vocabulary of 400 parts extracted by the Förstner interest operator.

Fig. 3. Examples of the clusters formed after grouping similar parts together.

270 clusters. While several clusters contained just one element, parts with high similarity were grouped together. Figure 3 shows some of the larger clusters that were formed. Parts belonging to the same cluster are treated as a single "conceptual" part by giving them the same feature id; in this way, by using a deliberately redundant representation that uses several similar parts to represent a single conceptual part, we are able to extract a higher-level, conceptual representation from the interest points. Experiments described in Section 3.3 show the role of the clustering procedure.

2.2 Image Representation

Having constructed the part vocabulary above, images are now represented using this vocabulary. This is done by determining which of the vocabulary parts are present in an image, and representing the image as a binary feature vector based on these detected parts and the spatial relations that are observed among them.

Part Detection Searching a whole image for the presence of vocabulary parts is computationally expensive. To focus our attention on the interesting regions in the image, the Förstner operator is first applied to the image, and patches around the interest points found in the image are highlighted. For each highlighted patch, we perform a similarity-based indexing into the part vocabulary. Each patch is compared to the vocabulary parts using the same similarity measure used earlier for clustering similar parts when constructing the vocabulary. If a sufficiently similar vocabulary part is found, the patch in the image is represented by the feature id corresponding to that vocabulary part. Figure 4 shows examples of the part detection process.

Relations over Detected Parts Spatial relations among the parts detected in an image are defined in terms of the distance and direction between each pair of parts. Several earlier approaches to recognition have relied on geometric constraints based on the distances and angles between object elements. For example, [17] models objects as polyhedra and, given input data from wich object patches can be inferred, performs

Fig. 4. Examples of the part detection process applied to a positive and a negative image during training. Center images show the patches highlighted by the interest operator; notice how this successfully captures the interesting regions in the image. These highlighted interest patches are then matched with vocabulary parts. In the right images, the highlighted patches are replaced by an arbitrary member of the part cluster (if any) matched by this detection process. These parts, together with the spatial relations among them, form our representation of the image.

recognition by searching for interpretations of the surface patches whose inter-patch distances and angles are consistent with those between corresponding model faces. In our approach, we do not assume a known model for the target object class, but attempt to use the object parts observed in object examples, together with the relations observed among them, to *learn* a model or representation for the object class. The distances and directions between parts in our approach are discretized into bins: in our implementation, the distances are defined relative to the part size and are discretized into 5 bins, while the directionss are discretized into 8 different ranges, each covering an angle of 45°. However, by considering the parts in a fixed order across the image, the number of direction bins that need to be represented is reduced to 4. This gives 20 possible relations (i.e. distance-direction combinations) between any two parts.

The 100×40 training images (and later, 100×40 windows in test images) that are converted to feature vectors have a very small number of parts actually present in them: on average, a positive window contains 6-8 parts, while a negative one contains around 2-4. Therefore, the cost of computing relations between all pairs of detected parts is negligible once the parts have been detected.

Feature Vector Each 100×40 training image (and later, each 100×40 window in the test images) is represented as a feature vector containing feature elements of two types:

(i) $P_n^{(i)}$, denoting the ith occurrence of a part of type n in the image ($1 \leq n \leq 270$ in our experiments; each n corresponds to a particular part cluster)
(ii) $R_m^{(j)}(P_{n_1}, P_{n_2})$[1], denoting the jth occurrence of relation R_m between a part of type n_1 and a part of type n_2 in the image ($1 \leq m \leq 20$ in our implementation; each R_m corresponds to a particular distance-direction combination)

These are binary features (each indicating whether or not a part or relation occurs in the image), each represented by a unique identifier. The re-representation of the image is a list of the identifiers corresponding to the features that are *active* (present) in the image.

[1] In the implementation, a part feature of the form $P_n^{(i)}$ is represented by a unique feature id, which is an integer determined as a function of n and i. Similarly, a relation feature of the form $R_m^{(j)}(P_{n_1}, P_{n_2})$ is assigned a unique feature id that is a function of m, n_1, n_2 and j.

