An Algorithm for the Detection of Faces on the Basis of Gabor Features and Information Maximization

Hitoshi Imaoka
h-imaoka@cb.jp.nec.com
Multimedia Research Laboratories, NEC Corporation, Miyamaeku, Kawasaki, Kanagawa, 216-8555 Japan

Kenji Okajima
okajima@frl.cl.nec.co.jp
Fundamental Research Laboratories, NEC Corporation, Tsukuba, 305-8501 Japan

We propose an algorithm for the detection of facial regions within input images. The characteristics of this algorithm are (1) a vast number of Gabor-type features (196,800) in various orientations, and with various frequencies and central positions, which are used as feature candidates in representing the patterns of an image, and (2) an information maximization principle, which is used to select several hundred features that are suitable for the detection of faces from among these candidates. Using only the selected features in face detection leads to reduced computational cost and is also expected to reduce generalization error. We applied the system, after training, to 42 input images with complex backgrounds (Test Set A from the Carnegie Mellon University face data set). The result was a high detection rate of 87.0%, with only six false detections. We compared the result with other published face detection algorithms.

1 Introduction

In this letter, we propose a feature-based algorithm for the detection of faces in frontal view within images. The algorithm operates by extracting sub-windows from the input image. It then applies Gabor filters to extract edge features from these subwindows and uses a modified decision tree classifier to decide whether each subwindow contains a face. The classifier is generated through a learning process based on the information-maximization principle (Linsker, 1989). The algorithm is applicable to the detection of other objects because it is not customized for faces.

Many algorithms for face detection have been proposed (Rowley, Baluja, & Kanade, 1995, 1998; Feriaud & Bernier, 1997; Feriaud, Bernier, Viallet, & Collobert, 2001; Sung & Poggio, 1998; Osuna, Freund, & Girosi, 1997; Terrillon, Shirazi, Sadek, Fukamachi, & Akamatsu, 2000; Schneiderman & Kanade, 1998, 2000; Wiskott, Fellous, Krüger, & von der Malsburg, 1997; Huang &
Mariani, 2000; Hung, Gutta, & Wechsler, 1996). This is because variations in illumination, view angle, expression, and so on make face detection difficult (Chellappa, Wilson, & Sirohey, 1995). In finding ways to detect an object with a high degree of accuracy, we have to answer two questions: How do we select effective features that are suitable for detecting the object? and How do we classify the pattern into the correct class (face or nonface) by using the selected features? In response to the second question, algorithms such as neural network-based algorithms (Rowley et al., 1995, 1998; Féraud & Bernier, 1997; Féraud et al., 2001), cluster-based algorithms (Sung & Poggio, 1998), support vector machines (Osuna et al., 1997; Terrillon et al., 2000), and the histogram method (Schneiderman & Kanade, 1998, 2000) have been applied. Attempts at answering the first question involve manual feature selection (Wiskott et al., 1997; Huang & Mariani, 2000), the use of a linear feature selection algorithm (Turk & Pentland, 1991), AdaBoost-based feature selection (Viola & Jones, 2001), and others. (See also Fasel & Movellan, 2002, for a comparison of techniques used in most successful neurally inspired face detectors.)

Our purposes in the work presented in this letter are to obtain a new method of feature selection that applies a probabilistic approach and to construct an algorithm that applies the features thus selected in face detection. Specifically, we prepare 196,800 Gabor-type features and select about 400 features from among these through a learning process based on an empirically determined informational criterion. The learning process also generates a decision tree, which we call a modified decision tree, as a classifier. In section 2, we give an outline of the face detection algorithm and in section 3 propose the modified decision tree classifier. In section 4, we present a face detection algorithm in which this classifier is applied. In section 5, we describe our evaluation of the algorithm’s performance on the CMU test images and its robustness on side views of faces. We summarize our results and provide discussion in section 6.

2 Outline of the Face Detection Algorithm

Figure 1 outlines the flow of the algorithm. The flow is almost the same as that of several other algorithms for the same purpose (Rowley et al., 1995, 1998; Féraud & Bernier, 1997; Féraud et al., 2001).

