International Journal of Computer Vision 56(3), 151–177, 2004

Object Detection Using the Statistics of Parts

HENRY SCHNEIDERMAN AND TAKEO KANADE*
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
hws@cs.cmu.edu
tk@cs.cmu.edu

Received April 20, 2001; Revised February 15, 2002; Accepted March 6, 2002

Abstract. In this paper we describe a trainable object detector and its instantiations for detecting faces and cars at any size, location, and pose. To cope with variation in object orientation, the detector uses multiple classifiers, each spanning a different range of orientation. Each of these classifiers determines whether the object is present at a specified size within a fixed-size image window. To find the object at any location and size, these classifiers scan the image exhaustively.

Each classifier is based on the statistics of localized *parts*. Each *part* is a transform from a subset of wavelet coefficients to a discrete set of values. Such *parts* are designed to capture various combinations of locality in space, frequency, and orientation. In building each classifier, we gathered the class-conditional statistics of these *part* values from representative samples of object and non-object images. We trained each classifier to minimize classification error on the training set by using Adaboost with Confidence-Weighted Predictions (Shapire and Singer, 1999). In detection, each classifier computes the *part* values within the image window and looks up their associated class-conditional probabilities. The classifier then makes a decision by applying a likelihood ratio test. For efficiency, the classifier evaluates this likelihood ratio in stages. At each stage, the classifier compares the partial likelihood ratio to a threshold and makes a decision about whether to cease evaluation—labeling the input as non-object—or to continue further evaluation. The detector orders these stages of evaluation from a low-resolution to a high-resolution search of the image. Our trainable object detector achieves reliable and efficient detection of human faces and passenger cars with out-of-plane rotation.

Keywords: object recognition, object detection, face detection, car detection, pattern recognition, machine learning, statistics, computer vision, wavelets, classification

1. Introduction

Object detection is a big part of people's lives. We, as human beings, constantly "detect" various objects such as people, buildings, and automobiles. Yet it remains a mystery how we detect objects accurately and with little apparent effort. Comprehensive explanations have defied psychologists and physiologists for more than a century.

*This work was supported in part by the Advanced Research and Development Activity (ARDA) under contract number MDA904-00-C-2109.

Our goal in this research is not to understand how humans perceive, but to create computer methods for automatic object detection. Automated object detection has many potential uses including image retrieval. Digital image collections have grown dramatically in recent years. Corbis estimates it has more than 67 million images in its collection. The Associated Press collects and archives an estimated 1,000 photographs a day. Currently, the usability of these collections is limited by a lack of effective retrieval methods. To find a specific image in such a collection, people must search using text-based captions and primitive image features such as color and texture. Automatic object detection

could be used to extract more information from these images and help label and categorize them. Improved search methods will make these databases accessible to wider groups of users, such as law enforcement agencies, medical practitioners, graphic and multimedia designers, and artists. Automatic object detection could also be useful in photography. As camera technology changes from film to digital capture, cameras will become part optics and part computer. Such a camera could automatically focus, color balance, and zoom on a specified object of interest, say, a human face. Also, detectors of a specific object have specialized uses: face detectors for face identification and car detectors for monitoring traffic.

1.1. Challenges in Object Detection

Automatic object detection is a difficult undertaking. In over 30 years of research in computer vision, progress has been limited. The main challenge is the amount of variation in visual appearance. An object detector must cope with both the variation within the object category and with the diversity of visual imagery that exists in the world at large. For example, cars vary in size, shape, coloring, and in small details such as the headlights, grille, and tires. The lighting, surrounding scenery, and an object's pose affect its appearance. A car detection algorithm must also distinguish cars from all other visual patterns that may occur in the world, such as similar looking rectangular objects.

1.2. Object Detection Using Classifiers

Our method for object detection factors out variation in the pose of the object. Our object detector uses a set of classifiers, each of which determines whether the object is present at a specific pose in a fixed-size rectangular image window. For faces, the detector uses classifiers for three discrete poses: front, left profile, and right profile. Taking advantage of facial symmetry, we only needed to train classifiers for the frontal and right profile viewpoints shown in Fig. 1(a), and we built a left profile detector by reflecting the right profile detector. For cars, we use 15 discrete viewpoints, and by exploiting symmetry again, we only trained classifiers for the eight viewpoints as shown in Fig. 1(b). These classifiers

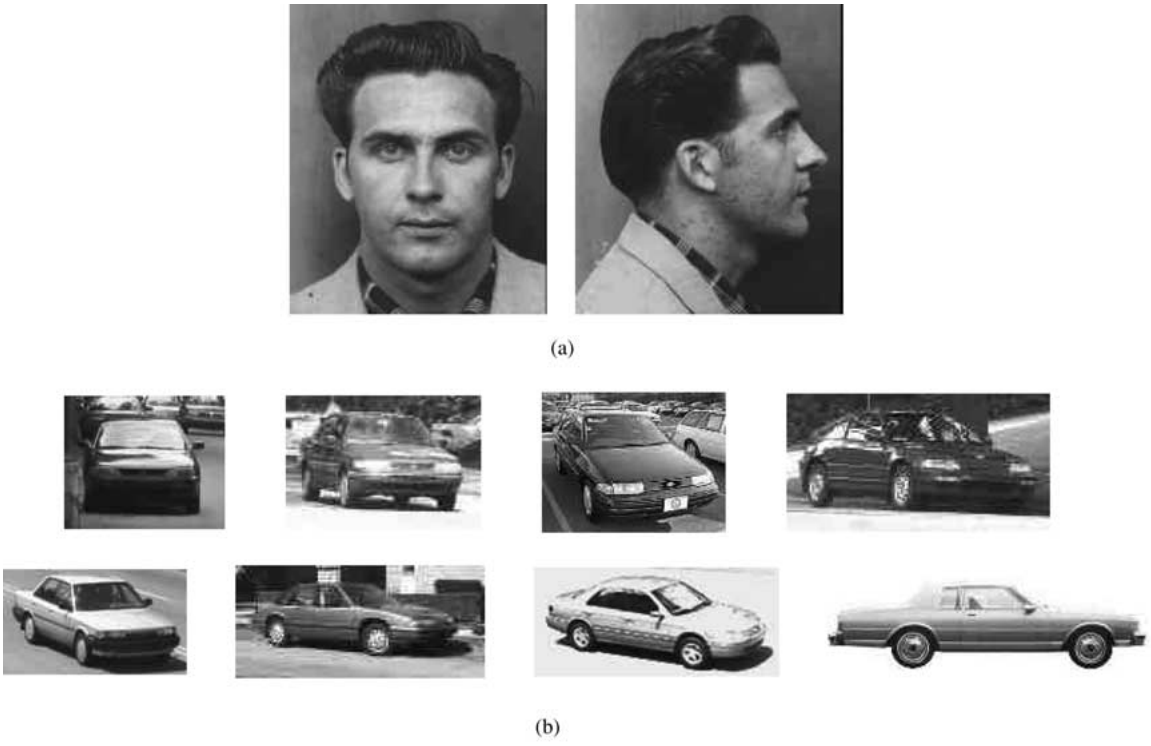


Figure 1. Multiple classifiers are built to deal with appearance changes due to pose. (a) For faces, classifiers are trained on 2 viewpoints and (b) for cars, classifiers are trained on 8 viewpoints.

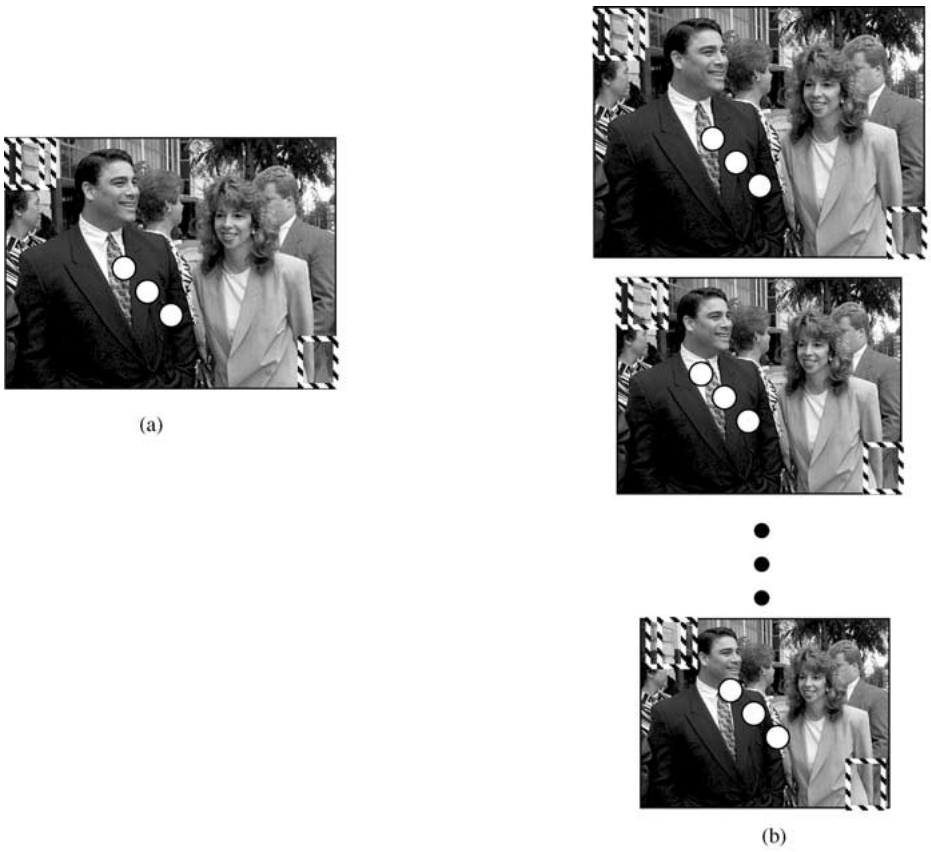


Figure 2. Detection by scanning classifier across image in both (a) position and (b) scale.

tolerate a small range of variation in object orientation, size, and alignment within the image window.

To perform detection, we scan each classifier over the original image and a series of resized versions of the original image, as illustrated in Fig. 2, where the rectangular blocks indicate successive applications of the classifier. This exhaustive scanning operation makes it possible to find the object over variation in location and size, and can be done with surprising efficiency, as we will describe later in Section 4. Often, the same entity is detected by more than one view-based detector, such as the woman in the foreground of the image in Fig. 3. To determine the final detection outcome the detector combines the results from various viewpoint classifiers by using simple arbitration heuristics, in the case of Fig. 3, selecting the frontal viewpoint.

1.3. Parts-Based Representation for Classifier

The central research issue is how to design a basic classifier that can cope with variation in appearance. One hypothetical method is to build the classifier as a table, enumerating the most probable classification for every combination of input variables (see Table 1).

Such a table would give the smallest average classification error, assuming we could label each input appropriately. Obviously, this table is not possible in practice. Even a classifier over a 20×20 input window requires $256^{400} \approx 10^{964}$ entries! Classifier design, therefore, must take advantage of the constraints of the

Table 1. Ideal but infeasible classifier.

