Learning to Detect Objects in Images via a Sparse, Part-Based Representation

Shivani Agarwal, Aatif Awan, and Dan Roth, *Member*, *IEEE Computer Society*

Abstract—We study the problem of detecting objects in still, gray-scale images. Our primary focus is the development of a learning-based approach to the problem that makes use of a sparse, part-based representation. A vocabulary of distinctive 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, together with spatial relations observed among the parts. Based on this representation, a learning algorithm is used to automatically learn to detect instances of the object class in new images. The approach can be applied to any object with distinguishable parts in a relatively fixed spatial configuration; it is evaluated here on difficult sets of real-world images containing side views of cars, and is seen to successfully detect objects in varying conditions amidst background clutter and mild occlusion. In evaluating object detection approaches, several important methodological issues arise that have not been satisfactorily addressed in previous work. A secondary focus of this paper is to highlight these issues and to develop rigorous evaluation standards for the object detection problem. A critical evaluation of our approach under the proposed standards is presented.

Index Terms—Object detection, image representation, machine learning, evaluation/methodology.

1 INTRODUCTION

The development of methods for automatic detection of objects in images has been a central challenge in computer vision and pattern analysis research. The main difficulty in developing a reliable object detection approach arises from the wide range of variations in images of objects belonging to the same object class. Different objects belonging to the same category often have large variations in appearance. In addition, the same object can appear vastly different under different viewing conditions, such as those resulting from changes in lighting, viewpoint, and imaging techniques [1]. A successful object detection approach must therefore be able to represent images in a manner that renders them invariant to such intraclass variations, but at the same time distinguishes images of the object class from all other images.

In this paper, we present an approach for learning to detect objects in images using a sparse, part-based representation. Part-based representations for object detection form the basis for a number of theories of biological vision [2], [3], [4], [5], and have also been shown to offer advantages in computational approaches [6]. In the approach presented here, the part-based representation is acquired automatically from a set of sample images of the object class of interest, thus capturing the variability in part appearances across the sample set. A classifier is then trained, using machine learning techniques, to distinguish between object and nonobject images based on this representation; this learning stage further captures the variation in the part structure of object images across the training set. As shown in our experiments, the resulting algorithm is able to accurately detect objects in complex natural scenes.

This paper also discusses several methodological issues that arise when evaluating object detection approaches. For an area that is increasingly becoming an active focus of research, it is necessary to have standardized and meaningful methods for evaluating and comparing different approaches. We identify some important issues in this regard that have not been satisfactorily addressed in previous work, and propose possible solutions to them.

1.1 Related Work

A number of different approaches to object detection that use some form of learning have been proposed in the past. In most such approaches, images are represented using some set of features, and a learning method is then used to identify regions in the feature space that correspond to the object class of interest. There has been considerable variety in both the types of features used and the learning methods applied; we briefly mention some of the main approaches that have been proposed and then discuss some recent methods that are most closely related to ours.

Image features used in learning-based approaches to object detection have included raw pixel intensities [7], [8], [9], features obtained via global image transformations [10], [11], and local features such as edge fragments [12], [13], rectangle features [14], Gabor filter-based representations [15], and wavelet features [16]. On the learning side, methods for classifying the feature space have ranged from simple nearest-neighbor schemes to more complex approaches such as neural networks [8], convolutional neural networks [17], probabilistic methods [11], [18], and linear or higher degree polynomial classifiers [13], [16].

In our approach, the features are designed to be object parts that are rich in information content and are specific to

• *The authors are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Ave., Urbana, IL 61801. E-mail: {sagarwal, mawan, danr}@uiuc.edu.*

Manuscript received 8 May 2003; revised 12 Mar. 2004; accepted 1 Apr. 2004. Recommended for acceptance by R. Basri.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0069-0503.

the object class of interest. A part-based representation was used in [6], 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 a person is present. However, the approach in [6] 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. One approach in this direction is developed in [19], [20], in which a large set of candidate parts is extracted from a set of sample images of the object class of interest, an explicit measure of information content is computed for each such candidate, and the candidates found to have the highest information content are then used as features. This framework is appealing in that it naturally allows for parts of different sizes and resolutions. However, the computational demands are high; indeed, as discussed in [20], after a few parts are chosen automatically, manual intervention is needed to guide the search for further parts so as to keep the computational costs reasonable. Our method for automatically selecting information-rich parts builds on an efficient technique described in [21], in which interest points are used to collect distinctive parts. Unlike [21], however, our approach does not assume any probabilistic model over the parts; instead, a discriminative classifier is directly learned over the parts that are collected. In addition, the model learned in [21] relies on a small number of fixed parts, making it potentially sensitive to large variations across images. By learning a classifier over a large feature space, we are able to learn a more expressive model that is robust to such variations.

In order to learn to identify regions in the feature space corresponding to the object class of interest, we make use of a feature-efficient learning algorithm that has been used in similar tasks in [13], [22]. However, [13], [22] use a pixel-based representation, whereas in our approach, images are represented using a higher-level, 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 (such as 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 as input, returns as output a list of locations (and, if applicable, corresponding scales) at which instances of the object class are detected in the image. It is important to note that this problem is distinct from (and more challenging than) the commonly studied problem of simply deciding whether or not an input image contains an instance of the object class; the latter problem requires only a "yes/no" output without necessarily localizing objects, and is therefore really an instance of an image classification problem rather than a detection problem. Evaluation criteria for the detection problem are discussed later in the paper.

1.3 Overview of the Approach

Our approach for learning to detect objects consists broadly of four stages; these are outlined briefly below:

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 patches from sample images of the object class of interest. Similar patches thus obtained are grouped together and treated as a single part.
2. *Image Representation*. Input images are represented in terms of parts from the vocabulary obtained in the first stage. This requires determining which parts from the vocabulary are present in an image; a correlation-based similarity 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 (nonobject), each image is converted into 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 nonmember 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 detector. We develop the notion of a *classifier activation map* in the single-scale case (when objects are sought at a single, prespecified scale), and a *classifier activation pyramid* in the multiscale case (when objects are sought at multiple scales); these are generated by applying the classifier to windows at various locations in a test image (and, in the multiscale case, at various scales), each window being represented as a feature vector as above. We present two algorithms for producing a good detection hypothesis using the activation map or pyramid obtained from an image.

The proposed 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, this object class will be used as a running example throughout the paper to illustrate the ideas and techniques involved.

The rest of the paper is organized as follows: Section 2 describes each of the four stages of our approach in detail. Section 3 presents an experimental evaluation of the approach. In this section, we first discuss several important methodological issues, including evaluation criteria and performance measurement techniques, and then present our experimental results. In Section 4, we analyze the performance of individual components of our approach; this gives some insight into the results described in

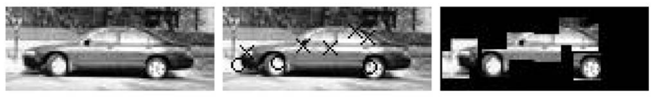


Fig. 1. (a) A sample object image used in vocabulary construction. (b) Interest points detected by the Förstner operator. Crosses denote intersection points; circles denote centers of circular patterns. (c) Patches extracted around the interest points.

Section 3. Finally, Section 5 concludes with a summary and possible directions for future work.

2 Approach

As outlined in the previous section, our approach for learning to detect objects consists broadly of four stages. Below, we describe each of these stages in detail.

2.1 Vocabulary Construction

The first stage in the approach is to develop a vocabulary of parts with which to represent images. To obtain an expressive representation for the object class of interest, we require distinctive parts that are specific 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 [21].

Interest points are points in an image 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 [23], object recognition, and image retrieval [24]. 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 certain degree of conceptual invariance.

We apply the Förstner interest operator [25], [21] to a set of representative images of the object class; this detects intersection points of lines and centers of circular patterns. Small image patches are then extracted around the interest points obtained. The goal of extracting a large set of patches 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 these patches.

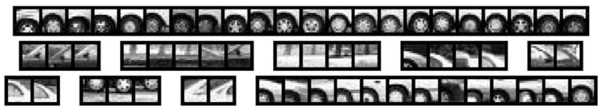


Fig. 3. Examples of some of the "part" clusters formed after grouping similar patches together. These form our part vocabulary.

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

As seen in Fig. 2, several of the patches extracted by this procedure are visually very similar to each other. To facilitate learning, it is important to abstract over these patches by mapping similar patches to the same feature id (and distinct patches to different feature ids). This is achieved via a bottom-up clustering procedure. Initially, each patch 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 patches:

$$\text{similarity}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{p_1 \in C_1} \sum_{p_2 \in C_2} \text{similarity}(p_1, p_2),$$

where the similarity between two patches is measured by normalized correlation, allowing for small shifts of up to two pixels. Two clusters are merged if the similarity between them exceeds a certain threshold (0.80 in our implementation). Using this technique, the 400 patches were grouped into 270 "part" clusters. While several clusters contained just one element, patches with high similarity were grouped together. Fig. 3 shows some of the larger clusters that were formed. Each cluster as a whole is then given a single feature id and treated as a single "conceptual" part. In this way, by using a deliberately redundant representation that uses several similar patches to represent a single conceptual part, we are able to extract a higher-level, conceptual representation from the interest points. The importance of the clustering procedure is demonstrated by experiments described in Section 3.4.3.

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 then representing the image as a binary feature

Fig. 2. The 400 patches extracted by the Förstner interest operator from 50 sample images.