The first step is to extract the subwindows from the input image. The subwindow size is a fixed 30 by 40 pixels. Each possible subwindow of this size is extracted from the image and then tested to see whether it contains a face. To detect faces of different size, subsampling is repeatedly applied to the input image, reducing its size on each step by a factor of 1/1.2, and subwindows are extracted and tested at each size. The number of extracted subwindows is thus large. For example, from one input image with 300 by 200 pixels, nine images, each having a different size, were generated, and about 114,000 subwindows were extracted and tested.
The second step is to narrow the candidate of face subwindows by a factor of $1/100 \sim 1/1,000$; this is achieved by filtering out subwindows that are more likely to contain faces. The purpose of this step is to reduce the algorithm's overall computational cost. In this step, we use a compact and fast classifier that is designed to have a high detection rate but also has a rather high false detection rate. We call this classifier a prefilter.

The final step is to make the final decision. Of the subwindows that were passed by the prefilter, we use an accurate classifier to detect the subwindows that contain faces. The accurate classifier is designed to have a high detection rate and low false-detection rate (but with a slower processing speed than the prefilter). We call this classifier an accurate filter. A subwindow that is passed by the accurate filter is taken to be a face subwindow, and its location and size within the original image are recorded.

In the next section, we propose a classifier for use as the accurate filter. A compact version of the same classifier is used as the prefilter.

3 Face/Nonface Classifier Based on a Modified Decision Tree

We describe our classifier in this section. Sections 3.1 and 3.2 describe the learning process. In section 3.1, we show how to select suitable features for use in the detection of faces. In section 3.2, we show how the selected features are used in constructing the classifier, a modified decision tree. In section 3.3, we describe the classification rule that is used to detect faces during detection runs.

3.1 Feature Selection. First, we let $F = \{f_1, f_2, \ldots, f_n\}$ be a set of features to be extracted from a subwindow. Each feature $f_i$ takes states $\{0, 1\}$, which are determined as described below. The subwindow image, $I(\vec{x})$, is filtered through a Gabor filter, and the magnitude (i.e., the absolute value) of the complex response is calculated by

$$g_{\vec{u}_i, \vec{k}, \sigma_i} = \left| \sum_{\vec{x}} I(\vec{x}) \psi_{\vec{k}, \sigma_i}(\vec{x} - \vec{u}_i) \right| / \vec{u}_i,$$

(3.1)
where $\psi_{\vec{k},\sigma_i}(\vec{u})$ denotes a Gabor filter that is defined by

$$
\psi_{\vec{k},\sigma_i}(\vec{u}) = \exp\left(-|\vec{u}|^2/2\sigma_i^2 + i\vec{k} \cdot \vec{u}\right),
$$

with $\sigma_i$ and $\vec{k}$ denoting the width and the wave vector of the Gabor filter, respectively. The denominator $a_i$ is a normalization factor, which is introduced to make $g_{\vec{u},\vec{k},\sigma_i}$ stable against changes in the absolute brightness, $I(\vec{x}) \rightarrow aI(\vec{x})$:

$$
a_i = \sqrt{\sum_{\vec{x}} P(\vec{x}) \exp\left(-|\vec{x} - \vec{u}|^2/2\sigma_i^2\right)^2}.
$$

(3.3)

Here, we use the magnitude of the Gabor filter’s output, because this remains almost constant against changes in the positions of objects in the subwindow. We are then able to extract features that are robust against shifts in object position. According to the energy model (Adelson & Bergen, 1985; Heeger, 1992), the response of a complex cell in the visual cortex is approximated by the magnitude of a complex Gabor filter’s output (see also Fasel, Bartlett, & Movellan, 2002, for the Gabor filter for use in the facial landmarks detection). The state of a feature $f_i$ is determined by comparing $g_{\vec{u},\vec{k},\sigma_i}$ with a threshold $t_i$, that is, $f_i$ takes state 1 if $g_{\vec{u},\vec{k},\sigma_i} \geq t_i$ and $f_i$ takes state 0 otherwise. Actually, however, we introduce additive gaussian noise to the filter’s output $g_{\vec{u},\vec{k},\sigma_i}$ and use the following equation to compute the probability that the feature $f_i$ takes the state 1:

$$
p(f_i = 1) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \int_{t_i}^{\infty} \exp\left(-\left(g - g_{\vec{u},\vec{k},\sigma_i}\right)^2/2\sigma_i^2\right) dg.
$$