(1, 1)	(1, 2)	...	(20, 20)	Classification
0	0	...	0	Non-object
0	0	...	1	Non-object
...	...	...	...	...
35	45	...	28	Object
...	...	...	...	...
255	255	...	255	Non-object

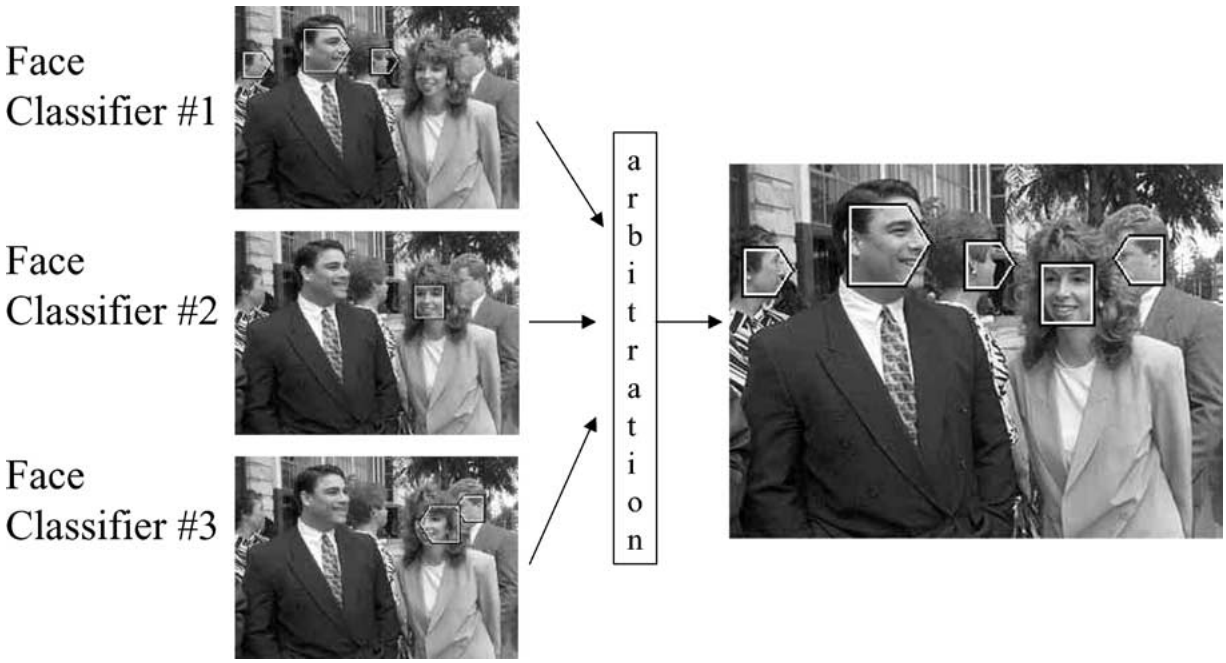


Figure 3. Combining the results from multiple view-based detectors.

visual world in order to obtain a much more compact representation.

In choosing a representation for the classifier, we can differentiate between two types of approaches: global and parts-based. Global representations, like an ideal table, try to model the joint behavior of all input variables. Computational and storage limitations, however, do not allow for a fully general function of the input variables. As a practical solution, global models must use limited functional forms of the input variables, such as linear, quadratic, or third order. Alternatively, global models may reduce the dimensionality of the input space using certain transforms or a reduction in the number of input variables. A higher-order model or even a non-parametric method can then describe the joint behavior of this reduced set of variables.

In the parts-based approach, the input variables are grouped into sets, where the relationships within each set are more accurately modeled than those across sets. We refer to each such set as a *part*. For example, parts of a face, such as the eyes, nose, and mouth, can be considered as *parts* and modeled separately. However, it should be emphasized that *parts* need not have a natural meaning to us (such as a nose or an eye), but could be defined as a group of pixels, or transform variables, that satisfy certain mathematical properties. In addition, these *parts* do not have to be composed from disjoint groups of variables; a variable can be re-used in multiple *parts*. In this paper, we denote *parts* with italics to refer to this more general meaning.

This parts-based approach is based on an implicit assumption that for a given object, each pixel is statistically related with some pixels more than others. Under this assumption, a global model does not make good use of modeling resources, as it makes no distinction whether a combination of pixels is useful or irrelevant. A quadratic filter, for example, represents correlation within each pair of pixels; the majority of these pairs may have negligible statistical dependency. A parts-based approach, in contrast, can select each *part* to represent a small group of variables that are known to be statistically dependent. Such an approach avoids devoting representational resources to weak relationships and instead allocates richer models to the stronger relationships.

The parts-based assumption becomes more pronounced if statistical dependency is measured among transformed variables that decorrelate the imagery rather than among the original pixels. By decorrelation, statistical dependency will be concentrated in small sets of variables. We chose a wavelet basis as a transform for decorrelation because it tends to work well on imagery of natural scenes (Field, 1999). In particular, we

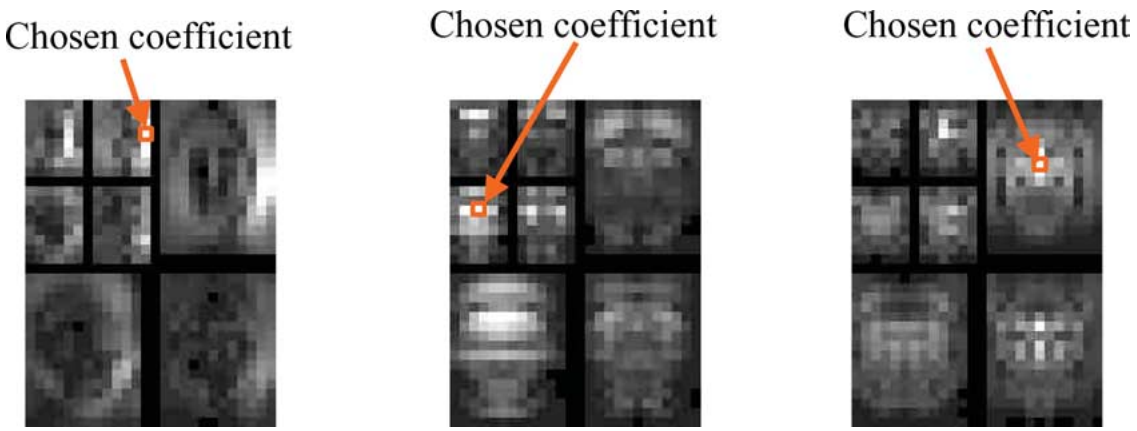


Figure 4. Pair-wise mutual information between a chosen coefficient location and all other locations in the wavelet transform for frontal faces.

chose a 5/3 linear phase wavelet filterbank (Strang and Nguyen, 1997).

Experiments support the validity of the parts-based assumption. We measured statistical dependency among 5/3 linear phase wavelet coefficients for several viewpoints of faces and cars. For each viewpoint, we collected the joint probability distribution for each pair of wavelet coefficients by using a large set of geometrically aligned images of the object. By quantizing each coefficient to 5 discrete levels, we represented each distribution as a histogram with 25 (5×5) bins. These distributions allowed us to compute the mutual information for each wavelet coefficient pair. Mutual information measures the strength of the statistical dependence between the two variables. Figure 4 illustrates some of the results of this experiment for frontal faces. Each "image" graphically represents the mutual information values between one chosen coefficient (indicated by an arrow) and all the other coefficients in the wavelet transform. The brightness at each location indicates the mutual information between the chosen coefficient and coefficient at that location. Notice that each coefficient is statistically related only with a relatively small number of the other coefficients (Naturally, since a variable has the strongest mutual information with itself, the location of the chosen coefficient is the brightest point.) This phenomenon of limited statistical dependency is typical for faces and cars.

1.4. Properties of Classifier

We used a combination of the following eight design choices to develop a parts-based classifier.

- Decomposition into *parts*

Given input variables, such as wavelet coefficients, we form a set of *parts*, each consisting of a group of variables that are statistically dependent. We then treat these *parts* as statistically independent. With this assumption, our classifier takes the following form as a likelihood ratio test; that is, it decides that the object is present if the left side is greater than λ:

$$\prod_r \frac{P_r(part_r \mid object)}{P_r(part_r \mid non\text{-}object)} > \lambda \qquad (1)$$

where $part_r$ is a discrete-valued variable obtained as a function of a chosen group of wavelet coefficients within the classification window, $P_r(part_r \mid object)$ represents the probability distribution over the discrete range of $part_r$ conditioned on the presence of the object, and similarly, $P_r(part_r \mid non\text{-}object)$ is conditioned on the absence of the object.

Strictly speaking, the assumption of statistical independence of the *parts* is not true. Yet, we still obtained accurate classification results; a possible interpretation of this will be discussed in Section 2.4.

- Probabilistic representation of *parts*

Usually, parts are thought of as being binary-valued and deterministic. For example, an eye can either be present or absent. However, in Eq. (1), we designed each $part_r$, to take a range of values. In our face and car examples, the range was approximately 10^4. Our classifier represents these values as probabilistic quantities rather than deterministic quantities; that is,

$P_r(part_r \mid non\text{-}object)$ and $P_r(part_r \mid object)$ associate probabilities to each value of $part_r$.

- *Parts* with locality in space, frequency, and orientation

Our classifier uses a variety of *parts* to embody various combinations of locality in space, frequency, and orientation. Some *parts* represent small regions over high frequencies, other *parts* represent large regions over low frequencies, and still other *parts* are specialized in horizontal and vertical information. These choices are designed to capture common statistical dependencies in appearance of an object. The wavelet representation allowed us to directly design *parts* with these locality properties.

- Maximalist collection of *parts*

Our classifier represents *parts* from all areas across the entire extent of the object. This representation could be considered maximalist in contrast to a minimalist one that relies on a few features (e.g., eyes, nose, mouth). More information, if used properly, will always improve the detection result. In particular, we have found that *parts* with even seemingly indistinct cues such as uniform areas are indeed discriminative.

- Geometric arrangement of *parts*

It should be noted that the geometrical relationships of the *parts* are implicit in Eq. (1). Each *part* is defined as a function of a specific group of wavelet coefficients. All wavelet coefficients are represented with respect to a common coordinate frame affixed to the image window to be classified. Therefore, this representation captures geometry by placing all *parts* in a common coordinate system. In the next section, this geometric representation will become more explicit by an equivalent representation of Eq. (1) in terms of "local operators." This representation allows for a flexible configuration of parts in the object unlike a single template that implies a rigid configuration.

- Tables for *part* statistics

Our classifier represents each set of statistical distributions, $P_r(part_r \mid non\text{-}object)$ and $P_r(part_r \mid object)$, using tables. This representation is possible because we have chosen each *part* to have a discrete range of values. Retrieval of probability values thus involves only a lookup into a table representing the likelihood ratio $P_r(part_r \mid object)/P_r(part_r \mid non\text{-}object)$. We can estimate these probability distributions by simply counting the occurrences of each *part* value over a large set of training images. A table avoids assumptions about the distributional structure of *part* statistics (e.g., Gaussian), while retaining good properties for estimation, including satisfaction of the Cramer-Rao lower bound, closed form solution, and no bias.

- AdaBoost to weight training examples

To build the overall classifier, we could separately estimate $P_r(part_r \mid non\text{-}object)$ and $P_r(part_r \mid object)$, and plug them into the likelihood ratio test, Eq. (1). This approach would give the best possible performance (with this functional form) if our training data is truly representative. However, the fact that we have only a finite set of training examples will limit the estimation accuracy, particularly for the non-object class. This limitation can be partially overcome by training the classifier to explicitly minimize classification error on the training set. We chose to use a unique training method involving Adaboost with Confidence Weighted Predictions (Shapire and Singer, 1999) that guarantees to minimize an upper bound on the classification error on the training set. This method also has the natural interpretation of re-weighting of the training examples. This method allowed us to count weighted occurrences of each *part* value, and thereby retain the advantage of estimating each distribution as a table.