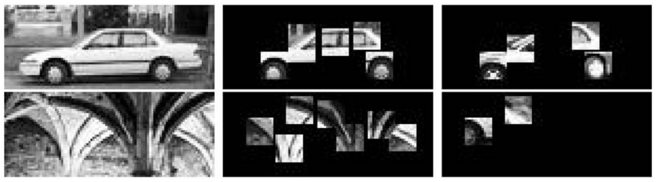


Fig. 4. Examples of the part detection process applied to a positive image (top row) and a negative image (bottom row) during training. Center images show the patches highlighted by the interest operator; notice how this successfully captures the interesting regions in the image. These highlighted interest 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.

vector based on these detected parts and the spatial relations that are observed among them.

2.2.1 Part Detection

Since the vocabulary parts are all based on interest points, the search for parts in an image is restricted to interesting regions by first applying the same interest operator to the image and highlighting patches around the interest points found. For each such patch in the image, we perform a similarity-based indexing into the part vocabulary. The similarity of a vocabulary part P (which may be a cluster containing several patches) to a highlighted patch q is computed as

$$\text{similarity}(P, q) = \frac{1}{\lceil \lambda |P| \rceil} \sum_{p \in P_{(\lambda,q)}} \text{similarity}(p, q),$$

where $0 < \lambda \leq 1$, and $P_{(\lambda,q)}$ denotes the subset of the part cluster P that contains the $\lceil \lambda |P| \rceil$ patches in P that are most similar to q. (In our implementation, $\lambda = 0.5$.) The similarity between patches p and q is measured by normalized correlation, allowing for small shifts of up to two pixels. For each highlighted patch q, the most similar vocabulary part $P^*(q)$ is given by

$$P^*(q) = \arg\max_{P} \; \text{similarity}(P, q).$$

If a sufficiently similar vocabulary part is found, i.e., if $\text{similarity}(P^*(q), q)$ exceeds a certain threshold (0.75 in our implementation), then the patch q in the image is represented by the feature id corresponding to the vocabulary part $P^*(q)$. Fig. 4 shows examples of this process.

2.2.2 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. The distances and directions are discretized into bins: In our implementation, the distances are defined relative to the part size and are discretized into five bins, while the directions are discretized into eight different ranges, each covering an angle of 45°. By considering the parts in a fixed order across the image, the number of direction bins that need to be represented is reduced to four. 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 around two to six parts, while a negative one contains around zero to four. Therefore, the cost of computing relations between all pairs of detected parts is negligible once the parts have been detected.

2.2.3 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:

1. $P_n^{(i)}$, 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).
2. $R_m^{(j)}(P_{n_1}, P_{n_2})$, 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 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.[1] The rerepresentation of the image is a list of the identifiers corresponding to the features that are *active* (present) in the image.

2.3 Learning a Classifier

Using the above feature vector representation, a classifier is trained to classify a 100×40 image as car or noncar. We used a training set of 1,000 labeled images (500 positive and 500 negative), each 100×40 pixels in size.[2] The images were acquired partly by taking still photographs of parked cars and partly by grabbing frames from digitized video sequences of cars in motion. The photographs and video sequences were all taken in the Champaign-Urbana area. 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 training 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; we do not use any synthetic training images, as has been done, for example, in [8], [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

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. Note that the 50 car images used for constructing the part vocabulary are not part of the training set.

advantage of this sparseness property, we train our classifier using the Sparse Network of Winnows (SNoW) learning architecture [26], [27], which is especially well-suited for such sparse feature representations.[3] SNoW learns a linear function over the feature space using a variation of the feature-efficient Winnow learning algorithm [28]; it allows input vectors to specify only active features, and as is the case for Winnow, 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 learned over the 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 nonobject 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[4] 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, due to the invariance of the classifier to small translations of an object, several windows in the vicinity of an object in the image will be classified as positive, giving rise to multiple detections corresponding to a single object in the scene. A 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 criterion for a correct detection; this is discussed later in Section 3.2). However, such a system fails both to locate the objects in the image, and 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 classification output, to produce a coherent detection hypothesis.

A few studies have attempted to develop such a processing step. A simple strategy is used in [14]: Detected windows are partitioned into disjoint (nonoverlapping) groups, and each group gives a single detection located at the centroid of the corresponding original detections. 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. The system in [8] uses the very property of multiple detections to its advantage, taking the number of detections in a small neighborhood as a measure of the detector's confidence in the presence of an object within the neighborhood; if a high confidence is obtained, the multiple detections are collapsed into a single detection located at the centroid of the original detections. Our approach also uses a confidence measure to correctly localize an object; however, this confidence is obtained directly from the classifier. In addition, our approach offers a more systematic method for dealing with overlaps; like [14], [8] also uses a zero-overlap strategy, which is too restrictive for general object classes.

As a more general solution to the problem, we develop the notion of a *classifier activation map* in the single-scale case when objects are sought at a single, prespecified scale, and a *classifier activation pyramid* in the multiscale case when objects are sought at multiple scales. These can be generated from any classifier that can produce a real-valued activation or confidence value in addition to a binary classification output.

2.4.1 Classifier Activation Map for Single-Scale Detections

In the single-scale case (where, in our case, cars are sought at a fixed size of 100×40 pixels), a fixed-size window (of size 100×40 pixels in our case) is moved over the image and the learned classifier is applied to each such window (represented as a feature vector) in the image. Windows classified as negative 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. This map can then be analyzed to find high-activation peaks, giving the desired object locations.

We propose two algorithms for analyzing the classifier activation map obtained from a test image. The first algorithm, which we refer to as the *neighborhood suppression* algorithm, is based on the idea of nonmaximum suppression. All activations in the map start out as "unsuppressed." At each step, the algorithm finds the highest unsuppressed activation in the map. If this activation is the highest among all activations (both suppressed and unsuppressed) within some predefined neighborhood, then the location of the corresponding window is output as a detection, and all activations within the neighborhood are marked as "suppressed;" this means they are no longer considered as candidates for detection. If the highest unsuppressed activation found is lower than some (suppressed) activation within the neighborhood, it is simply marked as suppressed. The process is repeated until all activations in the map are either zero or have been suppressed. The shape and size of the neighborhood used can be chosen appropriately depending on the object class and window size. In our experiments, we used a rectangular neighborhood of size 71 pixels (width) $\times$ 81 pixels (height), centered at the location under consideration.

Our second algorithm for analyzing the classifier activation map obtained from a test image is referred to as the

3. Software for SNoW is freely available from http://L2R.cs.uiuc.edu/~cogcomp/.

4. The SNoW parameters 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.

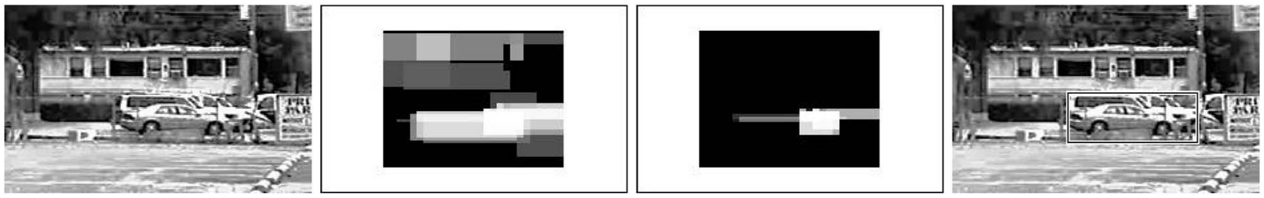


Fig. 5. The second image shows the classifier activation map generated from the test image on the left; the activation at each point corresponds to the confidence of the classifier when applied to the 100×40 window centered at that point. The activations in the map have been scaled by 255 to produce the image; black corresponds to an activation of 0, white to an activation of 1. The third image shows the map after applying a threshold of 0.9: All activations below 0.9 have been set to zero. The activations in this map have been rescaled; the activation range of 0.9-1 is now represented by the full black-white range. The bright white peak corresponds to the highest activation, producing the detection result shown on the right. The method prevents the system from producing multiple detections for a single object.

repeated part elimination algorithm. This algorithm finds the highest activation in the map and outputs the location of the corresponding window as a detection. It then removes all parts that are contained in this window, and recomputes the activation map by reapplying the classifier to the affected windows. This process is then repeated until all activations in the map become zero. This algorithm requires repeated application of the learned classifier, but avoids the need to determine appropriate neighborhood parameters as in the neighborhood suppression algorithm.

In both algorithms, there is a trade-off between the number of correct detections and number of false detections. An activation threshold is introduced in the algorithms to determine where to lie on this trade-off curve; all activations in the classifier activation map that fall below the threshold are automatically set to zero. Lowering the threshold increases the correct detections but also increases the false positives; raising the threshold has the opposite effect. Fig. 5 shows the classifier activation map generated from a sample test image, the map after applying a threshold, and the associated detection result (obtained using the neighborhood suppression algorithm).

2.4.2 Classifier Activation Pyramid for Multiscale Detections

The approach described above for detecting objects at a single scale can be extended to detect objects at different scales in an image by processing the image at several scales. This can be done by scaling the input image a number of times to form a multiscale image pyramid, applying the learned classifier to fixed-size windows in each image in the pyramid, and forming a three-dimensional classifier activation pyramid instead of the earlier two-dimensional classifier activation map. This activation pyramid can then be analyzed to detect objects in both location and scale (analogous to finding peaks corresponding to object locations in the two-dimensional map). In our multiscale experiments, a test image is scaled to sizes ranging from 0.48 to 1.2 times the original size, each scale differing from the next by a factor of 1.2. The learned classifier is applied to 100×40 windows in each of the scaled images, resulting in a classifier activation pyramid with six scale levels.