2.3 Learning a Classifier

Using the above feature vector representation, a classifier is trained to classify a 100×40 image as car or non-car. We used a training set of 1000 labeled images (500 positive and 500 negative), each 100×40 pixels in size. The images were acquired from different sources: partly from the World Wide Web, partly with a camera, and partly by taking video sequences of cars in motion and processing the digitized video with a frame grabber. After cropping and scaling to the required size, histogram equalization was performed on all images to reduce sensitivity to changes in illumination conditions. The positive examples contain images of different kinds of cars against a variety of backgrounds, and include images of partially occluded cars. The negative examples include images of natural scenes, buildings and road views. Note that our training set is relatively small and all images in our data set are natural; no synthetic training images are used to simplify the learning problem, as has been done, for example, in [13, 18].

Each of these training images is converted into a feature vector as described in Section 2.2. Note that the potential number of features in any vector is very large, since there are 270 different types of parts that may be present, 20 possible relations between each possible pair of parts, and several of the parts and relations may potentially be repeated. However, in any single image, only a very small number of these possible features is actually active. Taking advantage of this sparseness property, we train our classifier using the Sparse Network of Winnows (SNoW) learning architecture [19, 20], which is especially well-suited for such sparse feature representations.[2] SNoW learns a linear function over the feature space using a feature-efficient variation of the Winnow learning algorithm; it allows input vectors to specify only active features, and its sample complexity grows linearly with the number of relevant features and only logarithmically with the total number of potential features. A separate function is learnt over this common feature space for each target class in the classification task. In our task, feature vectors obtained from object training images are taken as positive examples for the object class and negative examples for the non-object class, and vice-versa. Given a new input vector, the learned function corresponding to each class outputs an activation value, which is the dot product of the input vector with the learned weight vector, passed through a sigmoid function to lie between 0 and 1. Classification then takes place via a winner-take-all decision based on these activations (i.e. the class with the highest activation wins). The activation levels have also been shown to provide a robust measure of confidence; we use this property in the final stage as described in Section 2.4 below. Using this learning algorithm, the representation learned for an object class is a linear threshold function over the feature space, i.e. over the part and relation features.

2.4 Detection Hypothesis Using the Learned Classifier

Having learned a classifier[3] that can classify 100×40 images as positive or negative, cars can be detected in an image by moving a 100×40 window over the image and classifying each such window as positive or negative. However there is one issue that needs to be resolved in doing this.

[2] Software for SNoW is freely available at `http://L2R.cs.uiuc.edu/~cogcomp/`

[3] The SNoW parameters we used to train the classifier were 1.25, 0.8, 4.0 and 1.0 respectively for the promotion and demotion rates, the threshold and the default weight.

It is clear that in the vicinity of an object in the image, several windows will be classified as positive, giving rise to multiple detections corresponding to a single object in the scene. The question that arises is how the system should be evaluated in the presence of these multiple detections. In much previous work in object detection, multiple detections output by the system are all considered to be correct detections (provided they satisfy the criteria for a correct detection; this is discussed later in Section 3.1). However, going back to the requirements specified in Section 1.2, such a system fails not only to locate the objects in the image, but also to form a correct hypothesis about the number of object instances present in the image. Therefore in using a classifier to perform detection, it is necessary to have another processing step, above the level of the classifier output, to produce a coherent detection hypothesis.

A few studies have attempted to develop such a processing step. [10] uses a simple strategy: detected windows are partitioned into disjoint (non-overlapping) groups, and each group gives a single detection. While this may be suitable for the face detection database used there, in general, imposing a zero-overlap constraint on detected windows may be too strong a condition.

As a more general solution to this problem, we introduce the notion of a *classifier activation map* (CAM), which can be generated from any classifier that can produce an activation or confidence value in addition to a binary classification output. Given a test image, a 100×40 window is moved over the image and the learned classifier is applied to each such window (represented as a feature vector) in the image. However, instead of using the binary classification output of the classifier, we take advantage of the activation values it produces. Negatively classified windows are mapped to a zero activation value. Windows classified as positive are mapped to the activation value produced by the classifier. This produces a map with high activation values at points where the classifier has a high confidence in its positive classification. Our algorithm analyzes this map and finds the peaks at which the activation is highest in some neighbourhood, giving the desired locations of the target object. Different neighbourhoods can be used for this purpose. Our algorithm searches for activation peaks in a rectangular neighbourhood, defined by two parameters *hNbhd* and *wNbhd*, so that a point (i_0, j_0) is considered to be an object location if

(i) activation$(i_0, j_0) \geq$ activation(i, j) for all $(i, j) \in N$, where $N = \{(i, j) : |i - i_0| \leq$ *hNbhd*, $|j - j_0| \leq$ *wNbhd*$\}$, and
(ii) no other point in N has already been declared an object location (the map is analyzed in row-wise order).