- Coarse-to-fine search evaluation strategy

We use several strategies to make detection computationally efficient. In Fig. 2, it may appear that the classifier completely re-evaluates each window. However, overlapping windows share much information that does not need to be re-computed. The classifier computes the wavelet transform and the *part* values once, at most, for the entire image scale. We also avoid repeating computations over successive octaves in the search across scale. It is not necessary to re-compute the entire wavelet transform, nor *parts* that are computationally equivalent at multiple octaves. The classifier also does not need to evaluate the complete likelihood ratio to make a decision in most cases. A partial evaluation of the likelihood ratio is often sufficient to rule out the presence of an object. The classifier orders the evaluation of the likelihood ratio from *parts* in coarse resolution to *parts* in fine resolution. First, the detector

evaluates the image at a coarse resolution, reduced by a factor of 8 in both directions; that is, evaluation sites are spaced 8 pixels apart in the original image. Then the resolution is reduced by a factor of 4. Such progressive evaluation techniques enabled us to achieve significant computational savings and implement a relatively efficient algorithm with little penalty in accuracy.

1.5. *Overview of Classifier*

Let us overview how our classifier based on Eq. (1) works. The description here is primarily for illustrative purposes, and more details can be found in Section 2.

The classification algorithm involves three steps, as shown in Fig. 5. In the first step, N local operators, $f_k(x, y), k = 1, N$, evaluate the image window. The resulting measurements are discrete-valued. Each output at each location represents a separate *part* and the conglomerate of the outputs from all N operators, each sampled at all locations, represent the entire set of *parts*.

Rewriting Eq. (1) in terms of local operators gives:

$$\prod_r \frac{P_r(part_r \mid object)}{P_r(part_r \mid non\text{-}object)} = \prod_k \prod_{x,y} \frac{P_k(f_k(x, y), x, y \mid object)}{P_k(f_k(x, y), x, y \mid non\text{-}object)} \quad (2)$$

where each $part_r$ corresponds to a unique combination of k, x, and y.

The classifier then retrieves two probabilities associated with each operator output $f_k(x, y)$. It obtains these probabilities from two pre-computed probability distributions for each operator, $P_k(f_k(x, y), x, y \mid object)$ and $P_k(f_k(x, y), x, y \mid non\text{-}object)$. $P_k(f_k(x, y), x, y \mid object)$ represents the statistical knowledge of the object's appearance. Figure 5 illustrates a case that the probability of output value #5710 from operator "1" at position (0, 0) on the object is 0.53. The other probability distribution, $P_k(f_k(x, y), x, y \mid non\text{-}object)$, describes the visual world other than the object.

Note that each of these distributions is a joint function of operator value f_k and operator position (x, y)

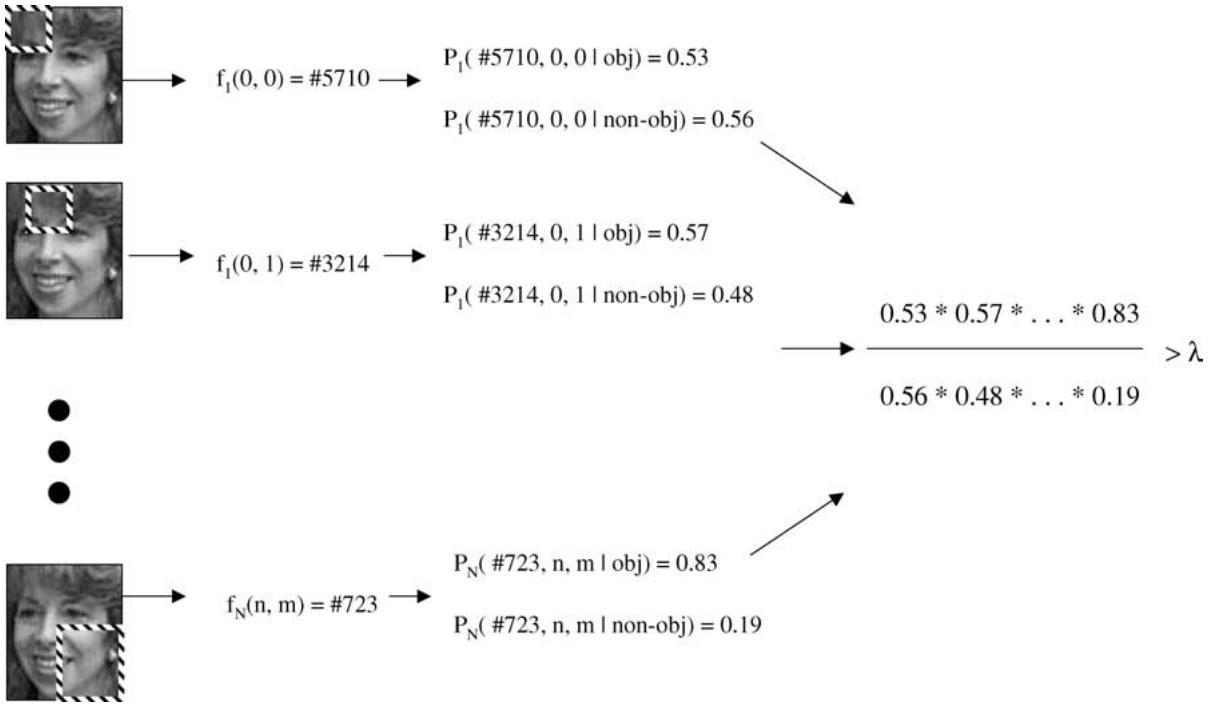


Figure 5. Classification algorithm overview. N local operators evaluate the image window at nm locations. Class-conditional probabilities are retrieved for each output and combined in a likelihood ratio test.

within the classification window. This joint representation explicitly models the geometric configuration of the *parts*. Recall, also, that each operator takes on a discrete value, allowing us to represent each probability distribution as a table.

Next, the classifier makes a decision by computing a likelihood ratio test as in Eq. (2). It multiplies all the probabilities retrieved from $P_k(f_k(x, y), x, y \mid object)$ in the numerator and divides by the product of the probabilities from $P_k(f_k(x, y), x, y \mid non\text{-}object)$. It then compares the resulting value to a threshold. If the value is greater than the threshold, it decides that the object is present in the window; otherwise, it decides it is not present.

1.6. Related Work

The idea of using statistical independence assumptions in probabilistic modeling problems has a long history in the literature of pattern recognition, beginning possibly with Lewis (1959) and Chow and Liu (1966). In computer vision, several researchers have used a parts-like decomposition with an explicit statistical independence assumption. Recent representative work includes Burl and Perona (1996), Burl et al. (1998), Geman and Flueret (2001), and Amit (2000), who use binary or deterministic representations of the parts. Schiele and Crowley (1996, 2000) use a probabilistic representation where probability is estimated over the response to Gaussian derivative filters. However, there is no notion of geometry in Schiele and Crowley's representation. They only represent the probability distribution of local operator output, whereas our classifier represents probability as a joint function of local operator output and operator position.

Various parts-based methods differ from each other in whether the parts are represented in a rigid or flexible configuration like ours. Lades et al. (1993) and Wiskott et al. (1997) allow for flexibility through a flexible graph. Burl et al. (1996, 1998) allow for flexibility through Gaussian models of part position. Prior parts-based methods are confined to features that are local in the spatial sense. We extend the idea of "parts" by considering a more general concept that includes locality in frequency and orientation.

Using tables to represent probability distributions is not uncommon. Swain and Ballard (1991) have used them for object recognition by color, and Schiele and Crowley (1996, 2000) for representing the quantized output of Gaussian derivatives. However, our classifier design is distinctive in that we use multi-dimensional tables that jointly represent local operator output with operator position.

Much object detection work uses a coarse-to-fine search heuristic to reduce computational time (Rowley et al., 1998; Geman and Flueret, 2001; Romdhani et al., 2001; Viola and Jones, 2001). Such methods first evaluate the entire original image at coarse resolution and then selectively evaluate the image at higher resolution based on the outcome of the lower-resolution evaluations. Our coarse-to-fine strategy takes natural advantage of a wavelet-base multi-resolution image representation.

2. Derivation of Functional Form of Classifier

Based on the design choices we have made in the previous section, we will derive the actual functional form of our classification algorithm and examine implications of its underlying assumptions. The final form that we will obtain at the end of this section will be:

$$\prod_k \prod_{x,y} \frac{P_k(f_k(x, y), [x/M], [y/M] \mid object)}{P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)} > \lambda \tag{3}$$

where $[v]$ denotes the rounded integer value of v, and $M = 4$ or 8. The derivation of Eq. (3) consists of the following series of transformations and approximations to the ideal, but infeasible, classifier we introduced in Section 1.3:

- Two generalizations to the ideal classifier form
- Wavelet transform of input pixels
- Three approximations:
  - Statistical independence of *parts*
  - Quantization of *part* value
  - Coarse quantization of *part* position

The derivation gives a complete record of all the modeling assumptions used in the design of the classifier. The performance of the classifier then directly depends on these assumptions and the accuracy of the statistics gathered for object and non-object classes. Explicit knowledge of these assumptions also helps us decide how to collect statistics for the classifier.

2.1. *Ideal Form of Classifier*

We introduced the hypothetical ideal classifier as a large table in Section 1.3. This classifier would be ideal in several ways. First, it is based on a full representation of the input: the classifier is based on a joint function of the entire raw input, not a selected or filtered portion of it. No information is lost from or added to the input. Second, this table minimizes the probability of classification error, assuming each entry in the table is labeled with the most probable classification. Finally, the table concisely represents the output by simply listing each input's classification and nothing extraneous, such as probability values. Of course, such a table is not feasible. It is not possible to enumerate every possible input in a table. Although not feasible, it provides a useful point of comparison with any feasible classifier, in particular the final functional form of our classifier Eq. (3).

2.2. *Generalizations to the Functional Form of Ideal Classifier*

There are several differences between the ideal classifier and the final functional form of our classifier in Eq. (3). Our classifier represents object and non-object properties separately, whereas the ideal table does not separate them. The left side of Eq. (3) outputs a continuous number, whereas the ideal classifier directly outputs a classification, *object* or *non-object*. To transform the ideal table into Eq. (3), we must first make the table more general. This generalization has important implications for the training of the probability distributions in Eq. (3).

Our first transformation is to generalize the output of the table from a binary value (*object*, *non-object*) to a posterior probability, $P(object \mid image)$ (see Table 2).

To re-derive the ideal classification table from the posterior probability, we can apply Bayes' decision rule: If the probability is greater than 0.5, the classifier decides that the object is present in the image.

Table 2. Ideal classifier using posterior probabilities.

(1, 1)	(1, 2)	...	(20, 20)	$P(object \mid image)$
0	0	...	0	0.000001
0	0	...	1	0.000003
...	...	...	...	...
35	45	...	28	0.87521
...	...	...	...	...
255	255	...	255	0.00004

Table 3. Ideal classifier using separate models for object and non-object probabilities.

(1, 1)	(1, 2)	...	(20, 20)	$P(Image \mid Object)$, $P(Image \mid Non\text{-}object)$
0	0	...	0	0.00000013, 0.013
0	0	...	1	0.00000032, 0.014
...	...	...	...	...
35	45	...	28	0.0092, 0.00045
...	...	...	...	...
255	255	...	255	0.00007, 0.03

To generalize the classification function further, we use Bayes' theorem to re-write the Bayes' decision rule in an equivalent form as a likelihood ratio test:

$$\frac{P(image \mid object)}{P(image \mid non\text{-}object)} > \lambda = \frac{P(non\text{-}object)}{P(object)} \quad (4)$$

If the likelihood ratio (left side) is greater than the threshold on the right side, the classifier decides the object is present. With this expansion, the classification table would include two entries for each input image window (see Table 3).