Both the neighborhood suppression algorithm and the repeated part elimination algorithm used to analyze activation maps in the single-scale case can be extended naturally to analyze activation pyramids in the multiscale case. In this case, the algorithms output both the location and the scale of the window corresponding to an activation flagged as a detection. The neighborhood suppression algorithm now uses a three-dimensional neighborhood that extends across all scales; the neighborhood size at each scale is obtained by scaling the original neighborhood size with the image. Similarly, the repeated part elimination algorithm now removes parts at all scales that arise from the region of an image contained within the window corresponding to an activation flagged as a detection. Again, an activation threshold is introduced in the algorithms to determine where to lie in the trade-off between correct detections and false detections.

3 Evaluation

This section presents an experimental evaluation of the object detection approach developed in the previous section. The approach is evaluated both for the single-scale case and for the multiscale case. We start by describing the data sets used for evaluation in Section 3.1. In Sections 3.2 and 3.3, we discuss in detail the evaluation criteria and performance measures we use. We emphasize the importance of identifying and specifying a suitable evaluation methodology and discuss some important issues in this regard that have not been addressed satisfactorily in previous object detection research. Section 3.4 contains our experimental results.

3.1 Test Sets

We collected two sets of test images, the first for the single-scale case and the second for the multiscale case. We refer to these as test set I and test set II, respectively. Test set I consists of 170 images containing 200 cars; the cars in this set are all roughly the same size as in the training images. Test set II consists of 108 images containing 139 cars; the cars in this set are of different sizes, ranging from roughly 0.8 to 2 times the size of cars in the training images. The images were all taken in the Champaign-Urbana area and were acquired in the same manner as the training images: partly from still images taken with a camera and partly by grabbing frames from video sequences of cars in motion. 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.

3.2 Evaluation Criteria

Past work on object detection has often emphasized the need for standardized data sets for comparing different approaches. Although several studies have reported results on common data sets, it is often not clear how the different approaches have 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, giving rising to different results. To ensure that the comparison between different approaches is truly fair, it is essential that the same evaluation criterion be used. Therefore, in addition to standard data sets for object detection, we also need appropriate standardized evaluation criteria to be associated with them. Here, we specify in detail the criteria we have used to evaluate our approach.[5]

In the single-scale case, for each car in the test images, we determined manually the location of the best 100×40 window containing the car. For a location output by the detector to be evaluated as a correct detection, we require it to lie within an ellipse of a certain size centered at the true location. In other words, if (i^*, j^*) denotes the center of the window corresponding to the true location and (i, j) denotes the center of the window corresponding to the location output by the detector, then for (i, j) to be evaluated as a correct detection we require it to satisfy

$$\frac{|i - i^*|^2}{\alpha^2_{\text{height}}} + \frac{|j - j^*|^2}{\alpha^2_{\text{width}}} \leq 1, \tag{1}$$

where $\alpha_{\text{height}}, \alpha_{\text{width}}$ determine the size of the allowed ellipse. We allowed the axes of the ellipse to be 25 percent of the object size along each dimension, thus taking $\alpha_{\text{height}} = 0.25 \times 40 = 10$ and $\alpha_{\text{width}} = 0.25 \times 100 = 25$. In addition, if two or more locations output by the detector satisfy the above criterion for the same object, only one is considered a correct detection; the others are counted as false positives (see Section 2.4 for a discussion on this). The above criterion is more strict than the criterion used in [29], and we have found that it corresponds more closely with human judgement.

In the multiscale case, we determined manually both the location and the scale of the best window containing each car in the test images. Since we assume a fixed ratio between the height and width of any instance of the object class under study, the scale of the window containing an object can be represented simply by its width. The ellipse criterion of the single-scale case is extended in this case to an ellipsoid criterion; if (i^*, j^*) denotes the center of the window corresponding to the true location and w^* its width, and (i, j) denotes the center of the window corresponding to the location output by the detector and w the width, then for (i, j, w) to be evaluated as a correct detection we require it to satisfy

5. Both the data sets we have used and the evaluation routines are available from http://L2R.cs.uiuc.edu/~cogcomp/.

TABLE 1
Symbols Used in Defining Performance Measurement Quantities, Together with Their Meanings

TP	Number of true positives
FP	Number of false positives
nP	Total number of positives in data set
nN	Total number of negatives in data set

$$\frac{|i - i^*|^2}{\alpha^2_{\text{height}}} + \frac{|j - j^*|^2}{\alpha^2_{\text{width}}} + \frac{|w - w^*|^2}{\alpha^2_{\text{scale}}} \leq 1, \tag{2}$$

where $\alpha_{\text{height}}, \alpha_{\text{width}}, \alpha_{\text{scale}}$ determine the size of the allowed ellipsoid. In this case, we allowed the axes of the ellipsoid to be 25 percent of the *true* object size along each dimension, thus taking $\alpha_{\text{height}} = 0.25 \times h^*$, $\alpha_{\text{width}} = 0.25 \times w^*$, and $\alpha_{\text{scale}} = 0.25 \times w^*$, where h^* is the height of the window corresponding to the true location (in our case, $h^* = (40/100)w^*$). Again, if two or more location-scale pairs output by the detector satisfy the above criterion for the same object, only one is considered a correct detection; the others are counted as false positives.

3.3 Performance Measures

In measuring the performance of an object detection approach, the two quantities of interest are clearly the number of correct detections, which we wish to maximize, and the number of false detections, which we wish to minimize. Most detection algorithms include a threshold parameter (such as the activation threshold in our case, described in Section 2.4) which can be varied to lie at different points in the trade-off between correct and false detections. It is then of interest to measure how well an algorithm trades off the two quantities over a range of values of this parameter. Different methods for measuring performance measure this trade-off in different ways and, again, it is important to identify a suitable method that captures the trade-off correctly in the context of the object detection problem.

One method for expressing the trade-off is the receiver operating characteristics (ROC) curve. The ROC curve plots the true positive rate versus the false positive rate, where

$$\text{True positive rate} = \frac{TP}{nP}, \tag{3}$$

$$\text{False positive rate} = \frac{FP}{nN}, \tag{4}$$

the symbols being explained in Table 1. However, note that, in the problem of object detection, the number of negatives in the data set, nN (required in the definition of the false positive rate in (4) above), is not defined. The number of negative windows evaluated by a 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, a large absolute number of false detections appears to be small under this measure.

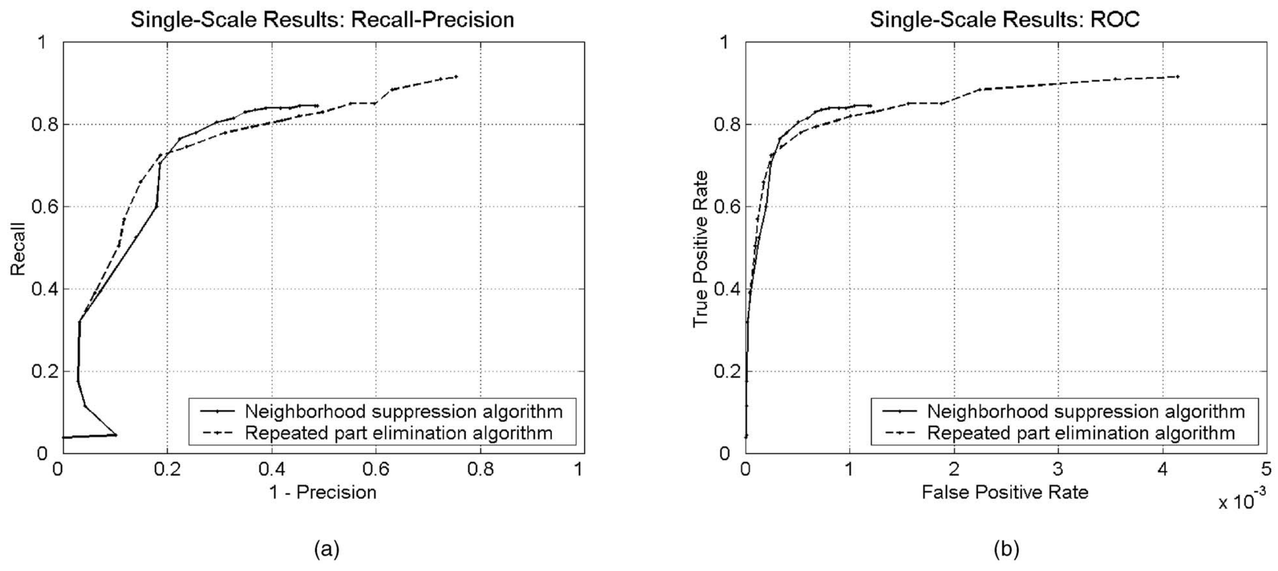


Fig. 6. (a) Recall-precision curves showing the performance of our single-scale car detection system with the two algorithms described in Section 2.4. (b) ROC curves showing the same results. It is important to note that the x-axis scales in the two curves are different; the x-axis values in the ROC curve are much smaller than in the recall-precision curve. Note also that precision need not necessarily decrease monotonically with increasing recall; this is exhibited by the inward bend on the lower left corner of the first curve (consequently, recall is not necessarily a function of precision). See Section 3.3 for definitions of the different quantities and a discussion of why the recall-precision curve is a more appropriate method for expressing object detection results than the ROC curve.