The neighbourhood size determines the extent of overlap that is allowed between detected windows, and can be chosen appropriately depending on the object class and window size. In our experiments, we used *hNbhd* = 40 pixels, *wNbhd* = 35 pixels.

There is a trade-off between the number of correct detections and number of false detections. An activation threshold is introduced in the algorithm to determine where to lie on this trade-off curve. All activations in the CAM that fall below the threshold are set to zero. Lowering the threshold increases the correct detections but also increases the false positives; raising the threshold has the opposite effect. Figure 5 shows a thresholded CAM generated from a sample test image, and the associated detection result.

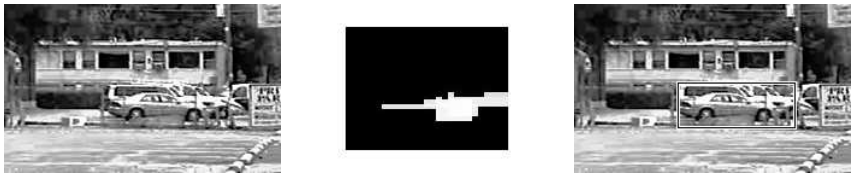

Fig. 5. The center image shows the classifier activation map (CAM) generated from the test image on the left using an activation threshold of 0.85: all activations below 0.85 have been set to zero. The activations in the map have been scaled by 255 to produce the image; the bright white peak corresponds to the highest activation, producing the detection result shown in the right image. The method prevents the system from producing multiple detections for a single object.

3 Evaluation

To evaluate the performance of the system, we collected a set of 170 test images containing 200 cars in all. These are all distinct from the training set. The images were acquired in the same way as the training images. They are of different resolutions and include instances of partially occluded cars, cars that have low contrast with the background, and images with highly textured backgrounds.

Before presenting our results, we discuss some important methodological issues related to evaluation and performance measurement methods in object detection.

3.1 Evaluation Scheme

Past work on object detection has often emphasized the need for standardized data sets to allow a fair comparison of different methods. Although several studies have reported results on common data sets, it is often not clear how the performance of the different methods has been *evaluated* on these data sets. Problems such as image classification have a naturally defined evaluation criterion associated with them. However, in object detection, there is no such natural criterion: correct detections and false detections can be defined in different ways which may give rise to different results. To ensure that the comparison between different methods is truly fair, it is essential that the same evaluation scheme be used. Therefore in addition to standard data sets for object detection, we also need appropriate standardized evaluation schemes to be associated with them.

We describe in detail here the scheme used to evaluate the accuracy of our system.[4]

For each car in the test images, we manually determined the location $(i_{\text{true}}, j_{\text{true}})$ of the best 100×40 window containing the car. For a location $(i_{\text{det}}, j_{\text{det}})$ output by the detector to be considered a correct detection, we require three conditions to be satisfied:

(i) $|i_{\text{true}} - i_{\text{det}}| \leq \delta_{\text{height}}$,
(ii) $|j_{\text{true}} - j_{\text{det}}| \leq \delta_{\text{width}}$, and
(iii) the windows at the detected and true locations have an overlap of at least θ_{area}.

[4] Both the data set we have used and the evaluation routine are available at `http://L2R.cs.uiuc.edu/~cogcomp/`.

The parameters δ_{height}, δ_{width} and θ_{area} are chosen appropriately depending on the target object class and window size, such that these conditions together ensure that no false detection is counted as a correct detection. In our experiments, we used $\delta_{\text{height}} = 14$ pixels, $\delta_{\text{width}} = 28$ pixels and $\theta_{\text{area}} = 50\%$.

In addition, if two or more detected windows satisfy these conditions for the same object location, only one is considered a correct detection; the others are counted as false positives. (See Section 2.4 for discussion.)