Writing Bayes' decision rule as a likelihood ratio test has several advantages. It is easier to collect statistics separately for the two class-conditional probability functions, $P(image \mid object)$ and $P(image \mid non\text{-}object)$, since they are based on separate sets of images, than it is to directly estimate the posterior probability function, $P(object \mid image)$. In this form, we also factor out the contributions of the prior probabilities, $P(object)$ and $P(non\text{-}object)$, and combine them in a single threshold, λ. This threshold controls the sensitivity of the classifier.

Several disadvantages arise when using a decision rule based on class-conditional probabilities. They come mostly from practical limitations in the number of available training examples. First, by estimating $P(image \mid object)$ and $P(image \mid non\text{-}object)$ separately, we may be estimating more parameters than necessary in a direct classification function, such as a posterior probability function, and as a result, our estimation errors for the model parameters could be greater than those in a more tightly constrained classification function. Second, it is difficult to obtain a truly representative set of training images, particularly for the *non-object* class. Statistical estimates

based on these limited sets will not be as accurate and classification accuracy will suffer. In Section 3.4, we will explain how the technique of AdaBoost with Confidence Weighted Predictions (Shapire and Singer, 1999) partially compensates for such deficiency by weighting the training samples such that the resulting classifier minimizes classification error on the training set.

2.3. *Wavelet Transform of Image Window*

The classifier performs a wavelet transform on the input window using a linear phase 5/3 perfect reconstruction filter bank (Strang and Nguyen, 1997). This wavelet transform is fully invertible, and thus this transform has no consequences in terms of information content. Yet it has two advantages. First, the wavelet transform partially decorrelates natural imagery, so smaller subsets of variables will capture greater statistical dependency within the image. Second, the wavelet transform makes it convenient to design *parts* with locality in frequency and orientation, as well as locality in space. The multi-resolution nature of wavelets also allows us to efficiently search the image in a coarse-to-fine manner.

2.4. *Three Approximations to the Generalized Ideal Form of the Classifier*

We make three approximations to the generalized ideal form of the classifier, Eq. (4): statistical independence of *parts*, quantization of local operator outputs, and reduced resolution in *parts* position representation. We also describe the design of the local operators whose outputs form the *parts*.

2.4.1. Statistical Independence of Parts. Our most significant approximation is to decompose the image window into *parts* that are treated as statistically independent. As illustrated in Fig. 5, various local operators are evaluated at all possible positions within the image window, and each *part* corresponds to a local operator output. However, Fig. 5 is a simplified illustration. These operators actually sample a fixed arrangement of wavelet coefficients, instead of directly sampling the input pixels. Thus, each *part* represents a different, but not necessarily disjoint, subset of wavelet coefficients. An operator translates by moving its arrangement of wavelet coefficients as a block in steps of one wavelet coefficient.

The local operators sample the wavelet transform in many ways. For example, as shown in Fig. 6, one

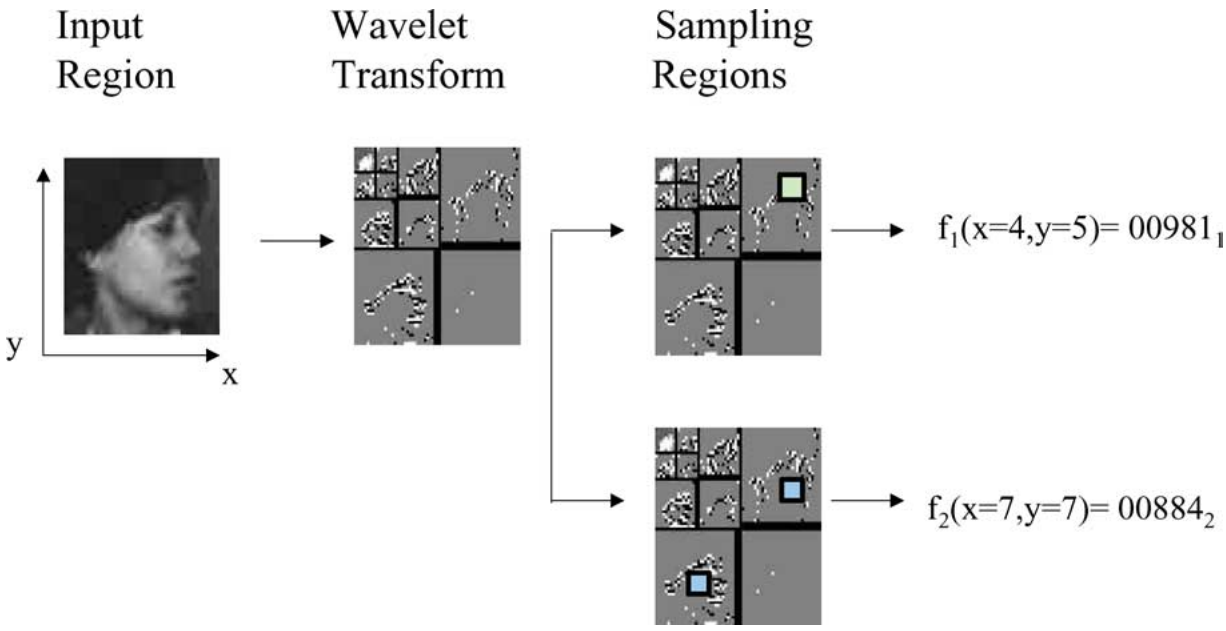


Figure 6. We may define many operators, each of which samples a certain arrangement of wavelet coefficients. In this figure, operator "1" samples a block of coefficients from within one subband and operator "2" combines spatially registered blocks from two subbands.

operator samples a block of wavelet coefficients within one wavelet subband. Another operator combines two blocks from two different subbands. In the next subsection we describe all the types of local operators our classifier uses.

Under the assumption of *part*s independence, the form of classifier, Eq. (4), now becomes:

$$\prod_k \prod_{x,y} \frac{P_k(f_k(x, y), x, y \mid object)}{P_k(f_k(x, y), x, y \mid non\text{-}object)} > \lambda \qquad (5)$$

where $f_k(x, y)$ is the kth operator output at position (x, y).

This assumption of statistical independence greatly reduces the complexity of the classifier. We believe this is a reasonable assumption because for faces, cars, and many other objects, a given coefficient on the object is strongly statistically dependent only on a few other coefficients and is weakly related with the rest. We hope to capture the stronger dependencies by appropriate choices of *parts* and pay a limited penalty by neglecting the weaker dependencies among the coefficients.

We can gain another perspective on this simplification by taking the logarithm of Eq. (5) making the classifier a sum of log probabilities:

$$\sum_k \sum_{x,y} \log\left(\frac{P_k(f_k(x, y), x, y \mid object)}{P_k(f_k(x, y), x, y \mid non\text{-}object)}\right) > \log \lambda \qquad (6)$$

In this form, we can interpret the classifier as a linear discriminator:

$$\sum_k \sum_{x,y} w_k(x, y)^t a_k(x, y) > \log \lambda$$

where $w_k(x, y)$ is a vector concatenating the log likelihood values corresponding to each value of operator k at position (x, y) and $a_k(x, y)$ selects the appropriate log likelihood value from this vector, given by the computed value of operator k, by assigning '1' to one of its elements and '0' to the remaining elements.

We can also view Eq. (5) as a modification of the naïve Bayes classifier. The naïve Bayes classifier models all variables as statistically independent, whereas our approach models groups of variables as statistically independent. Domingos and Pazzani (1997) have demonstrated that the naïve Bayes classifier performs surprisingly well in a number of classification problems, even when there is significant statistical dependency among the independently modeled components. Although there is not a full theoretical understanding of why this is true, Domingos and Pazzani show that the classifier is optimal for conjunctions and disjunctions and other problems in which statistical independence does not necessarily hold.

2.4.2. Design of Local Operators. At this point in the derivation, the central question we face is how to design local operators. Our goal is to design local operators that capture common statistical dependencies. We emphasize the notion of common for two reasons. First, we seek to represent statistical dependencies that exist both for the object (e.g., face) and in the rest of the world (e.g., non-face). Second, since the local operators are each scanned over the full extent of the input window, we need local operators that are not just specialized to one site but useful at all sites on the object.

Our approach is to make educated guesses about the types of statistical dependency we might encounter. We would expect statistical dependencies are stronger in localized regions, and that as pixels are farther apart, dependency decreases. We therefore emphasize locality in position when designing local operators. In particular, we want operators to be capable of capturing the statistical dependencies that exist in small, highly detailed structures, such as the eyes, nose, and mouth on a human face or the headlights and grille on a car. However, statistical dependency can exist over large regions as well. In these larger regions, the dependencies usually involve lower-frequency attributes. For example, on a face we would expect that the eye sockets would be darker than the forehead and cheeks. To represent dependencies over both small and large regions, we combine spatial locality with locality in frequency. Some operators represent large areas at coarse resolution (localized coefficients in the upper levels of the wavelet transform) and others represent smaller areas at high resolution (localized coefficients in the lower levels of the transform). Locality in orientation is another factor we need to combine in operator design. Since the physical world tends to be continuous, we would expect horizontal edges to co-occur with other horizontal edges. Similarly, we would expect vertical edges to co-occur with other vertical edges.

We have defined 17 local operators that comprise various combinations of locality in space, frequency, and/or orientation. Each operator consists of a moving arrangement of wavelet coefficients. As an operator scans across the image window, this arrangement moves as a block.

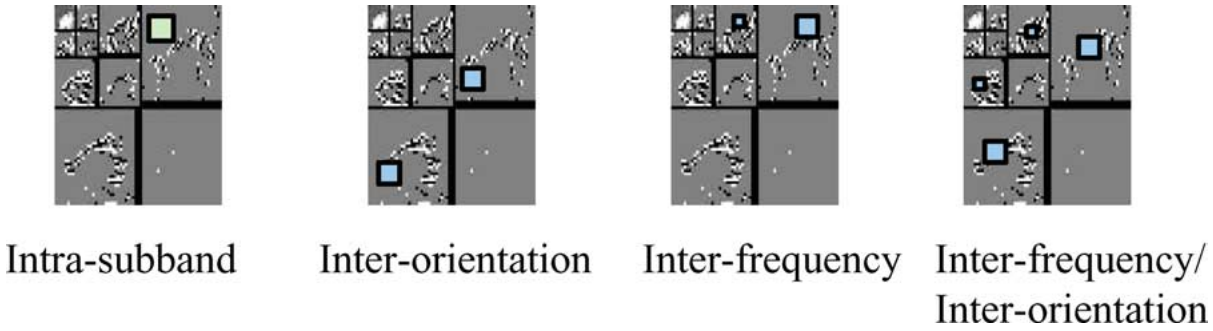


Figure 7. Different local operator types.

The operators are divided into four categories[1] by the composition of their arrangements of wavelet coefficients: intra-subband, inter-orientation, inter-frequency, and combined inter-frequency/inter-orientation arrangement. They are illustrated in Fig. 7. See Appendix for a description of the notation we use for describing the components of the wavelet transform.

The first six operators (see Table 4) out of 17 are intra-subband operators, $Op_b(level, orientation)$. These operators sample a contiguous block of coefficients within one subband specified by the (*level*, *orientation*) combination. Such operators capture features that are jointly localized in space, frequency, and orientation.

The next four operators (see Table 5) are of type inter-orientation, $Op_o(level, orientation_1, orientation_2)$. These operators combine coefficients from two subbands of different orientation but within the same level (frequency) in the transform. These capture features that have both horizontal and vertical components but are otherwise localized in space and frequency.

The next six operators (see Table 6) are inter-frequency operators, $Op_f(level_1, level_2, orientation)$. These jointly sample from two subbands of the same orientation but different levels (or frequencies). They capture features that have broad frequency content, such as edges.

Finally the last operator (see Table 7) combines coefficients across multiple subbands in both orientation and frequency, $Op_{of}(level_1, level_2, orientation_1, orientation_2)$. The operator, called a combined inter-orientation/inter-frequency operator, is useful for

Table 4. Intra-subband operators, $Op_b(level, orientation)$.