The second and more fundamental problem is that the number of negative windows evaluated is not a property of either the input to the problem or the output, 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

$$\text{Recall} = \frac{TP}{nP}, \tag{5}$$

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{6}$$

The first quantity of interest, namely, the proportion of objects that are detected, is given by the recall (which is the same as the true positive rate in (3) above). 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

$$1 - \text{Precision} = \frac{FP}{TP + FP}. \tag{7}$$

Plotting recall versus $(1 - \text{precision})$, therefore, expresses the desired trade-off.

We shall also be interested in the setting of the threshold parameter that achieves the best trade-off between the two quantities. This will be measured by the point of highest F-measure, where

$$\text{F-measure} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{8}$$

The F-measure summarizes the trade-off between recall and precision, giving equal importance to both.

3.4 Experimental Results

We present our single-scale results in Section 3.4.1 below, followed by a comparison of our approach with baseline methods in Section 3.4.2 and a study of the contributions of different factors in the approach in Section 3.4.3. Our multiscale results are presented in Section 3.4.4.

3.4.1 Single-Scale Results

We applied our single-scale detector, with both the neighborhood suppression algorithm and the repeated part elimination algorithm (see Section 2.4), to test set I (described in Section 3.1), consisting of 170 images containing 200 cars. To reduce computational costs, the 100×40 window was moved in steps of size 5 percent of the window size in each dimension, i.e., steps of five pixels and two pixels, respectively, in the horizontal and vertical directions in our experiments. Training over 1,000 images took around 10 minutes in our implementation on a machine with two Sun UltraSPARC-II 296 MHz processors and 1,024 MB memory. The time to test a 200×150 image was approximately 2.5 seconds.[6] In all, 147,802 test windows were evaluated by the system, of which more than 134,333 were negative.[7]

6. The improvements in computation time over [29] are mainly due to two factors: a faster method for computing correlations, and the observation that image patches in test images need not be compared to vocabulary parts that are not seen during training.

7. The number of negative windows was calculated by counting the windows outside the permissible ellipses around objects (see Section 3.2). However, since only one window within the permissible ellipse for any object would be counted as positive in evaluation, the effective number of negatives is actually larger than this number.

TABLE 2
Performance of Our Single-Scale Detection System with the Neighborhood Suppression Algorithm (see Section 2.4) on Test Set I, Containing 200 Cars

Activation threshold	No. of correct detections, TP	No. of false detections, FP	Recall, R $TP/200$	Precision, P $TP/(TP+FP)$	F-measure $2 \cdot R \cdot P/(R+P)$	False positive rate, $FP/134333$
0.40	169	140	**84.5 %**	54.69 %	66.40 %	0.104 %
0.55	168	107	84.0 %	61.09 %	70.74 %	0.080 %
0.65	166	89	83.0 %	65.10 %	72.97 %	0.066 %
0.75	161	67	80.5 %	70.61 %	75.23 %	0.050 %
0.85	153	44	76.5 %	77.66 %	**77.08 %**	0.033 %
0.90	141	32	70.5 %	81.50 %	75.60 %	0.024 %
0.95	120	26	60.0 %	82.19 %	69.36 %	0.019 %
0.99	79	6	39.5 %	92.94 %	55.44 %	0.004 %
0.999	35	1	17.5 %	97.22 %	29.66 %	0.001 %
0.99995	8	0	4.0 %	**100.0 %**	7.69 %	0.0 %

Points of highest recall, highest precision, and highest F-measure are shown in bold.

TABLE 3
Performance of Our Single-Scale Detection System with the Repeated Part Elimination Algorithm (see Section 2.4) on Test Set I, Containing 200 Cars

Activation threshold	No. of correct detections, TP	No. of false detections, FP	Recall, R $TP/200$	Precision, P $TP/(TP+FP)$	F-measure $2 \cdot R \cdot P/(R+P)$	False positive rate, $FP/134333$
0.20	183	557	**91.5 %**	24.73 %	38.94 %	0.415 %
0.40	177	302	88.5 %	36.95 %	52.14 %	0.225 %
0.55	166	165	83.0 %	50.15 %	62.52 %	0.123 %
0.65	162	117	81.0 %	58.06 %	67.64 %	0.087 %
0.75	156	70	78.0 %	69.03 %	73.24 %	0.052 %
0.85	145	33	72.5 %	81.46 %	**76.72 %**	0.025 %
0.90	132	23	66.0 %	85.16 %	74.37 %	0.017 %
0.95	114	15	57.0 %	88.37 %	69.30 %	0.011 %
0.99	78	5	39.0 %	93.98 %	55.12 %	0.004 %
0.999	35	1	17.5 %	97.22 %	29.66 %	0.001 %
0.99995	8	0	4.0 %	**100.0 %**	7.69 %	0.0 %

Points of highest recall, highest precision, and highest F-measure are shown in bold.

Following the discussion in Section 3.3, we present our results as recall versus $(1 - \text{precision})$ in Fig. 6. The different points on the curves are obtained by varying the activation threshold parameter as described in Section 2.4. For comparison, we also calculate the ROC curves as has been done before (using the number of negative windows evaluated by the system as the total number of negatives); these are also shown in Fig. 6.

Tables 2 and 3 show some sample points from the recall-precision curves of Fig. 6.[8] Again, for comparison, we also show the false positive rate at each point corresponding to the ROC curves. The repeated part elimination algorithm allows for a higher recall than the neighborhood suppression algorithm. However, the highest F-measure achieved by the two algorithms is essentially the same.

Figs. 7 and 8 show the output of our detector on some sample test images.

3.4.2 Comparison with Baseline Methods

As baselines for comparison, we implemented two additional detection systems. The first is a SNoW-based detector that simply uses single pixel intensities (discretized into 16 intensity ranges) as features. Since this uses the same learning algorithm as our system and differs only in the representation, it provides a good basis for judging the importance of representation in learning. The second baseline system is a nearest-neighbor based detector that uses the normalized correlation between test windows and training images (in raw pixel intensity representation) as the similarity measure. The classifier activation map for the SNoW-based method was computed as before, using SNoW activations. In the case of nearest-neighbor, the classifier activation for a positively classified test window was taken to be the correlation of the window with the nearest training image. The results (using the neighborhood suppression algorithm) are shown in Fig. 9. The poor performance of the baseline detectors is an indicator of the difficulty level of our test set: For the COIL object database, nearest-neighbor gives above 95 percent recognition accuracy, while on the face detection database in [13], the pixel-based SNoW method achieves above 94 percent recall.

3.4.3 Contributions of Different Factors

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 (using the neighborhood suppression algorithm) are shown in Fig. 10. In the first experiment, we eliminated the relation features, representing the images simply by the parts present in them. This showed a decrease in performance, suggesting that some additional information is captured by the relations. In the second experiment,

8. The reason for the lower numbers in Table 2 compared to [29] is the use of a more rigorous evaluation criterion; see Section 3.2.

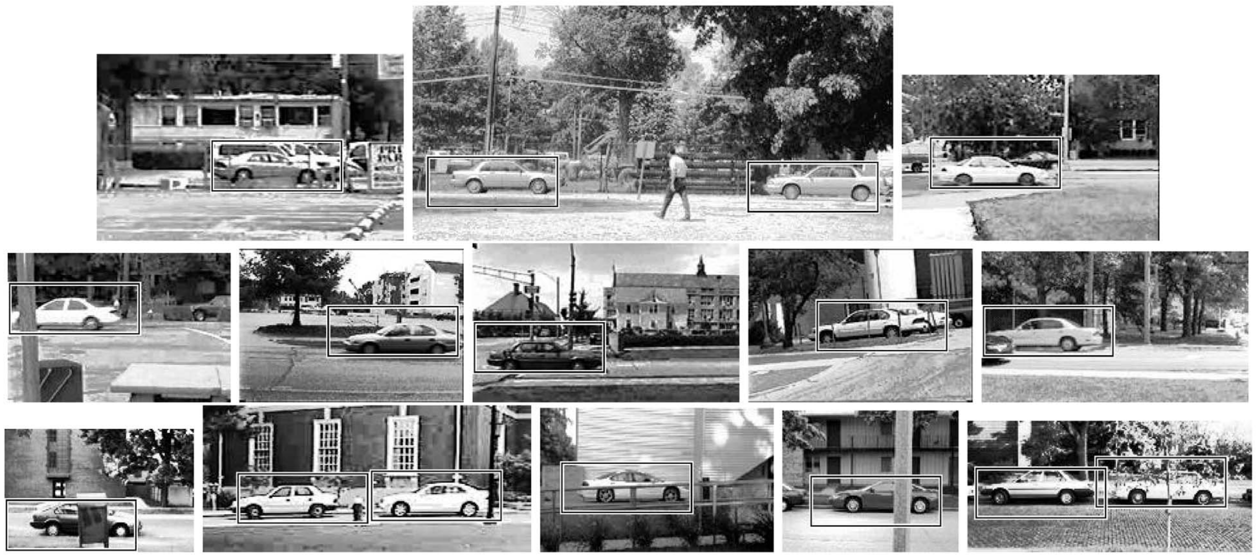


Fig. 7. Examples of test images on which our single-scale detection system achieved perfect detection results. Results shown are with the neighborhood suppression algorithm (see Section 2.4) at the point of highest F-measure from Table 2, i.e., using an activation threshold of 0.85. The windows are drawn by a separate evaluator program at the *exact* locations output by the detector.