3.2 Measuring Performance

In measuring the accuracy of a detection system, the two quantities of interest are clearly the correct detections which we wish to maximize, and the false detections that we wish to minimize. Different methods for reporting and interpreting results present this trade-off in different ways, and again it is necessary to identify a suitable method that captures the trade-off correctly in the context of the object detection problem.

One method for expressing this trade-off is the Receiver Operating Characteristics (ROC) curve. This plots the positive detection rate vs. the false detection rate, where

$$\text{(1)} \quad \textit{Positive Detection Rate} = \frac{\textit{Number of correct positives}}{\textit{Total number of positives in data set}}$$

$$\text{(2)} \quad \textit{False Detection Rate} = \frac{\textit{Number of false positives}}{\textit{Total number of negatives in data set}}$$

However, note that in the problem of object detection, the number of negatives in the data set (required in (2) above) is not well-defined. The number of negative windows evaluated by the detection system has commonly been used for this purpose. However, there are two problems with this approach. The first is that this measures the accuracy of the system as a *classifier*, not as a *detector*. Since the number of negative windows is typically very large compared to the number of positive windows, this allows a large one-sided error: a large absolute number of false detections appears to be small under this measure. The second and more fundamental problem is that the number of negative windows evaluated is not a property of the input to the problem, but rather a property internal to the implementation of the detection system.

When a detection system is put into practice, we are interested in knowing how many of the objects it detects, and how often the detections it makes are false. This trade-off is captured more accurately by a variation of the recall-precision curve, where

$$\text{(3)} \quad \textit{Recall} = \frac{\textit{Number of correct positives}}{\textit{Total number of positives in data set}} \quad (= \textit{Positive Detection Rate} \text{ above})$$

$$\text{(4)} \quad \textit{Precision} = \frac{\textit{Number of correct positives}}{\textit{Number of correct positives + Number of false positives}}$$

The first quantity of interest, namely the proportion of objects that are detected, is given by the recall. The second quantity of interest, namely the number of false detections relative to the total number of detections made by the system, is given by

$$\text{(5)} \quad 1 - \textit{Precision} = \frac{\textit{Number of false positives}}{\textit{Number of correct positives + Number of false positives}}$$

Plotting *recall* vs. ($1 - $ *precision*) therefore expresses the desired trade-off.

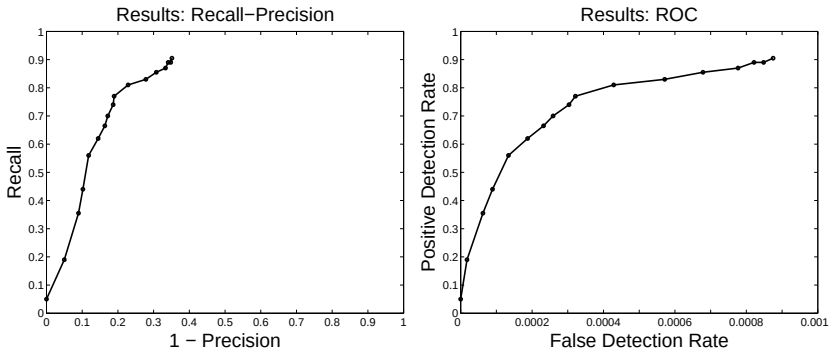

Fig. 6. Left: Recall-precision curve showing the performance of our car detection system. **Right:** ROC curve showing the same results. (Important: note that the X-axis scales in the two curves are different; the X-axis values in the ROC curve are much smaller than in the recall-precision curve.) See Section 3.2 for definitions of the different parameters and a discussion of why the recall-precision curve is a more appropriate method for expressing object detection results.

3.3 Results

We applied our detector to the test set (described earlier) of 170 images containing 200 cars. To reduce computational costs, the 100×40 window was moved in steps of size 5% of the window size in each dimension, i.e. steps of 5 pixels and 2 pixels respectively in the horizontal and vertical directions in our experiments. In all, 147,802 windows were evaluated by the system, of which approximately 112,000 were negative.

Following the discussion in Section 3.2, we present our results as *recall* vs. $(1 - \textit{precision})$ in Figure 6. The different points on the curve are obtained by varying the activation threshold parameter as described in Section 2.4. We also calculate the ROC curve as has been done before (using the number of negative windows evaluated by the system as the total number of negatives) for comparison; this is also shown in Figure 6.