(3.4)

The probability $p(f_i = 0)$ is determined by $p(f_i = 0) = 1 - p(f_i = 1)$. In equation 3.4, $t_i$ and $\sigma_i^2$ denote the threshold and noise variance, respectively, and these are determined depending on the feature. A procedure for determining the threshold and noise variance is described in appendix A. We introduce the noise so that those features whose outputs $g_{\vec{u},\vec{k},\sigma_i}$ take values above or below the threshold by large margins are selected during the learning process. This improves the detection ability for new examples that were not used in training (see appendix A).

For each subwindow, we prepared a large number of feature candidates, each having a different set of the parameters $(\vec{u}_i, \vec{k}_i, \sigma_i)$ ($i = 1, 2, \ldots, 196,800$) from among the values listed in Table 1. The central positions $\vec{u}_i$ include all possible positions in the subwindows ($30 \times 40 = 1200$). For each central position, we prepare 14 sets of $(\sigma, |\vec{k}|)$ (among them, $|\vec{k}| = 0$ in four sets) and 16 angles as orientations of $\vec{k}$. From Table 1, we see that the wavelengths $\lambda = 2\pi/|\vec{k}|$ of the filters (whose $|\vec{k}| \neq 0$) are $2\sqrt{2}, 4\sqrt{2},$ and 8 pixels. The average distance between the centers of eyes is 18.2 pixels in our $30 \times 40$ image plane.
Table 1: Parameter Values of $\vec{u}$, $\sigma$, $\vec{k}$ Used in the Gabor Filter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Number of Possible Parameter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{u}$</td>
<td>All possible positions in a subwindow</td>
<td>1200</td>
</tr>
<tr>
<td>$(\sigma,</td>
<td>\vec{k}</td>
<td>)$</td>
</tr>
<tr>
<td>$\sigma$ (pixels)</td>
<td>$(2\sqrt{2}/3, \sqrt{2}\pi/2), (4/3, \pi/2), (4/3, \sqrt{2}\pi/2),$</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>\vec{k}</td>
<td>$ (1/pixels)</td>
</tr>
<tr>
<td>Angle of $\vec{k}$ (radians)</td>
<td>$0, \pi/16, \ldots, 15\pi/16$</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 2: Distance between the centers of eyes.

(see Figure 2). The total number of feature candidates per subwindow was thus 196,800 ($= 1200 \times 10 \times 16 + 1200 \times 4$). We use $F = \{f_1, f_2, \ldots, f_{196,800}\}$ to denote these feature candidates.

Next, we apply a learning process to select suitable features from among the above features. Suppose we select $m$ features $F^* = (f_1^*, f_2^*, \ldots, f_m^*)$ from the feature candidates $F$. We introduce a variable $\nu$ as a label for each subwindow (we set $\nu = 1$ for a face subwindow and $\nu = 0$ for a nonface subwindow). Our purpose is to find features that are applicable to classifying the subwindows as a face subwindow or a nonface subwindow with minimal error. A suitable objective function for this purpose is the mutual information between $\nu$ and $F^* = (f_1^*, f_2^*, \ldots, f_m^*)$, that is, $I(\nu; F^*)$. A procedure for calculating an empirically based value for mutual information is described in appendix B. We select an optimal set of features $\tilde{F}^* = (\tilde{f}_1^*, \tilde{f}_2^*, \ldots, \tilde{f}_m^*)$ by maximizing the mutual information:

$$\tilde{F}^* = \arg \max_{F^*} I(\nu; F^*).$$

(3.5)
However, $nC_m$ calculations of mutual information are necessary in selecting features from among all possible $F^*$ according to equation 3.5. If $n$ or $m$ is large, the computational cost of finding features $\tilde{F}^*$ becomes prohibitive. We avoid this difficulty by using a simplified sequential maximization algorithm instead of the exact maximization algorithm, 3.5. We start by applying the chain rule to write the mutual information as

$$I(\nu; F^*) = I(\nu; f_1^*) + I(\nu; f_2^* | f_1^*) + \cdots + I(\nu; f_m^* | f_1^*, f_2^*, \ldots, f_{m-1}^*)$$