Operator	Level	Orientation
1	3	LH
2	3	HL
3	2	LH
4	2	HL
5	1	LH
6	1	HL

Table 5. Inter-orientation operators $Op_o(level, orientation_1, orientation_2)$.

Operator	Orientations	Level
7	LL (horizontal), LL (vertical)	3
8	LH, HL	3
9	LH, HL	2
10	LH, HL	1

Table 6. Inter-frequency subbands $Op_f(level_1, level_2, orientation)$.

Operator	Levels	Orientation
11	3 LL (horizontal), 3	LH
12	3, 2	LH
13	2, 1	LH
14	3 LL (vertical), 3	HL
15	3, 2	HL
16	2, 1	HL

Table 7. Inter-orientation operators $Op_{of}(level_1, level_2, orientation_1, orientation_2)$.

Operator	$Level_1$	$Level_2$	$Orientation_1$	$Orientation_2$
17	3	2	LH	HL

features that combine horizontal and vertical information and information across frequency.

2.4.3. Quantization of Local Operator Output. The output of each operator is a discrete value of finite range representing the combination of wavelet coefficients. In this section we explain why we choose a discrete representation and describe how we quantize each subset of wavelet coefficients to a discrete value. Our classifier uses a discrete representation of operator output because it permits us to represent the statistics of operator output using a table.

There are alternative statistical representations to a table-based representation. The most flexible ones are non-parametric ones, such as nearest-neighbor, Parzen windows, and other kernel-density models. These models, however, have a high computational cost for probability retrieval. Probability retrieval, for a given input, involves a large computation requiring comparison of the input to every example in the entire training data. Another alternative would be to use a flexible parametric distribution, such as a mixture model or a multi-layer perceptron (artificial neural network). Each of these has flexibility to model multi-modal distributions and acts as a universal approximator as its resource size (i.e., hidden units or nodes) increases. However, there are no closed form solutions for fitting these models to a set of training examples; their parameters must be estimated by iterative procedures, such as gradient descent (backpropagation) and E-M. These parameter estimates could become trapped in local minima and become sub-optimal. Also, it is questionable whether such models are appropriate for the task of detection. For example, a multi-layer perceptron forms decision boundaries that are combinations of hyperplanes to a first approximation. It is not clear whether such decision boundaries will be good for separating two or more classes in a high dimensional space (Gori and Scarselli, 1998; Kung, 1993).

Table representation of a probability distribution is almost as flexible as various non-parametric methods. Tables have the advantage of being able to retrieve probabilities directly by one table look-up rather than a computation over the entire set of training data. Estimation of a table is relatively straightforward. We simply build a histogram by counting how often each value occurs in the training data. This process involves just one pass through the training images. The resulting estimates are statistically optimal—maximum likelihood, no bias, consistent, and efficient, satisfying the Cramer-Rao lower bound. The main drawback of a table is that we can only represent probability over a limited range of discrete values; the amount of training data and the size of memory impose limitations on the size of the probability table. To get reasonably accurate estimates for our tables, we limited ourselves to approximately 10,000 discrete values per operator. This limitation restricts the number of wavelet coefficients each operator samples and the resolution at which the operator quantizes its subset of coefficients.

Our compromise is to quantize the output of each operator to 3^8 discrete values. We do so by having each operator sample the arrangements of 8 coefficients and quantize each coefficient to three levels. Note that the quantization threshold may differ from operator to operator, so different operators may sample the same coefficient but quantize it differently. The 3-level quantization is fairly coarse but still retains the information content of the image. Figure 8 shows several pairs

Figure 8. Images reconstructed by inverse wavelet transform. All wavelet coeffients in LH and HL bands were quantized to three levels per coefficient.

of original and reconstructed images where the reconstruction is done by an inverse wavelet transform of the quantized wavelet values. Objects in the reconstructed images are still easily identifiable as faces and cars.

2.4.4. Reduced Resolution in Part Position. Our last approximation reduces resolution in representing *part* position (x, y). Instead of representing $P_k(f_k(x, y), x, y \mid object)$ and $P_k(f_k(x, y), x, y \mid non\text{-}object)$ at each position (x, y) in the original resolution, we reduce resolution by a factor of M. A given position (x, y) is represented at reduced resolution by $([x/M], [y/M])$ where $[v]$ denotes the rounded integer of v. This approximation reduces the size of each probability table by a factor of M^2. With this simplification, the final form of the classifier is (repeat of (3)):

$$\prod_k \prod_{x,y} \frac{P_k(f_k(x, y), [x/M], [y/M] \mid object)}{P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)} > \lambda \tag{7}$$

Or alternatively, in terms of log likelihood functions:

$$\begin{aligned} &\sum_k \sum_{x,y} L_k(f_k(x, y), [x/M], [y/M]) > \log \lambda \\ &L_k(f_k(x, y), [x/M], [y/M]) \\ &\quad = \log \frac{P(f_k(x, y), [x/M], [y/M] \mid object)}{P(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)}. \end{aligned} \tag{8}$$

As the reduction, we use either $M = 4$ or 8 depending on the *part*; for level 1 and 2 operators, we reduce $M = 4$ (a factor of table reduction $=16$), and for level 3 *parts*, $M = 8$ (a factor of table reduction $=64$). We found that reducing the resolution does not drastically compromise the geometric representation of the classifier. Interestingly, the resolution reduction seems to implicitly accommodate small variations in the arrangement of *parts* as an unintended but positive side effect.

3. Collecting Statistics

This section describes how we gathered the statistics that go into each set of tables $P_k(f_k(x, y), [x/M], [y/M] \mid object)$ and $P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)$ where $k = 1 \ldots 17$ in Eq. (7). We estimate each of these tables by using a large set of labeled and pre-processed training images. For the probabilities tables conditioned on the object, $P_k(f_k(x, y), [x/M], [y/M] \mid object)$, we use images of the object, and for $P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)$, images that do not contain the object. Preparing the training set for the "object" class is relatively straightforward, but preparing the training set for the "non-object" class requires some consideration. We will discuss both a basic training algorithm by which we estimated each table separately and an alternative training procedure that minimizes a classification error criterion using AdaBoost with Confidence Weighted Predictions (Shapire and Singer, 1999).

3.1. Pre-Processing Images of the Object

Each training image is geometrically normalized and aligned, corrected for lighting, and perturbed to create many synthetic variations.

3.1.1. Size Normalization and Spatial Alignment. To standardize training images, we aligned all the images with respect to a prototype using pre-defined, hand-labeled landmark points on the object. For example, for frontal faces, we used the locations of the eyes, the bridge of the nose, and the center and sides of the mouth. Using these landmark points, we applied the translation, scaling, and rotation (Arun et al., 1987) that brought each image into alignment with the prototype.

3.1.2. Intensity Normalization. We normalized the image intensity as well. This normalization procedure is object-dependent.

For faces, we normalized the left and right sides of each training image separately, by scaling all the intensity values with specified correction factors, α_{left} and α_{right}:

$$\begin{aligned} I'(x, y) &= \alpha_{\text{left}} I(x, y) \\ I'(x, y) &= \alpha_{\text{right}} I(x, y) \end{aligned} \tag{9}$$

For a local operator that uses inputs from both sides of the face, we normalize the entire sample using the value of α of the center pixel. In training, we chose these correction factors by hand for each training example by visually comparing them to a group of prototypes.

When the face detector searches for a face, the classifier evaluates both sides of each candidate window over a set of 5 discrete values for α. For each side, the classifier compares the responses (sum of the log likelihoods) for each value of α and chooses the largest response. It then sums the two chosen responses to obtain the total log likelihood for the candidate.

For cars we did not normalize the training images in intensity, because cars differ greatly in color, making normalization difficult to compute.

3.1.3. Creating Synthetic Variants of Training Images. We generated additional training data by artificially creating variations to the original training images. The purpose in doing so was to increase the accuracy of our probability estimates.

For each image, we generated between 1,600 and 6,400 synthetic variations through small, controlled variations in position (both by positional perturbation and overcomplete evaluation of the wavelet transform), orientation, size, aspect ratio, background scenery, lighting intensity, and frequency content. We applied these variations after we first aligned the image geometrically and corrected for the lighting as described above. For substitution of different background scenery, we segmented the objects from the background in many of the training images. We segmented these images either by hand and by automatic methods when possible. Also, we modified frequency content by using various low-pass filters.

3.2. Non-Object Images

We collected non-object images from various photography collections[2] and from the World Wide Web, particularly the Washington University archive.[3] We tried to get a balance of indoor, outdoor, urban, industrial, and rural scenes. We used more than 2,000 such non-object images.

To select non-object samples, we used the bootstrapping technique.[4] The goal of bootstrapping is to select non-object training examples that resemble the object. By doing so, the training data will emphasize the distinctions between the object and non-object classes and performance will be improved. Bootstrapping is a two-stage process. We first trained a preliminary detector using non-object image windows drawn randomly from the non-object image collection. We then ran this preliminary detector over the entire collection of non-object images, selecting additional non-object training examples where the detector gave a high response. For some detectors we repeated this process several times, gathering more and more non-object samples. We then used the combined set of samples for training the final detector.

3.3. Training Method (I)—Probabilistic Approximation

The most direct way to build our classifier was to separately estimate each of its constituent probability distributions, $P_k(f_k(x, y), [x/M], [y/M] \mid object)$ and $P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)$. To estimate these distributions, we simply counted how often each value of local operator output value occurs at each position in the appropriate set of training examples. For a local operator, k, this process involves building a histogram, $H_k(i, j, f)$ over a combination of operator output values, f, and positions, i and j:

Initialize all bins in histogram, $H_k(i, j, f)$, to zero
For all training images, I_p
For each local operator, $f_k, k = 1 \ldots N$
For each position, (x, y)

$$
\begin{aligned}
i &\leftarrow [x/M] \\
j &\leftarrow [y/M] \\
f &\leftarrow f_k(x, y, I_p) \\
H_k(i, j, f) &\leftarrow H_k(i, j, f) + 1
\end{aligned}
$$

Occasionally a bin in the histogram may receive zero count. Since it may not be desirable to actually assign zero probability, we simply added one to each bin in the histogram:

$$H_k(i, j, f) \leftarrow H_k(i, j, f) + 1 \quad \text{for all } k, i, j, f$$

3.4. Training Method (II)—Minimization of Classification Error Using AdaBoost

Training class-conditional distributions separately, as described above, will give the best possible performance (with this functional form) if our training data is truly representative. However, a finite set of training data will have limitations, particularly in representing the non-object class. To achieve better results, we can explicitly train our classifier to minimize classification error over the training set. For this purpose, we use the algorithm of AdaBoost with Confidence Weighted Predictions, a modification of standard AdaBoost (Freund and Shapire, 1997). This modification of standard AdaBoost allows for the base classifier to output a continuous value, proportional to confidence, rather than a binary classification as in the standard formulation.

AdaBoost, in general, works by sequentially training multiple instances $h_1(x), h_2(x), \ldots, h_T(x)$ of a base

classifier $h(x)$. In our case, the base classifier takes the following form (modified from Eq. (8)):

$$h(I) = \sum_k \sum_{x,y} L_k(f_k(x, y), [x/M], [y/M]) - \log \lambda$$

$$L_k(f_k(x, y), [x/M], [y/M]) = \log \frac{P(f_k(x, y), [x/M], [y/M] \mid object)}{P(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)} \quad (10)$$

Where the input to our classifier is an image window denoted by I.