Table 1 shows some sample points from the recall-precision curve of Figure 6. Again, for comparison, we also show the false detection rate at each point as determined by the ROC curve. Training over 1000 images takes less than half an hour in our implementation. The average time to test a 200×150 image is 8 seconds.

Activation threshold	**No. of correct detections**, N	**Recall** $\frac{N}{200}$	**No. of false detections**, M	**Precision** $\frac{N}{N+M}$	**False detection rate**, $\frac{M}{112000}$
0.55	181	90.5%	98	64.9%	0.09%
0.65	178	89.0%	92	65.9%	0.08%
0.75	171	85.5%	76	69.2%	0.07%
0.85	162	81.0%	48	77.1%	0.04%
0.90	154	77.0%	36	81.1%	0.03%
0.95	140	70.0%	29	82.8%	0.03%

Table 1. Accuracy of our car detection system on the test set containing 200 cars.

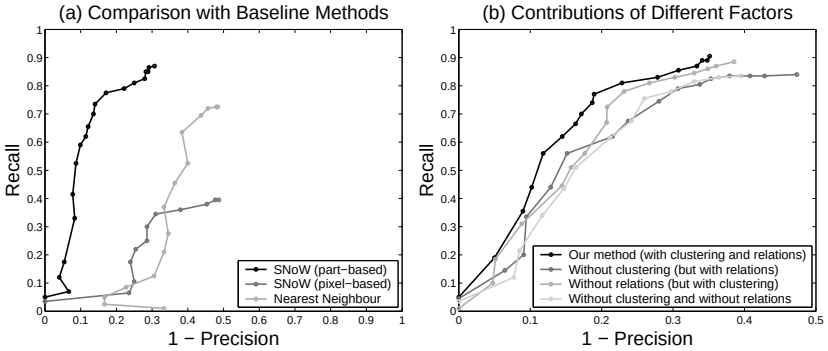

Fig. 7. (a) Comparison with baseline methods. The poor performance of the baseline methods is an indicator of the difficulty level of our test set. In addition, the poor performance of the pixel-based detector that uses the same learning procedure as ours, and differs only in the representation scheme, shows the importance of choosing a good representation. **(b)** Contributions of different elements of our method to the overall performance. Both the part clustering step and the relation features have significant contributions to the effectiveness of our representation. (Important: note that the X-axis range in this graph is [0,0.5] for clarity.)

As baselines for comparison, we implemented a SNoW-based detector using simply pixel intensities as features, and a nearest-neighbour based detector that uses the normalized correlation between test windows and training images as the similarity measure. The CAM for the SNoW-based method was computed as before, using SNoW activations. In the case of nearest-neighbour, the classifier activation for a test window was taken to be the correlation of the window with the nearest training image. The results are shown in Figure 7(a).[5] The poor performance of the baseline detectors is an indicator of the difficulty level of our test set: for the COIL object database, nearest-neighbour gives above 95% recognition accuracy, while on the face image database in [13], the pixel-based SNoW method achieves above 94% detection accuracy.

To gain a better understanding of the different factors contributing to the success of our approach, we conducted experiments in which we eliminated certain steps of our method. The results of these experiments are shown in Figure 7(b). In the first experiment, we eliminated the part clustering step when constructing the part vocabulary, assigning a different feature id to each of the 400 parts. This resulted in a significant decrease in performance, confirming our intuition that representing similar parts as a single conceptual-level part is important for the learning algorithm to generalize well. In the second experiment, we retained the clustering step, but did not use the relation features, representing the images simply by the parts present in them. Again, this showed a decrease in performance, suggesting that important information is captured by the relations. Finally, in the third experiment, we eliminated both the clustering step and the relation features. In this case the decrease in performance was not significant relative to

[5] The pixel-based method has extremely high computational and space costs. Therefore in all experiments shown in Figure 7(a), the 100×40 window was shifted over the images in larger steps of 10 and 4 pixels respectively in the horizontal and vertical directions, with *hNbhd* = *wNbhd* = 40 pixels (see Section 2.4).