(3.6)

and then determine optimal features by sequentially maximizing each of the terms in the right-hand side of this equation, that is,

$$\tilde{f}_1^* = \arg \max_{f_i \in F} I(\nu; f_1),$$

$$\tilde{f}_2^* = \arg \max_{f_i \in F} I(\nu; f_1 | \tilde{f}_1^*),$$

$$\vdots$$

$$\tilde{f}_m^* = \arg \max_{f_i \in F} I(\nu; f_1 | \tilde{f}_1^*, \tilde{f}_2^*, \ldots, \tilde{f}_{m-1}^*).$$

(3.7)

It should be noted that $\tilde{F}^*$ obtained through equation 3.7 is in general different from the true solution. However, equation 3.7 requires only that we evaluate $m \times n$ values of mutual information. Hence, the computational cost is markedly lower than with equation 3.5. In equation 3.7, the mutual information $I(\nu; f_1 | \tilde{f}_1^*)$, for example, is defined as $I(\nu; f_1 | \tilde{f}_1^*) = H(\nu | \tilde{f}_1^*) - H(\nu | f_1, \tilde{f}_1^*)$. The conditional entropies, $H(\nu | \tilde{f}_1^*)$ and $H(\nu | f_1, \tilde{f}_1^*)$, can be empirically evaluated in a similar way as that described in appendix B. We stop selecting new features when the maximal information gain falls below a threshold, that is, when $\max_{f_i \in F} I(\nu; f_1 | \tilde{f}_1^*, \tilde{f}_2^*, \ldots, \tilde{f}_m^*) < \text{threshold}$.

3.2 Generation of Modified Decision Tree. In selecting the optimal features according to equation 3.7, a decision tree is produced automatically. In the top layer, the first feature $\tilde{f}_1^*$ is used as the test to classify the subwindows, and two nodes $\{0, 1\}$ are generated (e.g., a subwindow for which $\tilde{f}_1^* = 1$ is assigned to node 1). Similarly, the feature $\tilde{f}_2^*$ is used as the test in the second layer, and four nodes $(\tilde{f}_1^*, \tilde{f}_2^*) = (0, 0), (0, 1), (1, 0), (1, 1)$ are generated. In this way, a decision tree with $m$ layers is generated (see Figure 3), and this is used to classify unknown subwindows during detection runs.

The following problems remain even when we follow the procedure based on equation 3.7: when the number of selected features $m$ is large (several hundreds), the number of nodes explodes (e.g., there are $2^m$ nodes
An Algorithm for the Detection of Faces

Figure 3: Structure of the decision tree.

in the \( n \)th layer), and too few training subwindows are assigned to each of the nodes when the number of nodes is large. For this reason, we merge some of the nodes. In creating the rule for this process of merging, we apply an important property of the mutual information: if any two nodes \( D_1 \) and \( D_2 \) satisfy a relation \( p(\nu \mid D_1) = p(\nu \mid D_2) \) for \( \forall \nu \), the mutual information has the same value after merging of the nodes \( D_1 \) and \( D_2 \) as before the merging. Here, \( p(\nu \mid D) \) denotes the probability that a subwindow belongs to a class \( \nu \) when it belongs to the node \( D \). The proof of the node-merging property is given in appendix C. According to this property, we introduce a variable \( q_i = p(\nu = 1 \mid D_i)/p(\nu = 0 \mid D_i) \). If \( q_1 \) and \( q_2 \) satisfy \( q_1 = q_2 \), the relation \( p(\nu \mid D_1) = p(\nu \mid D_2) \) is satisfied for \( \forall \nu \). Hence, we use the variable \( q_i \) in the criterion for merging; that is, when \( q_i \approx q_j \), we merge nodes \( D_i \) and \( D_j \).