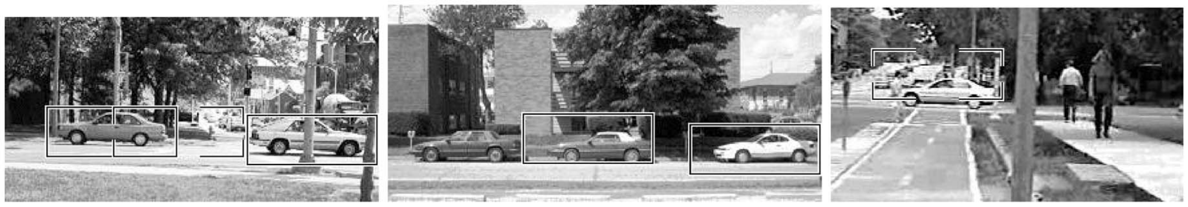


Fig. 8. Examples of test images on which our single-scale detection system missed objects or produced false detections. As in Fig. 7, results shown are with the neighborhood suppression algorithm using an activation threshold of 0.85. The evaluator program draws a window at *each* location output by the detector; locations evaluated as false positives are displayed with broken windows.

we retained the relation features, but eliminated the patch clustering step when constructing the part vocabulary, assigning a different feature id to each of the 400 patches extracted from the sample object images. This resulted in a significant decrease in performance, confirming our intuition that representing similar patches as a single conceptual-level part is important for the learning algorithm to

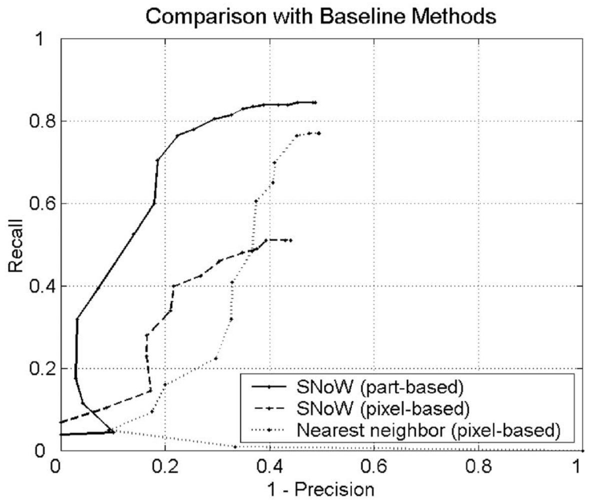


Fig. 9. Comparison of our detection system with baseline methods. The poor performance of the baseline methods is an indicator of the difficulty level of our test set. In addition, the poor performance of the pixel-based detector that uses the same learning algorithm as ours, and differs only in the representation, demonstrates the importance of choosing a good representation.

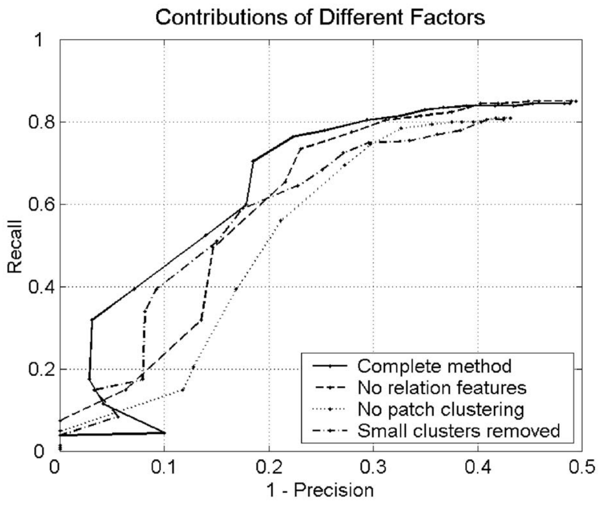
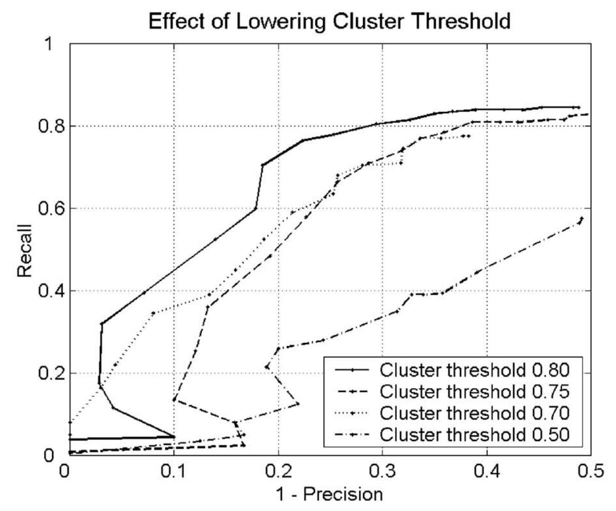
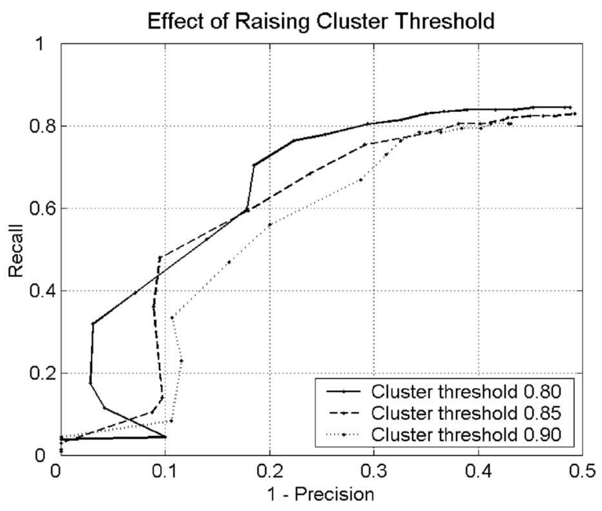


Fig. 10. Contributions of different factors in our approach to the overall performance. Both the relation features and the patch clustering step are important elements in our representation. Small clusters also have a role to play.



(a)



(b)

Fig. 11. Effect of degree of clustering. (a) Lowering the clustering threshold causes dissimilar patches to be clustered together, leading to poor generalization. (b) Raising the threshold causes patches representing the same object part to be assigned to different clusters; this again leads to poor generalization.

generalize well. We also tested the intuition that a small number of conceptual parts corresponding to frequently seen patches should be sufficient for successful detection by ignoring all one-patch clusters in the part vocabulary. However, this decreased the performance, suggesting that the small clusters also play an important role.

To further investigate the role of the patch clustering process, we studied the effect of the degree of clustering by varying the similarity threshold that two clusters are required to exceed in order to be merged into a single cluster (as mentioned in Section 2.1, the threshold we have used is 0.80). The results of these experiments (again with the neighborhood suppression algorithm) are shown in Fig. 11. The results indicate that the threshold we have used (selected on the basis of visual appearance of the clusters formed) is in fact optimal for the learning algorithm. Lowering the threshold leads to dissimilar patches being assigned to the same cluster and receiving the same feature id; on the other hand, raising the threshold causes patches that actually represent the same object part, but do not cross the high cluster threshold, to be assigned to different clusters and be treated as different parts (features). Both lead to poorer generalization.

Another experiment we conducted was to include parts derived from negative (nonobject) images in the part vocabulary; the intuition was that this should facilitate more accurate representation of negative images and should therefore improve detection results. To test this intuition, we removed 50 negative images from the training set and added them to the set of 50 object images that were originally used in vocabulary construction. Using all 100 of these images to construct the part vocabulary, we then trained the classifier on 450 positive and 450 negative images. The results, shown in Fig. 12,[9] suggest that

9. Note that, in order to make a fair comparison, the top (solid) curve in Fig. 12 was obtained using only the same 450 positive and 450 negative images for training as the lower (dashed) curve. For this reason, it is different from the curve shown in previous figures (which uses 500 positive and 500 negative training images).

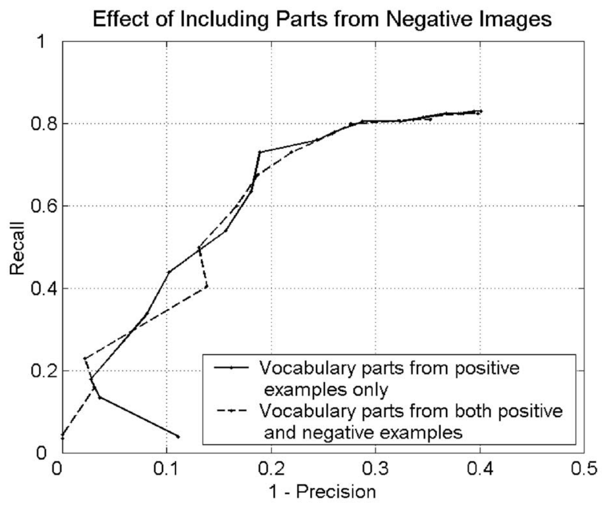


Fig. 12. Contrary to intuition, including negative images in constructing the part vocabulary does not seem to improve performance.