Each classifier, $h_t(x)$, is trained by assigning different weights to the training examples. For the first classifier, $h_1(x)$, all training examples are given equal weight. For the subsequent classifiers $h_t(x)$, $t > 1$, the algorithm assigns more weight to training examples that have been incorrectly classified by the previous classifier, $h_{t-1}(x)$ and, conversely, less weight to training examples that were correctly classified. AdaBoost then takes a weighted sum of these classifiers as the final classifier $H(x)$:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (11)$$

Both the standard AdaBoost algorithm and the AdaBoost with Confidence Weighted Predictions algorithm determine each α_t as a function of the accuracy of $h_t(x)$ on the training data, but differ in how they do this. In AdaBoost with Confidence Weighted, α_t can be determined by a binary search that minimizes a function of the margin.[5] Both algorithms guarantee that the final classifier satisfies a bound on the classification error on the training set. With such a bound the classification error can be driven toward zero after a few iterations, and thereafter the margin between the two classes can be increased.

In practice, for each $h_t(x)$, we estimate the distributions $P_k(f_k(x, y), [x/M], [y/M] \mid object)$ and $P_k(f_k(x, y), [x/M], [y/M] \mid non\text{-}object)$ by Training Method (I), but instead of incrementing each histogram bin by 1, we increment by the weight assigned to the training example. We scale and round the training image weights to integers for this purpose.

A disadvantage of Adaboost is the increased computational cost of applying all instances of the classifier in the sum of Eq. (11). However, in our case, this sum of classifiers does not actually increase the complexity of our overall classifier, Eq. (10). Since our classifier is linear, the linear sum in Eq. (11) actually reduces to the complexity of the base classifier. We can see this by directly substituting Eq. (10) in Eq. (11), giving:

$$H(I) = \sum_{t=1}^{T} \sum_k \sum_{x,y} \alpha_t L_{t,k}(f_k(x, y), i(x), j(y)) - \log \lambda$$

$$L_{t,k}(f_k(x, y), i(x), j(y)) = \log \frac{P_{k,t}(f_k(x, y), i(x), j(y) \mid object)}{P_{k,t}(f_k(x, y), i(x), j(y) \mid non\text{-}object)}$$

And by changing the order of the summations,

$$H(I) = \sum_k \sum_{x,y} L_k(f_k(x, y), [x/M], [y/M]) - \log \lambda$$

$$L_k(f_k(x, y), [x/M], [y/M]) = \sum_{t=1}^{T} \alpha_t L_{t,k}(f_k(x, y), [x/M], [y/M]) \quad (12)$$

Adaboost assigns appropriate weights for individual training images. Intuitively, it is not optimal to give all the training images equal weight, particularly among the non-object class. Some of the non-object examples are more important than others for determining the decision boundary. This re-weighting of the training examples is analogous to the way support vector machines identify the training examples that directly affect the placement of the decision boundary (Cortes and Vapnik, 1995).

One of the issues in using AdaBoost is when to stop the iteration. It is not well understood if AdaBoost is susceptible to overfitting as the number of iterations increase. Our approach was to monitor the performance of the classifier using a cross-validation image set and to stop the algorithm when performance seemed to stop improving.

4. Implementation and Efficient Processing for Detection

Our strategy for implementing a fast detector was to re-use multi-resolution information wherever possible and to use a coarse-to-fine search strategy together with various other heuristics to prune out unpromising object candidates.

4.1. Exhaustive Search

Each classifier is specialized for a specific orientation, size, alignment, and intensity of the object within an

image window. Detecting an object at any position in an image requires exhaustively scanning the classifier in position, size, and intensity, as illustrated in Fig. 2.

In searching across scale, the detector searches in 4 scales per octave, that is, in scale increments of $2^{1/4}$ in both x and y dimensions. We chose an integer root of 2 so we could reuse information at each octave in this search through scale.

Given an input image, it is ideal to perform intensity correction for each candidate. However, doing so would require the classifiers to re-compute all subsequent operations separately for each candidate. We can reduce computation cost by sharing computation among the candidates. Our normalization, therefore, pre-computes the operator values for 5 discrete levels of intensity correction, α (see Section 3.1.2). For each candidate, the classifier then evaluates the sum of the log likelihood for each half of the candidate at each of these five different intensity corrections, and for each half, selects the α value giving the largest sum of log likelihood.

The detector repeats this exhaustive search for each view-based classifier and combines their results. If there are multiple detections at the same or adjacent locations and/or scales, the detector chooses the strongest detection.

4.2. *Coarse-to-Fine Evaluation Strategy*

Let us consider the computation process within one resized image in the search across scale illustrated in Fig. 2. We organize this process so that as much computation as possible is shared among overlapping candidate windows. First, rather than computing the wavelet transform separately for each image window, the detector computes it once for the entire image. Since the wavelet transform is not shift invariant, we compute an overcomplete transform for each level of the wavelet transform by expanding the (even, even) phase from the previous level. We also compute local operator output for each value of intensity correction, α, at every candidate location. The detector classifies each candidate by accessing the appropriate local operator values and retrieving and summing their log likelihoods according to Eq. (8).

In practice, however, the detector rarely needs to evaluate the entire log likelihood ratio. Since the left side of Eq. (8) is a summation, the classifier can examine its value after any partial evaluation and rule out the presence of the object if the value is not high enough. We apply this strategy after each local operator is applied. We set these thresholds conservatively in order to avoid discarding actual object candidates while quickly removing many of the non-object candidates. The detector orders the local operators, evaluating first the coarse resolution ones (those involving coefficients from the top level of the wavelet transform). That is, in the first stages of evaluation, the detector evaluates the image over a coarse grid in which candidate windows are spaced 8 pixels apart. The remaining stages reduce resolution by a factor of 4. Through this strategy we found that we could reduce our computational requirements by 1 to 2 orders of magnitude with little loss in accuracy.

The candidate windows whose likelihood ratio exceeds the final threshold are initially marked as detections. Each real object in the image tends to produce a cluster of detections. Figure 9 shows the raw output from Eq. (8) (before thresholding) over 4 scales. The value shown at each pixel corresponds to the log likelihood sum associated with the window centered at that position. The responses to one face form a cluster of detections across position and scale. We now merge these detections using the following strategy. We determine the detection that has the highest response in the entire image and it classifies it as "object." The detector then discards all detections within a radius of the object and within one half to twice the object size. The detector then continues to search among the remaining detections, finds the one with the next-highest response, and continues this process until all the candidates have either been classified as "object" or discarded.

4.3. *Re-Using Wavelet Transform in Search Across Scale*

In searching for the object across scale, the detector iteratively searches resized versions of the input image where the image is reduced in size by a factor of $2^{1/4}$ in each dimension. This scaling process continues until the scaled image is smaller than the window sent as an input for the classifier. Recomputing the entire wavelet transform is not necessary for obtaining the wavelet transform for each successive scale. This is illustrated in Fig. 10. All 3 levels of the wavelet transform must to be computed for each of the first 4 scales, which comprise the first octave of the search across scale. After these first 4 scales, our detector re-uses portions of the wavelet transform computed at previous scales. In particular, we obtain the wavelet transform at scale

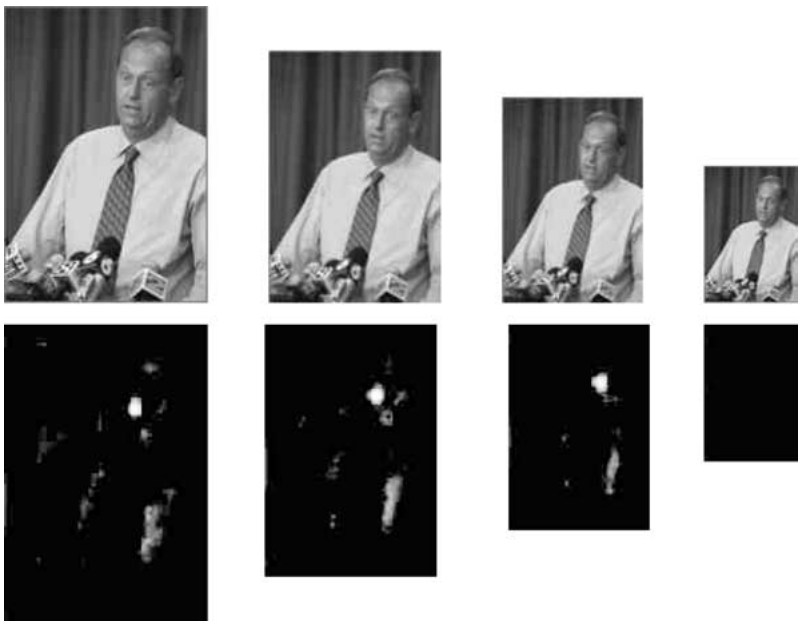


Figure 9. Four consecutive scales and the corresponding output before thresholding from the detector.

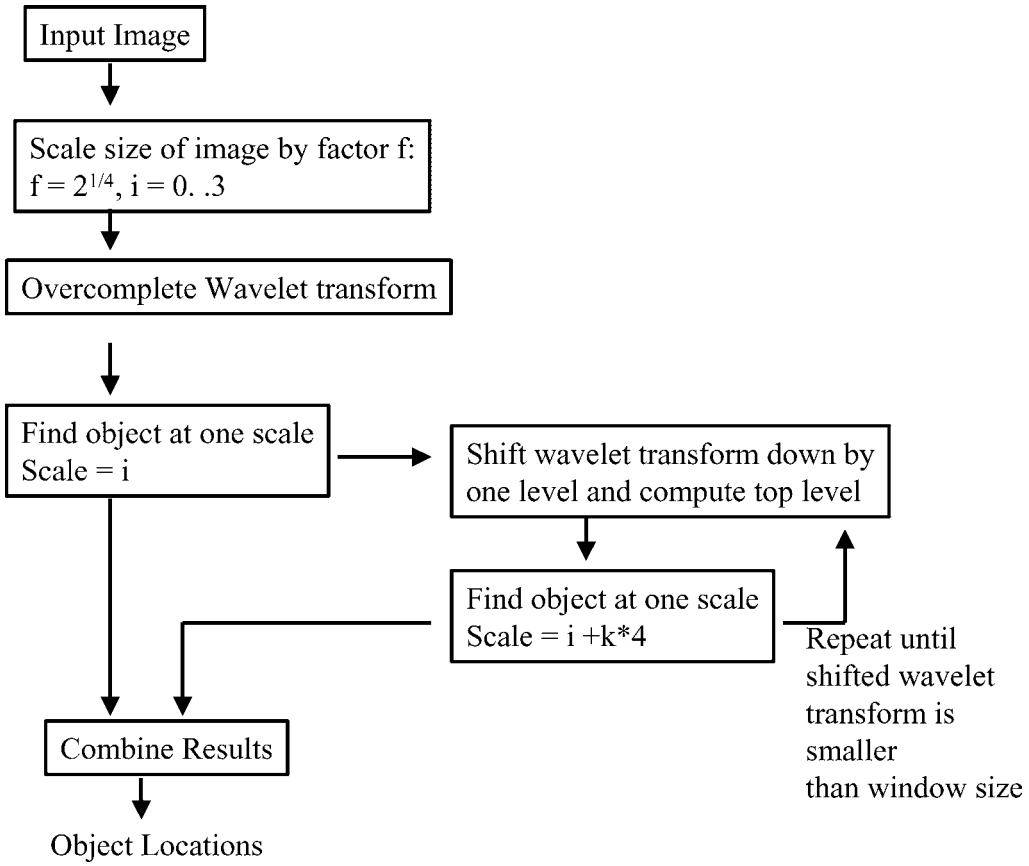


Figure 10. Re-using the wavelet transform at successive scalings.

Table 8. Face detection results on Kodak test set.