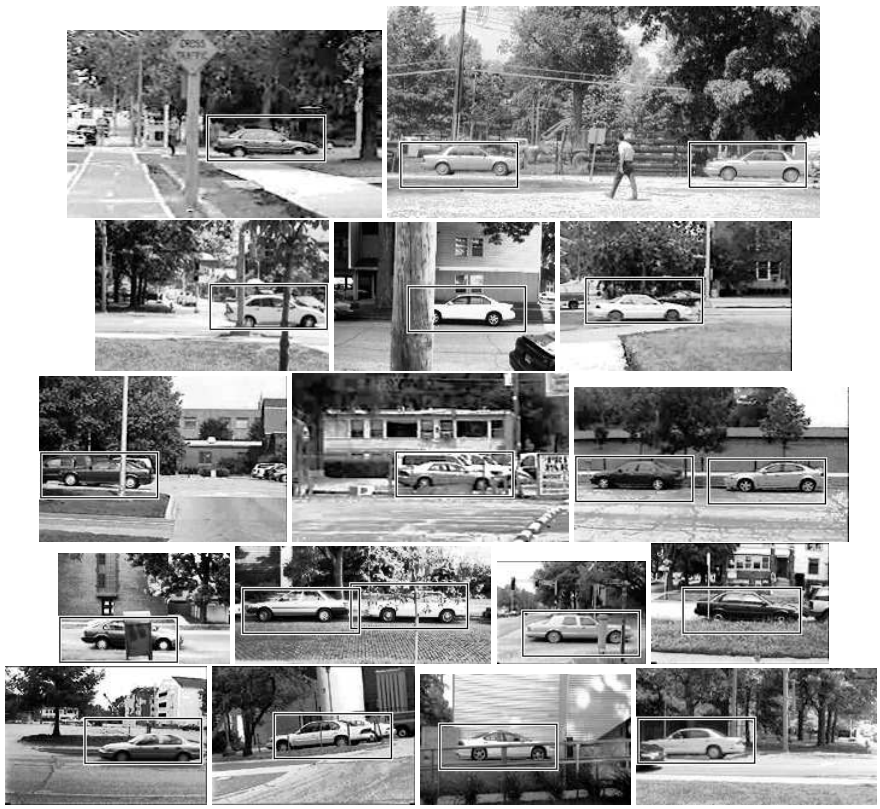

Fig. 8. Examples of test images on which our system achieved perfect detection results. An activation threshold of 0.9 was used. The windows are drawn by a separate evaluator program at the *exact* locations output by the system (see Section 2.4). Also see Figure 9.

the first experiment, in which only clusters were eliminated. This can be explained on the basis that, when the parts are all considered to be distinct, the algorithm sees many different parts in the training set, and therefore the relation features defined over them are also different; as a result, in this case the relations do not help in generalizing.

We also tested the intuition that a small number of parts should suffice for successful detection by ignoring all one-part clusters in the vocabulary. However, this decreased the performance, suggesting that the small clusters also play an important role.

Figures 8 and 9 show the output of our detector on some sample test images.

4 Conclusions and Future Work

We have presented an approach for learning a sparse, part-based representation for object detection. Expressive features are acquired automatically and capture information about the parts in an image and the spatial relations among them. An efficient learning algorithm over this feature space then learns a good classifier from a small training set.

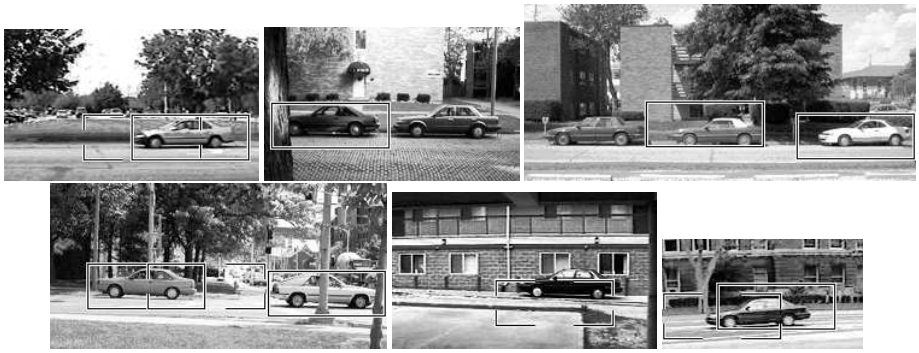

Fig. 9. Examples of test images on which our system missed some of the objects or produced false detections (using the same activation threshold as in Figure 8). The evaluator program draws a window at *each* location output by the detector; fully drawn windows represent correct detections, while broken windows represent false positives.