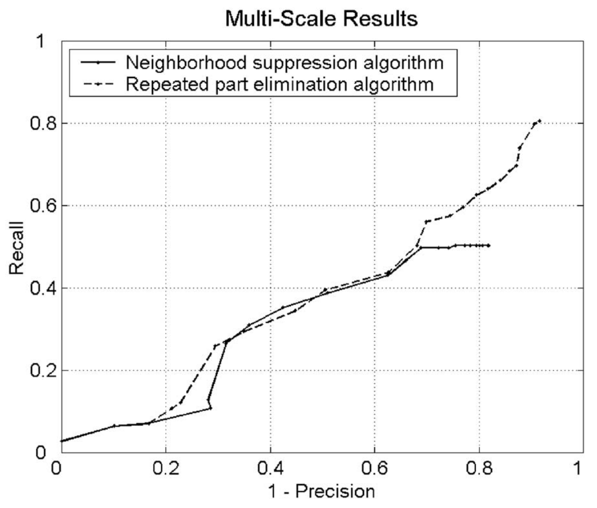


Fig. 13. Performance of our multiscale car detection system with the two algorithms described in Section 2.4.

TABLE 4
Performance of Our Multiscale Detection System with the Neighborhood Suppression Algorithm (see Section 2.4) on Test Set II, Containing 139 Cars

Activation threshold	**No. of correct detections, TP**	**No. of false detections, FP**	**Recall, R** $TP/139$	**Precision, P** $TP/(TP+FP)$	**F-measure** $2 \cdot R \cdot P/(R+P)$	**False positive rate, $FP/971763$**
0.65	70	215	**50.36 %**	24.56 %	33.02 %	0.0221 %
0.75	69	180	49.64 %	27.71 %	35.57 %	0.0185 %
0.85	65	126	46.76 %	34.03 %	39.39 %	0.0130 %
0.90	60	100	43.17 %	37.50 %	40.13 %	0.0103 %
0.95	54	56	38.85 %	49.09 %	**43.37 %**	0.0058 %
0.99	43	24	30.94 %	64.18 %	41.75 %	0.0025 %
0.999	18	7	12.95 %	72.0 %	21.95 %	0.0007 %
0.9999	10	2	7.19 %	83.33 %	13.25 %	0.0002 %
0.99999	4	0	2.88 %	**100.0 %**	5.59 %	0.0 %

Points of highest recall, highest precision, and highest F-measure are shown in bold.

TABLE 5
Performance of Our Multiscale Detection System with the Repeated Part Elimination Algorithm (see Section 2.4) on Test Set II, Containing 139 Cars

Activation threshold	**No. of correct detections, TP**	**No. of false detections, FP**	**Recall, R** $TP/139$	**Precision, P** $TP/(TP+FP)$	**F-measure** $2 \cdot R \cdot P/(R+P)$	**False positive rate, $FP/971763$**
0.20	112	1216	**80.58 %**	8.43 %	15.27 %	0.1251 %
0.40	103	739	74.10 %	12.23 %	21.00 %	0.0760 %
0.55	92	485	66.19 %	15.94 %	25.70 %	0.0499 %
0.65	87	338	62.59 %	20.47 %	30.85 %	0.0348 %
0.75	80	233	57.55 %	25.56 %	35.40 %	0.0240 %
0.85	70	149	50.36 %	31.96 %	39.11 %	0.0153 %
0.90	61	102	43.88 %	37.42 %	40.40 %	0.0105 %
0.95	55	56	39.57 %	49.55 %	**44.0 %**	0.0058 %
0.99	41	22	29.50 %	65.08 %	40.59 %	0.0023 %
0.999	17	5	12.23 %	77.27 %	21.12 %	0.0005 %
0.9999	10	2	7.19 %	83.33 %	13.25 %	0.0002 %
0.99999	4	0	2.88 %	**100.0 %**	5.59 %	0.0 %

Points of highest recall, highest precision, and highest F-measure are shown in bold.

constructing the part vocabulary from only positive examples of an object class gives an adequate representation for learning to detect instances of the object class.

3.4.4 Multiscale Results

Finally, we applied our multiscale detector, with both the neighborhood suppression algorithm and the repeated part elimination algorithm, to test set II (described in Section 3.1), consisting of 108 images containing 139 cars. As described in Section 2.4.2, images were scaled to sizes ranging from 0.48 to 1.2 times the original size, each scale differing from the next by a factor of 1.2; a 100×40 window was then moved over each scaled image. As in the single-scale case, the window was moved in steps of five pixels in the horizontal direction and two pixels in the vertical direction. The time to test a 200×150 image was approximately 12 seconds. In all, 989,106 test windows were evaluated by the system, of which over 971,763 were negative.[10]

Our multiscale results are shown in Fig. 13 and Tables 4 and 5. There are two observations to be made about these results. First, as in the single-scale case, the repeated part elimination algorithm allows for a higher recall than the neighborhood suppression algorithm (in this case, much higher, albeit at a considerable loss in precision). In terms of the highest F-measure achieved, however, the performance of the two algorithms is again similar.

The second observation about the multiscale results is that they are considerably poorer than the single-scale results; the highest F-measure drops from roughly 77 percent in the single-scale case to 44 percent in the multiscale case. This certainly leaves much room for improvement in approaches for multiscale detection. It is important to keep in mind, however, that our results are obtained using a rigorous evaluation criterion (see Section 3.2); indeed, this can be seen in Figs. 14 and 15, which show the ouput of our multiscale detector on some sample test images, together with the corresponding evaluations. In particular, our use of such a rigorous criterion makes previous results that have been reported for other approaches incomparable to ours.

4 Analysis

In this section, we analyze the performance of individual steps in our approach. In particular, we consider the results of applying the Förstner interest operator, of matching the image patches around interest points with vocabulary parts, and of applying the learned classifier.

4.1 Performance of Interest Operator

The first step in applying our detection approach is to find interest points in an image. Table 6 shows the number of

10. The number of negative windows was calculated as in the single-scale case; in this case, using the permissible ellipsoids around objects.

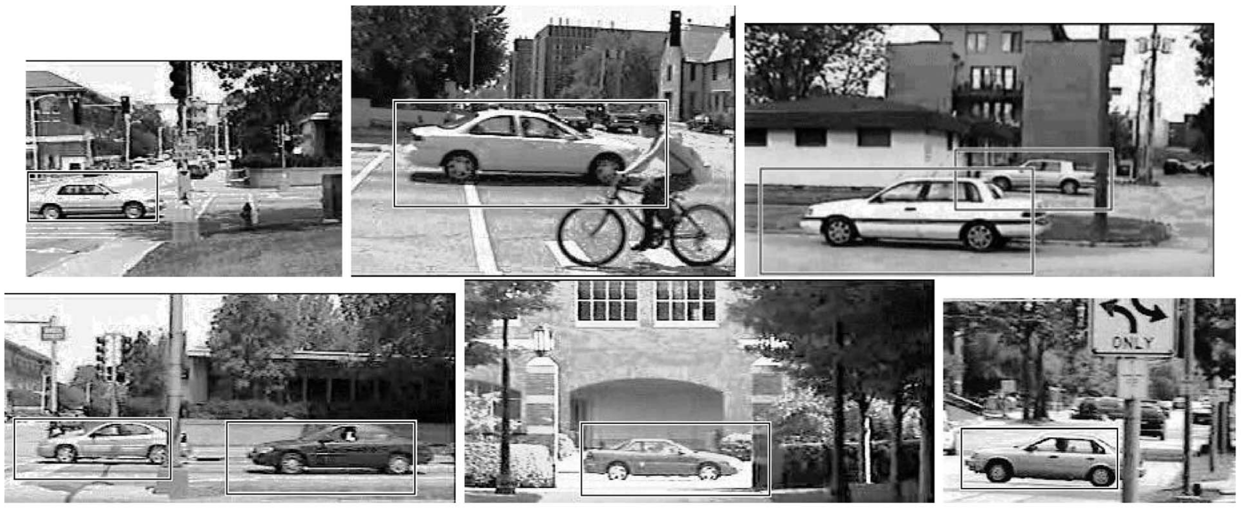


Fig. 14. Examples of test images on which our multiscale detection system achieved perfect detection results. Results shown are with the neighborhood suppression algorithm (see Section 2.4) at the point of highest F-measure from Table 4, i.e., using an activation threshold of 0.95. The windows are drawn by a separate evaluator program at the *exact* locations and scales output by the detector.

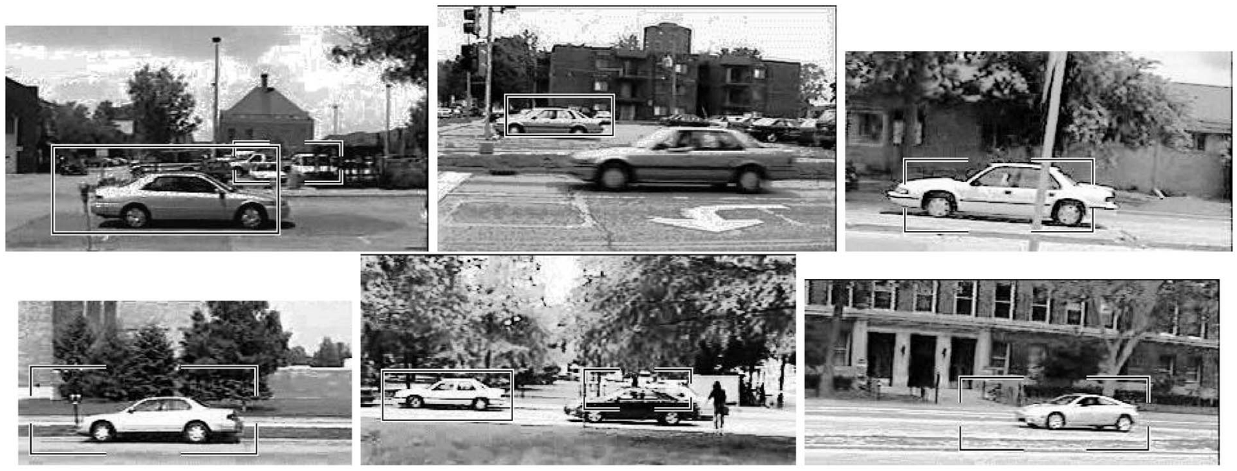


Fig. 15. Examples of test images on which our multiscale detection system missed objects or produced false detections. As in Fig. 14, results shown are with the neighborhood suppression algorithm using an activation threshold of 0.95. The evaluator program draws a window at *each* location-scale pair output by the detector; location-scale pairs evaluated as false positives are displayed with broken windows. Notice the rigorousness of the evaluation procedure (for details see Section 3.2).