(Rowley, 1999)		Schneiderman and Kanade (using AdaBoost)			
Detection (%)	False detections	γ	Detection (all faces) (%)	Detection (profiles only) (%)	False detections
58.7	1347	0.5	80.4	86.1	105
41.3	617	1.0	70.0	69.4	7
32.6	136	1.5	63.0	61.1	1

i from the wavelet transform at scale i-4 by shifting it by one level; that is, level 3 becomes level 2, and level 2 becomes level 1. Therefore, only level 1 has to be recomputed at scale i.

4.4. *Color Heuristics*

The detector we have described is designed for gray-scale images. When color images are available, color pre-processing is sometimes useful for pruning out unpromising candidates. This pre-processor uses $8 \times 8 \times 8$ probability tables (8 levels of quantization for each color band) to represent the color distribution of skin-color and non-skin color in RGB space and combines them into a likelihood ratio test for evaluating each candidate location. This pre-processing improves performance speed by a factor of 2 to 4, and discarded a few candidates that would have otherwise been false

Figure 11. Face detection results.

detections. However, a disadvantage of using color is that the detector removes many actual faces if the images are poorly color balanced.

4.5. Performance Time

Using the full set of heuristics described above, one classifier can evaluate a 240 × 256 image over a full range of scalings in 5 seconds, on average, using a Pentium II at 450 MHz.

5. Face Detection

In this section we describe specific design choices that comprise our face detector and discuss its accuracy on several sets of test images. Our face detector is available on-line at http: // www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi and it allows internet users to submit their own images and see the detection results.

5.1. Local Operators, Training Images, and Training

The face detector uses the 17 local operators described in Section 2.4.1.

We gathered a large number of face images for training from a number of sources: FERET,[6] NIST mug shot database,[7] Harvard's face database,[8] and CMU's face collection.[9] We also used many images we collected from the World Wide Web. Overall, we gathered about 2,000 images of frontal faces and 2,000 images of profile faces. We normalized the two sides of the face in the training images to compensate for situations in which the face was illuminated unevenly.

We trained the face classifiers using Training Method II (the AdaBoost method) described in Section 3.4.

5.2. Results in Face Detection

Table 8 compares the performance of our face detector with that reported by Rowley (1999) for the task of both frontal and profile face detection using images selected from proprietary images Kodak provided to Carnegie Mellon University. The test set consists of 17 images with 46 faces, of which 36 are in profile view (between 3/4 view and full profile view). These images contain some of the typical problems of amateur photographs including poor lighting, contrast, or focus.

Each row of Table 8 shows the result by a different setting of γ, which determines λ for each classifier as follows:

$$\lambda_{\text{front}} = d_{\text{front}} + e_{\text{front}}\gamma$$
$$\lambda_{\text{profile}} = d_{\text{profile}} + e_{\text{profile}}\gamma$$

where the d's and e's were tuned by hand.

Table 9. Face detection results on Schneiderman & Kanade test set.

	With AdaBoost			Without AdaBoost	
γ	Detections (all faces) (%)	Detection (profiles) (%)	False detections	Detection (all faces) (%)	False detections
0.0	92.7	92.8	700	82	137
1.5	85.5	86.4	91	74	27
2.5	75.2	78.6	12	60	3

Table 10. Frontal face detection on Sung & Poggio and Rowley & Baluja & Kanade combined test set[a].

	Detection rate rate (%)	False detections
Schneiderman and Kanade[b] (eigenvector)	94.4 (95.8)	65
(Roth et al., 1999)[c]	(94.8)	78
Schneiderman and Kanade (wavelet)[c]	90.2 (91.8)	110
(Rowley et al., 1998)	86.0	31
(Viola and Jones, 2001)	93.7	167

[a]At least 10 additional human faces are not labeled in the ground truth for this test set. We report our results in two ways. The figures not in parentheses indicate results on just the 483 labeled faces. Any additional detected faces were counted neither as detections nor false detections. To be consistent with Roth et al. (1999), we also indicate, in parentheses, the ratio between the total number of faces found by computer (labeled and unlabeled) divided by the number labeled by hand (483).

[b]Indicates the detection results on 125 images with ground truth of 483 labeled faces. The original MIT/CMU test set included 5 additional images of line-drawn faces.

[c]Indicates that 5 images of line-drawn faces were excluded, leaving 125 images with 483 hand-labeled faces. However, at least 10 additional human faces are not labeled in the ground truth for this test set. The figures in parentheses indicate results on just the 483 labeled faces. These numbers do not include the additional faces we detected, counting them neither as detections nor false detections. However, to be consistent with Roth et al. (1999), we also indicate, in parentheses, the ratio between the total number of faces found by computer and the number labeled by hand.

Table 11. Car detection results.

γ	Detections	Misses	False detections
1.05	177 (83%)	36 (17%)	7
1.0	183 (86%)	30 (14%)	10
0.9	197 (92%)	16 (8%)	71

To further evaluate accuracy on faces with out-of-plane rotation we collected a larger test set consisting of 208 images with 441 faces that vary in pose from full frontal to side view. This test set is available on-line at http://www.ri.cmu.edu/projects/project_419.html. Of these images, approximately 347 are profiles (between 3/4 view and full profile view). We gathered these images from a variety of sites on the World Wide Web, mainly news sites such as Yahoo! and the New York Times. Most of these images were unconstrained in terms of content, background scenery, and lighting, but were taken by professional photographers and are generally of better quality than the Kodak images in terms of composition, contrast, and focus. Table 9 shows the performance at different values of γ controlling the sensitivity of the detectors. The table also compares the performance of the detectors trained with AdaBoost and without AdaBoost. Figure 11 shows some typical results on this image set when our detector was trained with AdaBoost and used $\gamma = 1.0$.

The distinguishing characteristic of our face detector is that it works for both frontal and out-of-plane rotational views. To date, several researchers (Rowley et al., 1998; Sung and Poggio, 1998; Osuna et al., 1998; Roth et al., 1999; Viola and Jones, 2001) have had success developing algorithms that work for frontal views of faces, but none, to our knowledge, have had success with profile (side) views except Rowley (1999) which we compare our algorithm to in Table 8.

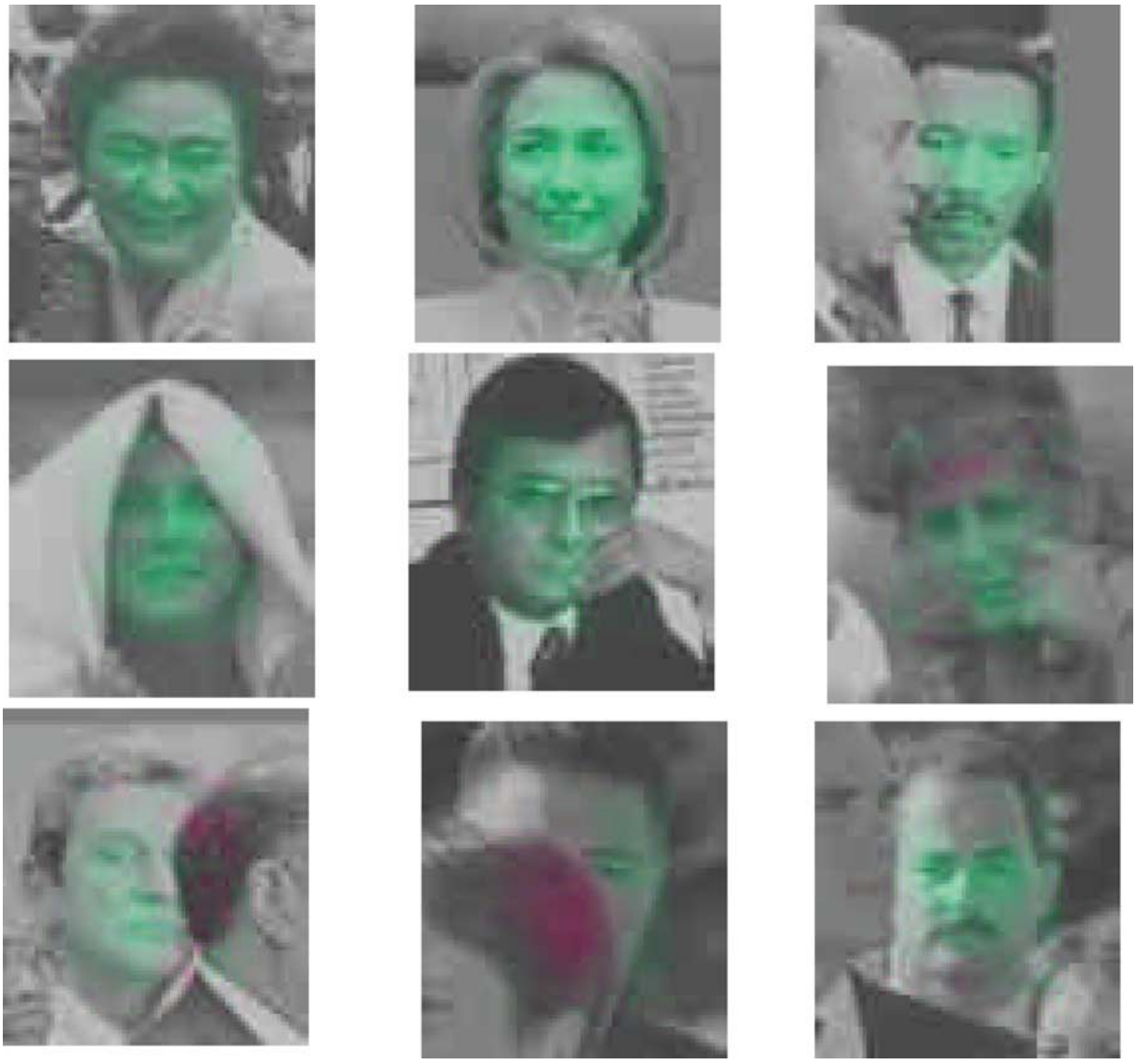
Profile-view faces are more difficult to detect than frontal views for several reasons. The salient features on the face (eyes, nose, and mouth) are not as prominent

Figure 12. Car detection results.

when viewed from the side as they are when viewed frontally. Also, for frontal views, these features are interior to the object, whereas on a profile, many of the features form the silhouette with respect to the background. Since the background can be almost any visual pattern, a profile detector must accommodate much more variation in the appearance of these features than a frontal detector needs to accommodate for interior features.

Table 10 compares the accuracy of our detectors with those of other researchers on the MIT/CMU test set of frontal face images combining test images from Sung and Poggio (1998) and Rowley et al. (1998).

In these experiments, we noticed some differences in performance between the detector described in this paper and an improved version of the detector we described in Schneiderman and Kanade (1998). Both detectors use similar probabilistic structures, but the



(a)

Figure 13. (a) Positional response of the classifier. Green areas are "face-like" and red areas are not "face-like." (b) Positional decomposition of classifier response to particular profile faces. Green areas are "face-like" and red areas are not "face-like."

(*Continued on next page*.)

(b)

Figure 13. (*Continued.*)

detector in Schneiderman and Kanade (1998) uses local operators based on localized eigenimages rather than wavelet coefficients. The wavelet-based detector described in this paper performs much better for profile-view faces. However, the localized eigenimage-based detector in Schneiderman and Kanade (1998) seems to be slightly more accurate on frontal faces.

6. Car Detection

We also trained a detector for finding passenger cars in an image.

6.1. Local Operators, Training Images, and Training

We used 13 of the 17 operators described in Section 2.4.1 by excluding the four operators that involved the level 3 LL subband. We excluded these operators because they represent average intensities over large areas. Since cars come in all colors and intensities, we expected these coefficients not to be informative.