The notion of a classifier activation map is introduced to form a good detector from this learned classifier.

We have shown that our method works successfully on a difficult test set of images containing side views of cars. We achieve high detection rates on real-world images with a high degree of clutter and occlusion. Our framework is easily extensible to other objects that have distinguishable parts in a relatively fixed spatial configuration.

We addressed several methodological issues that are important in evaluating object detection approaches. First, the distinction between classification and detection was highlighted, and a general method for producing a good detector from a classifier was developed. Second, we discussed the importance of specifying and standardizing evaluation schemes in object detection experiments. Finally, we argued that object detection approaches should be compared using recall-precision curves rather than measures such as ROC curves that are biased in ways that depend on the system implementation. Several of these issues have been addressed in (different) ad-hoc ways in earlier works, which may mask important differences that exist among approaches, and we believe that it is necessary to include them as an integral part of the evaluation process.

Our work can be extended in several directions. The computational costs of the current approach are relatively high; one way to make it appropriate for real-time applications can be to develop a fast filter with one-sided error that can quickly filter out regions unlikely to contain an object, following which our detector can be applied to more promising regions. The system can be made scale invariant by processing test images at multiple scales, and extending the classifier activation map to incorporate activation information from different scales. Orientation invariance can also be achieved via an exhaustive approach as in [18]. To avoid increasing computational costs, these multiple steps can be parallelized. At the learning level, a natural extension is to learn to detect several object classes at the same time. Learning multiple classifiers over different features and combining information from them may lead to improved performance. It would also be interesting to formulate a learning problem that directly addresses the problem of detection rather than classification.

Acknowledgements

We would like to thank Ashutosh Garg, David Kriegman and Cordelia Schmid for several helpful discussions and comments on an earlier version of the paper. This work was supported by NSF grants ITR IIS 00-85980 and ITR IIS 00-85836.

References

1. Logothetis, N.K., Sheinberg, D.L.: Visual object recognition. Ann. Rev. Neurosci. **19** (1996) 577–621
2. Palmer, S.E.: Hierarchical structure in perceptual representation. Cognitive Psychology **9** (1977) 441–474
3. Wachsmuth, E., Oram, M.W., Perrett, D.I.: Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque. Cerebral Cortex **4** (1994)
4. Biederman, I.: Recognition by components: a theory of human image understanding. Psychol. Review **94** (1987) 115–147
5. Colmenarez, A.J., Huang, T.S.: Face detection with information-based maximum discrimination. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (1997) 782–787
6. Rowley, H.A., Baluja, S., Kanade, T.: Neural network-based face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 23–38
7. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: an application to face detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (1997) 130–136
8. Amit, Y., Geman, D.: A computational model for visual selection. Neural Computation **11** (1999) 1691–1715
9. Roth, D., Yang, M.H., Ahuja, N.: Learning to recognize 3d objects. Neural Computation **14** (2002) To appear
10. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2001)
11. Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. IEEE Trans. on Pattern Analysis and Machine Intelligence **23** (2001) 349–361
12. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: Proceedings of the Sixth European Conference on Computer Vision. (2000) 18–32
13. Yang, M.H., Roth, D., Ahuja, N.: A SNoW-based face detector. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: Advances in Neural Information Processing Systems 12. (2000) 855–861
14. Moravec, H.P.: Towards automatic visual obstacle avoidance. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence. (1977)
15. Schmid, C., Mohr, R.: Local greyvalue invariants for image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence **19** (1997) 530–535
16. Haralick, R.M., Shapiro, L.G.: Computer and Robot Vision II. Addison-Wesley (1993)
17. Grimson, W.E.L., Lozano-Perez, T.: Recognition and localization of overlapping parts from sparse data in two and three dimensions. In: Proceedings of the IEEE International Conference on Robotics and Automation. (1985) 61–66
18. Schneiderman, H., Kanade, T.: A statistical method for 3D object detection applied to faces and cars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2000) 746–751
19. Carlson, A.J., Cumby, C., Rosen, J., Roth, D.: The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department (1999)
20. Roth, D.: Learning to resolve natural language ambiguities: A unified approach. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence. (1998) 806–813