TABLE 6
Numbers of Interest Points and Vocabulary Parts Found in Positive and Negative Image Windows

Image windows	No. of windows	Total no. of interest points in all windows	Average no. of interest points per window	Total no. of vocabulary parts in all windows	Average no. of vocabulary parts per window	% interest points matched by vocabulary parts
Positive train	500	4138	8.28	1890	3.78	45.67 %
Positive test I	13469	94743	7.03	50608	3.76	53.42 %
Positive test II	17343	108453	6.25	51786	2.99	47.75 %
Negative train	500	3827	7.65	1056	2.11	27.59 %
Negative test I	134333	535736	3.99	177677	1.32	33.17 %
Negative test II	971763	2710149	2.79	929218	0.96	34.29 %

interest points found by the Förstner interest operator in positive and negative image windows.[11] In Fig. 16, we show histograms over the number of interest points found. Positive windows mostly have a large number of interest points; over 90 percent positive windows in training, and over 75 percent in test set I and 70 percent in test set II, have five or more interest points. Negative windows have a more uniform distribution over the number of interest points in the training images, but mostly have a small number of interest points in test images; over 60 percent in test set I and over 75 percent in test set II have less than five interest points.

11. As before, in test images, all windows falling within the permissible ellipses/ellipsoids around objects were counted as positive; all others were counted as negative.

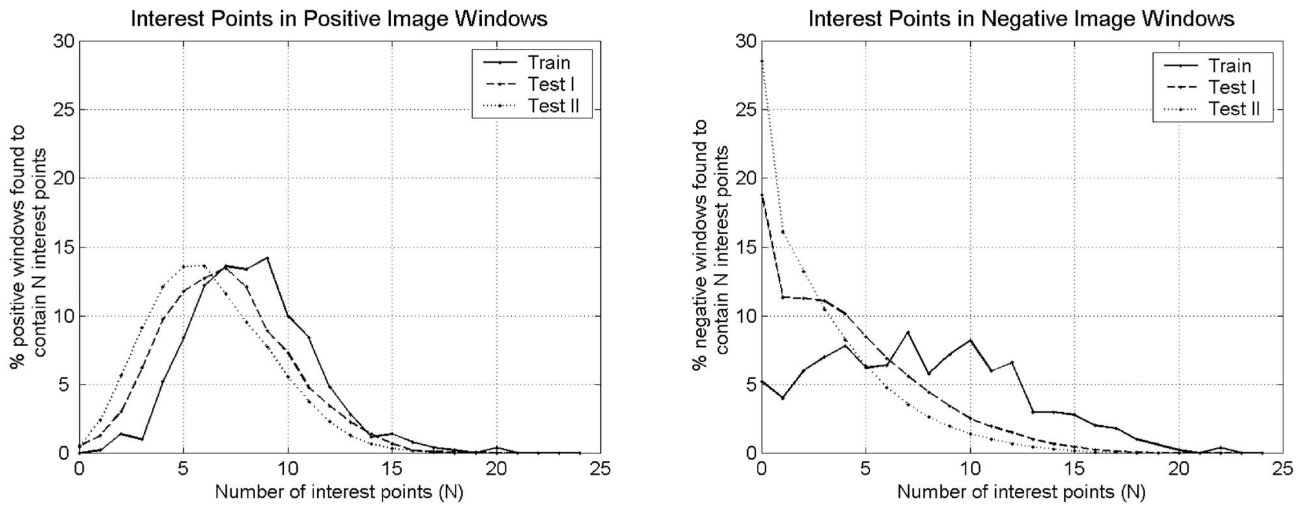


Fig. 16. Histograms showing distributions over the number of interest points in positive and negative image windows.

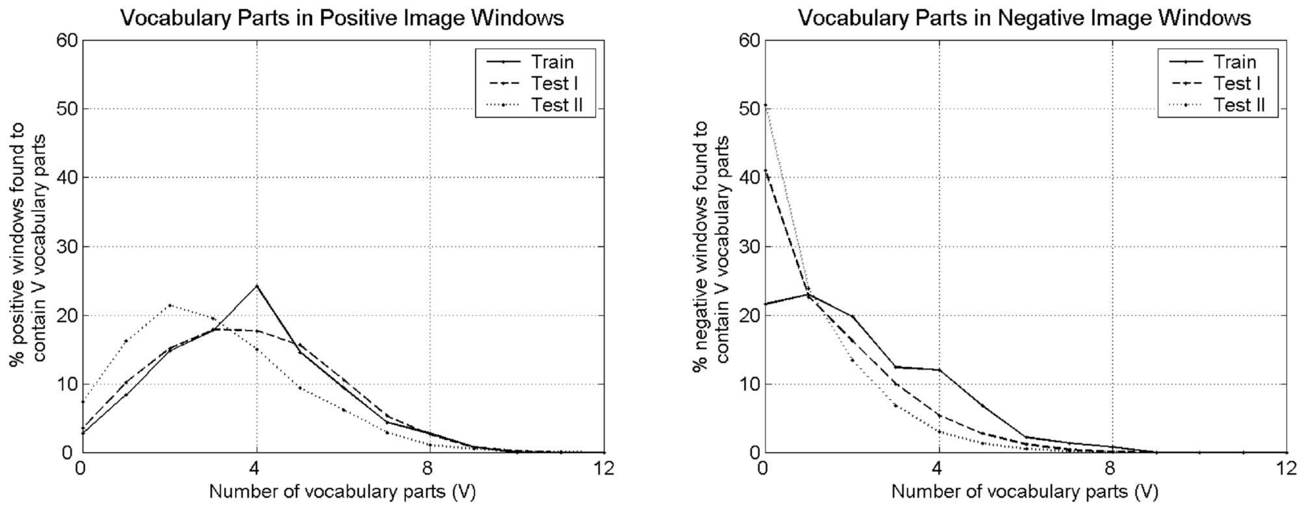


Fig. 17. Histograms showing distributions over the number of vocabulary parts in positive and negative image windows.

4.2 Performance of Part Matching Process

Once interest points have been found and image patches around them highlighted, the next step is to find vocabulary parts that match the highlighted patches. The result of this step is important since it determines the actual representation of an image window that is given to the classifier. Table 6 shows the number of vocabulary parts found in positive and negative image windows. The proportion of highlighted patches matched by vocabulary parts is more or less the same across training images and both test sets; roughly 50 percent for positive windows and 30 percent for negative windows. Histograms over the number of vocabulary parts are shown in Fig. 17; the distributions are similar in form to those over interest points. It is interesting to note that the distribution over the number of vocabulary parts in positive windows in test set I is very close to that over the number of vocabulary parts in positive training images.

4.3 Performance of Learned Classifier

The performance of the classifier is shown in Table 7. While its performance on test set I is similar to that on the training set, its performance on test set II is drastically different. In particular, the classification accuracy on positive windows in test set II is very low. Furthermore, as is seen from Fig. 18, positive windows in test set II that are classified correctly are done so with lower confidence than in test set I and the training set; the activations for true and false positives are well-separated in test set I and the training set, but the opposite is true for test set II. These observations shed some light on our multiscale results, although it remains to be

TABLE 7
Performance of Raw Classifier on Positive and Negative Image Windows

Image windows	No. of windows	No. of correct classifications	Classification accuracy
Positive train	500	470	94.0 %
Positive test I	13469	12054	89.49 %
Positive test II	17343	4935	28.46 %
Negative train	500	311	62.2 %
Negative test I	134333	75885	56.49 %
Negative test II	971763	797432	82.06 %