We collected car images with our own camera and from the World Wide Web, mostly from car sales and car enthusiast sites. The latter sites provided many

photographs of older cars. We gathered between 250 and 500 images per viewpoint with a total of over 2,000 images.

The car detector used Training Method (I) described in Section 3.3.

6.2. *Results in Car Detection*

To test the accuracy of the car detector, we collected, separately from the training set, a set of 104 images that contain 213 cars, spanning a wide variety of models, sizes, orientations, background scenery, lighting conditions and some partial occlusions. We gathered these images using several cameras and from sites on the World Wide Web. This image set is publicly available at http://vasc.ri.cmu.edu/idb/html/car/index.html. Table 11 displays our performance.

The sensitivity of the detectors is controlled by γ which linearly scales the detection thresholds. Figure 12 shows some typical results on this image set, evaluated at $\gamma = 1.0$.

7. Analysis of Positional Response of Classifier

Since the classifier uses *parts* across the full extent of the object, it is worth analyzing which *parts* or areas tended to be most influential. For example, are the eyes, nose, and mouth regions really the most important areas to detect faces? To study the behavior of the classifier, we computed the amount of contribution at each pixel (x, y) to the total log likelihood, Eq. (8). In other words, $C(x, y)$ is the partial sum of the total log likelihood due to the *parts* centered at (x, y).

$$C(x, y) = \sum_k L_k(f_k(x, y), [x/M], [y/M])$$
$$= \sum_k \log \frac{P(f_k(x, y), [x/M], [y/M] \mid object)}{P(f_k(x, y)[x/M], [y/M] \mid non\text{-}object)} \quad (13)$$

Figure 13 shows $C(x, y)$ as a color overlay on a continuum from red to green, with green indicating positive values and red indicating negative values.

It is interesting to notice that no particular region in a face seemed to be consistently more influential than the others, and the regions of particular positive influence were not sharply localized but tend to be spread out. Occluded areas usually contributed a negative influence. Also, characteristics that were uncommon in the training set, such as the mottled beard on the man in the lower right corner of Fig. 13(b), gave negative response.

We performed a similar analysis for car detection. Figure 14 shows the influence of the *parts* as a function

Figure 14. Positional decomposition of classifier response to particular cars. Green areas are "car-like" and red areas are not "car-like."

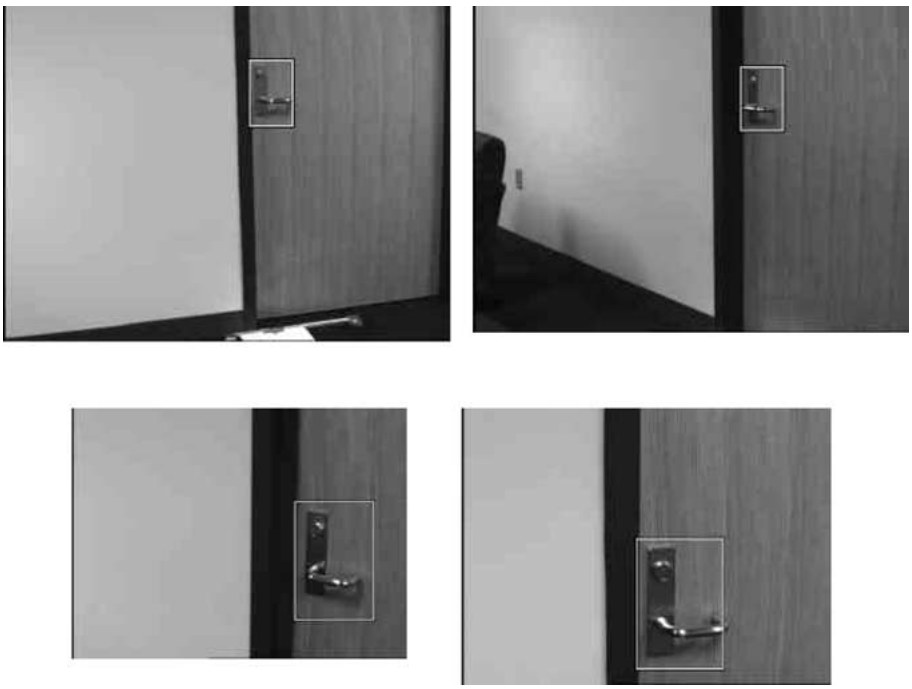


Figure 15. Doorknob detection results.

of position on the car. The areas of positive and negative response seem to vary somewhat from example to example, but the window posts, the grille, tires, and silhouette often gave positive response, and the background and reflections of the surrounding scenery on the shiny surfaces often gave negative responses.

8. Conclusion

We have described an algorithm for object detection using a set of viewpoint specific classifiers each of which is trainable using a large set of sample images. Each classifier forms a log likelihood ratio as the product of the log likelihoods of a large set of *parts*. Each *part* represents various local properties in space, frequency, and orientation.

This algorithm is generic and easily adaptable to new objects with little re-programming. We demonstrated its use for detecting faces and passenger cars. The same algorithm was also trained to detect doorknobs for indoor robot navigation (Fig. 15).

Our goal is to develop a system that detects and recognizes of many kinds of objects in photographs and video including everyday office objects, text captions in video, and various structures in biomedical imagery.

Appendix: Wavelet Transform

A wavelet transform is computed by passing an image through a cascade of filter bank stages. Figure 16 shows one such stage. This stage filters the image in the vertical direction using the filter pair given by $c(y)$ and $d(y)$. $c(y)$ and $d(y)$ are finite extent filters where $c(y)$ is low-pass and $d(y)$ is high-pass. These filter outputs are down-sampled by a factor of 2 in the vertical direction. The resulting outputs are filtered in the horizontal direction by an identical pairs of filters oriented horizontally, $c(x)$ and $d(x)$, and then down-sampled by a factor of 2 horizontally. The result is a decomposition of the image into 4 subbands denoted by LL, LH, HL, HH. Each of these represents information from a different orientation. LH represents vertical information (low-pass filtering in horizontal direction, high-pass filtering in vertical direction), HL represents horizontal information (low-pass filtering in vertical direction, high-pass filtering in horizontal direction), HH represents diagonal information (high-pass filtering in both directions), and LL (low-pass filtering in both directions) represents the original image at a lower resolution. Figure 17 shows the common representation for these subbands in an image form. These four subbands could be thought of as one frequency band or one level in a wavelet

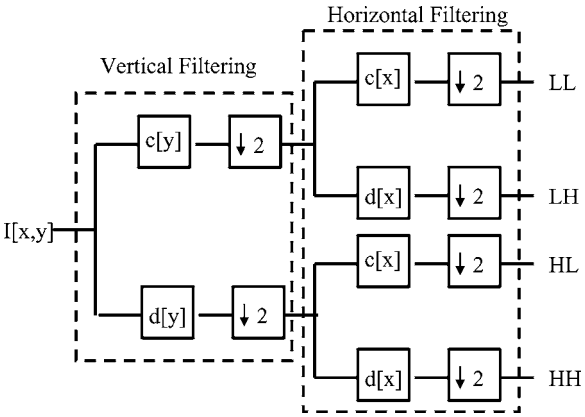


Figure 16. One stage in a filter-bank wavelet decomposition.

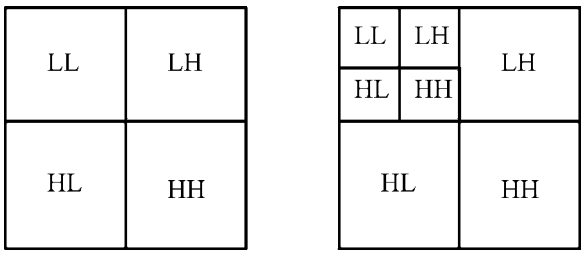


Figure 17. Representation of a one-level and two level wavelet transform.

transform. To expand the decomposition in frequency, we can iterate on the LL band; that is, we decompose the LL band as we decomposed the original image by passing it through another stage identical to the first. If the filter pair is chosen properly, the original image can be reconstructed from its transform with no loss of information. Such filter banks are called perfect reconstruction filter banks. Several books describe their design (Strang and Nguyen, 1997; Vetterli and Kovacevic, 1995).

Notes

1. These relationships (intra-subband, inter-subband, inter-frequency) were initially defined in Cosman et al. (1996).
2. John Krumm and Henry Rowley each provided image collections.
3. http://www.wuarchive.wustl.edu.
4. Introduced by Sung and Poggio (1998) for image classification and also used by Rowley et al. (1998).
5. See Shapire and Singer (1999) for a complete description of the algorithm.
6. Provided by Jonathon Phillips.
7. See http://www.nist.gov/srd/nistsd18.htm.
8. Provided by Woodward Yang.
9. Collected by Henry Rowley, Shumeet Baluja, Henry Schneiderman, and Takeo Kanade.

References

Amit, Y. 2000. A neural network architecture for visual selection. *Neural Computation*, 12:1059–1089.

Arun, K.S., Huang, T.S., and Blostein, S.D. 1987. Least-Squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, (9):698–700.

Burl, M.C. and Perona, P. 1996. Recognition of planar object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 223–230.

Burl, M.C., Weber, M., and Perona, P. 1998. A probabilistic approach to object recognition using local photometry and global geometry. In *Proc. of the 5th European Conf. on Computer Vision*.

Chow, C.K. and Liu, C.N. 1966. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3).

Cortes, C. and Vapnik, V. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

Cosman, P.C., Gray, R.M., and Vetterli, M. 1996. Vector quantization of image subbands: A survey. *IEEE Transactions on Image Processing*, 5(2):202–225.

Domingos, P. and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130.

Field, D.J. 1999. Wavelets, vision and the statistics of natural scenes. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 357(1760):2527–2542.

Freund, Y. and Shapire, R.E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Geman, D. and Flueret, F. 2001. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41:85–107.

Gori, M. and Scarselli, F. 1998. Are mulilayer perceptrons adequate for pattern recognition and verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1121–1132.

Kung, Y. 1993. *Digital Neural Networks*. Prentice-Hall.

Lades, M., Vorbruggen, J.C., Buhmann, J., Lange, J., Malsburg, C.v.d., Wurtz, R.P., and Konen, W. 1993. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311.

Lewis II, P.M. 1959. Approximating probability distributions to reduce storage requirements. *Information and Control*, 2:214–225.

Osuna, E., Freund, R., and Girosi, F. 1997. Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–136.

Romdhani, S., Torr, P., Scholkopf, B., and Blake, A. 2001. Computationally efficient face detection. In *International Conference on Computer Vision*, pp. 695–700.

Roth, D., Yang, M.-H., and Ahuja, N. 1999. A SNoW-based face detector. *NPPS-12*.

Rowley, H.A., Baluja, S., and Kanade, T. 1998. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38.

Rowley, H. 1999. Neural network-based face detection. Ph.D thesis. CMU-CS-99-117.

Schiele, B. and Crowley, J.L. 1996. Probabilistic object recognition using multidimensional receptive field histograms. In *International Conference on Pattern Recognition*.

Schiele, B. and Crowley, J.L. 2000. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50.

Schneiderman, H. and Kanade, T. 1998. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Shapire, R.E. and Singer, Y. 1999. Improving boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Strang, G. and Nguyen, T. 1997. *Wavelets and Filter Banks*. Wellesley, Cambridge Press: Wellesley, MA.

Sung, K.-K. and Poggio, T. 1998. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51.

Swain, M.J. and Ballard, D.H. 1991. Color indexing. *International Journal of Computer Vision*, 7(1):11–32.

Vetterli, M. and Kovacevic, J. 1995. *Wavelets and Subband Coding*. Prentice-Hall.

Viola, P. and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wiskott, L., Fellous, J.-M., Kruger, N., Malsburg, C.v.d. 1997. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779.