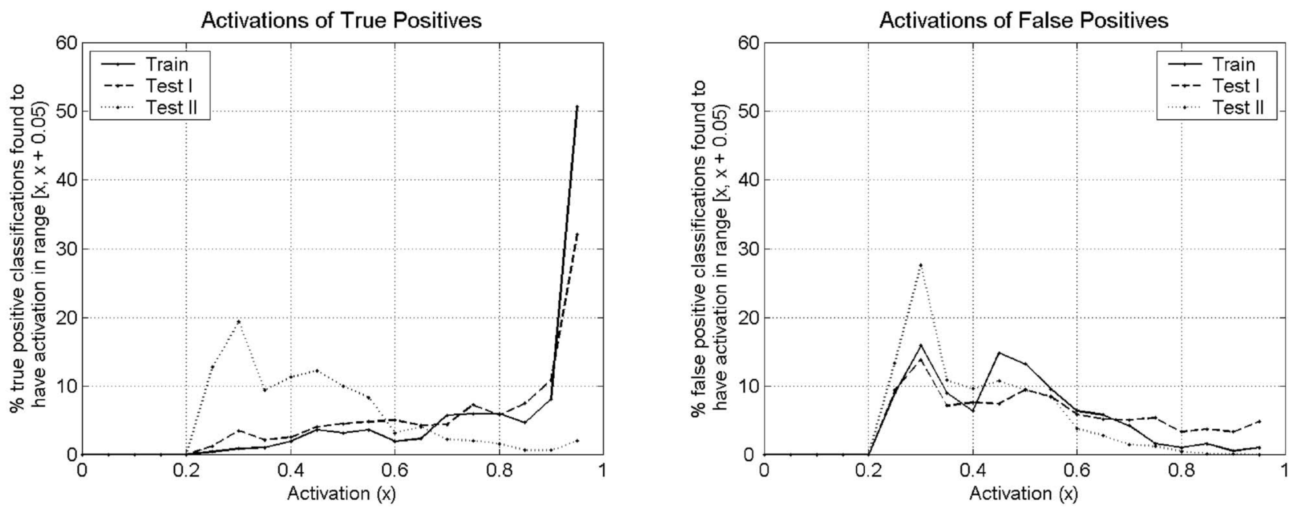


Fig. 18. Histograms showing distributions over the activation value produced by the learned classifier in the case of true positive classifications (positive windows correctly classified as positive) and false positive classifications (negative windows incorrectly classified as positive).

understood why the performance of the classifier differs in the two cases.

5 Conclusion

To summarize, we have presented an approach for learning to detect objects in images using a sparse, part-based representation. In our approach, a vocabulary of distinctive 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, together with spatial relations observed among the parts. Based on this representation, a learning algorithm is used to automatically learn a classifier that distinguishes between members and nonmembers of the object class. To detect instances of the object class in a new image, the learned classifier is applied to several windows in the image to generate what we term a classifier activation map in the single-scale case and a classifier activation pyramid in the multiscale case. The activation map or pyramid is then processed to produce a coherent detection hypothesis; we presented two algorithms for this process.

We also 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 emphasized the importance of specifying and standardizing evaluation criteria in object detection experiments; this is essential for comparisons between different approaches to be meaningful. As a step in this direction, we formulated rigorous, quantitative evaluation criteria for both single-scale and multiscale cases. Finally, we argued that recall-precision curves are more appropriate than ROC curves for measuring the performance of object detection approaches.

We presented a critical evaluation of our approach, for both the single-scale case and the multiscale case, under the proposed evaluation standards. We evaluated it here on images containing side views of cars; the approach is easily extensible to other objects that have distinguishable parts in a relatively fixed spatial configuration.

There are several avenues for further research. The multiscale detection problem is clearly harder than the single-scale one; much room seems to remain for improvement in this direction. One possibility is to incorporate scale information in the features; this may help improve the performance of the classifier. The general detection problem may also require detecting objects at different orientations; it may be possible to achieve this by learning a number of view-based classifiers as in [18], and extending the classifier activation map or pyramid to incorporate activation information from the different views. Computation time can be reduced by processing different scales and orientations in parallel.

At the learning level, a natural extension is to learn to detect several object classes at once. It also remains an open problem to formulate a learning problem that directly addresses the problem of detection rather than classification.

Acknowledgments

The authors would like to thank Ashutosh Garg, David Kriegman, Cordelia Schmid, and Dav Zimak for helpful discussions and comments. They are also grateful to the anonymous referees for several useful suggestions. This work was supported by US National Science Foundation ITR grants IIS 00-85980 and IIS 00-85836. A preliminary version of this work was presented at the Seventh European Conference on Computer Vision in May 2002.

References

[1] S. Ullman, *High-Level Vision: Object Recognition and Visual Cognition.* MIT Press, 1996.
[2] I. Biederman, "Recognition by Components: A Theory of Human Image Understanding," *Psychological Rev.,* vol. 94, pp. 115-147, 1987.
[3] N.K. Logothetis and D.L. Sheinberg, "Visual Object Recognition," *Ann. Rev. of Neuroscience,* vol. 19, pp. 577-621, 1996.
[4] S.E. Palmer, "Hierarchical Structure in Perceptual Representation," *Cognitive Psychology,* vol. 9, pp. 441-474, 1977.

[5] E. Wachsmuth, M.W. Oram, and D.I. Perrett, "Recognition of Objects and Their Component Parts: Responses of Single Units in the Temporal Cortex of the Macaque," *Cerebral Cortex,* vol. 4, pp. 509-522, 1994.

[6] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, pp. 349-361, 2001.

[7] A.J. Colmenarez and T.S. Huang, "Face Detection with Information-Based Maximum Discrimination," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 782-787, 1997.

[8] H.A. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 1, pp. 23-38, Jan. 1998.

[9] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 130-136, 1997.

[10] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.

[11] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Detection," *Proc. Fifth Int'l Conf. Computer Vision,* 1995.

[12] Y. Amit and D. Geman, "A Computational Model for Visual Selection," *Neural Computation,* vol. 11, no. 7, pp. 1691-1715, 1999.

[13] M-H. Yang, D. Roth, and N. Ahuja, "A SNoW-Based Face Detector," *Advances in Neural Information Processing Systems 12,* S.A. Solla, T.K. Leen, and K.-R. Müller, eds., pp. 855-861, 2000.

[14] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2001.

[15] L. Shams and J. Spoeslstra, "Learning Gabor-Based Features for Face Detection," *Proc. World Congress in Neural Networks, Int'l Neural Network Soc.,* pp. 15-20, 1996.

[16] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *Int'l J. Computer Vision,* vol. 38, no. 1, pp. 15-33, 2000.

[17] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," *Feature Grouping,* D. Forsyth, ed., 1999.

[18] H. Schneiderman and T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 746-751, 2000.

[19] S. Ullman, E. Sali, and M. Vidal-Naquet, "A Fragment-Based Approach to Object Representation and Classification," *Proc. Fourth Int'l Workshop Visual Form,* C. Arcelli, L.P. Cordella, and G. Sanniti di Baja, eds., pp. 85-100, 2001.

[20] S. Ullman, M. Vidal-Naquet, and E. Sali, "Visual Features of Intermediate Complexity and Their Use in Classification," *Nature Neuroscience,* vol. 5, no. 7, pp. 682-687, 2002.

[21] M. Weber, M. Welling, and P. Perona, "Unsupervised Learning of Models for Recognition," *Proc. Sixth European Conf. Computer Vision,* pp. 18-32, 2000.

[22] D. Roth, M-H. Yang, and N. Ahuja, "Learning to Recognize Three-Dimensional Objects," *Neural Computation,* vol. 14, no. 5, pp. 1071-1103, 2002.

[23] H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance," *Proc. Fifth Int'l Joint Conf. Artificial Intelligence,* 1977.

[24] C. Schmid and R. Mohr, "Local Greyvalue Invariants for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 530-535, May 1997.

[25] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision II.* Addison-Wesley, 1993.

[26] A.J. Carlson, C. Cumby, J. Rosen, and D. Roth, "The SNoW Learning Architecture," Technical Report UIUCDCS-R-99-2101, Computer Science Dept., Univ. Illinois at Urbana-Champaign, May 1999.

[27] D. Roth, "Learning to Resolve Natural Language Ambiguities: A Unified Approach," *Proc. 15th Nat'l Conf. Artificial Intelligence,* pp. 806-813, 1998.

[28] N. Littlestone, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," *Machine Learning,* vol. 2, no. 4, pp. 285-318, 1988.

[29] S. Agarwal and D. Roth, "Learning a Sparse Representation for Object Detection," *Proc. Seventh European Conf. Computer Vision,* vol. 4, pp. 113-130, 2002.

Shivani Agarwal received the BSc degree with honors in mathematics from the University of Delhi, India, in 1998, and the BA degree with honors in computer science from the University of Cambridge, UK, in 2000. In 2002, she was awarded the MS degree in computer science by the University of Illinois at Urbana-Champaign, where she is currently a doctoral student. Her main research interests are in machine learning, statistical learning theory, and pattern recognition.

Aatif Awan received the BS degree in computer system engineering from the GIK Institute of Engineering, Pakistan, in 2002. Currently, he is a graduate student and Sohaib and Sara Abbasi Fellow at the University of Illinois at Urbana-Champaign. His research interests include machine learning and data mining.

Dan Roth received the BA degree *summa cum laude* in mathematics from the Technion, Israel, in 1982, and the PhD degree in computer science from Harvard University in 1995. He is an associate professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign and the Beckman Institute. He is a fellow of the Institute of Advanced Studies at the University of Illinois and a Willett Faculty Scholar of the College of Engineering. His research spans both theoretical work in machine learning and intelligent reasoning and work on applying learning and inference to intelligent human-computer interaction—focusing on learning and inference for natural language understanding related tasks and for visual recognition. He has received a number of research awards, including the best paper award at the International Joint Conference on Artificial Intelligence (IJCAI) in 1999 for the paper "Learning in Natural Language," is on the editorial board of a number of journals in his research areas, and was the program cochair of ACL'03, the main international meeting of the Association for Computational Linguistics and the natural language processing community. He is a member of the IEEE Computer Society.