The details of the merging procedure are described below and illustrated in Figure 4. According to the procedure described below, the number of nodes in each layer always remains constant (= 2l).

1. A first feature \( \tilde{f}_1^* \) is selected according to equation 3.7:

\[
\tilde{f}_1^* = \arg \max_{f_i \in F} I(\nu; f_i).
\] (3.8)

2. We rewrite each node \( \{0, 1\} \), which are generated after the first test by using \( \tilde{f}_1^* \), as \( \{D_{11}, D_{12}\} \). We prepare 2l nodes \( *D_{11}, *D_{12}, \ldots, *D_{2l} \). We set \( n = 1 \).

3. All of the training subwindows that have been assigned to the node \( D_i^n \) (\( i = 1, 2 \) for \( n = 1 \) and \( i = 1, 2, \ldots, 4l \) for \( n \neq 1 \)) are transferred (merged) to node \( *D_j^n \) (\( j = 1, 2, \ldots, 2l \)) according to the following transition rule (merging rule). Let \( q_i = p(\nu = 1 \mid D_i^n)/p(\nu = 0 \mid D_i^n) \).
Figure 4: Structure of the modified decision tree. The merging process and splitting process are repeated in an alternating fashion. In each layer, the probability \( p(\nu_{\text{acc}} \mid *D) \) increases with \( j \).

The rule is then written as

\[
D_i^n \rightarrow *D_j^n = \begin{cases} 
* D_1^n & q_i < 2^{-l+1} \\
* D_2^n & 2^{-l+1} \leq q_i < 2^{-l+2} \\
\vdots & \vdots \\
* D_{2^{l-1}}^n & 2^{l-2} \leq q_i < 2^{l-1} \\
* D_{2^l}^n & 2^{l-1} \leq q_i 
\end{cases} 
\]  

(3.9)
4. The \((n+1)\)th feature is selected by

\[
\hat{\tilde{f}}_{n+1} = \arg \max_{f_i \in F} I(v; f_i | \ast D^n).
\]

(3.10)

5. We split each node \(\ast D^n_j\) according to a test by using the \((n+1)\)th feature

\[\hat{\tilde{f}}_{n+1}\] for example, a subwindow \(x \in \ast D^n_k\) for which \(\hat{\tilde{f}}_{n+1} = 0\) is assigned to \(D_{2k}^{n+1}\) and that for which \(\hat{\tilde{f}}_{n+1} = 1\) is assigned to \(D_{2k+1}^{n+1}\). Thus, we obtain \(4^l\) nodes, \(D_{1}^{n+1}, D_{2}^{n+1}, \ldots, D_{2^l}^{n+1}\). We prepare nodes \(\ast D^{n+1}_1, \ast D^{n+1}_2, \ldots, \ast D^{n+1}_{2^l}\). We set \(n = n + 1\) and go to 3.

6. We continue to apply procedures 3 to 5 until the maximal information gain has fallen below a threshold: \(\max_{f_i \in F} I(v; f_i | \ast D^n) < \text{threshold}\).

Applying this rule for merging ensures that the number of nodes \(\ast D^n_j\) in each layer is always 2. That is, the number of nodes in the \(n\)th layer is decreased from \(2^n\) to \(2^l\). In each layer \(n\), the probability \(p(v = 1 | \ast D^n_j)\) increases with \(j\).

Note that we select features according to equations 3.8 and 3.10 instead of 3.7. In equation 3.10, the mutual information \(I(v; f_i | \ast D^n)\) is defined as

\[
I(v; f_i | \ast D^n) = H(v | \ast D^n) - H(v | f_i, \ast D^n),
\]

(3.11)

where the conditional entropies, \(H(v | \ast D^n)\) and \(H(v | f_i, \ast D^n)\), are respectively defined as

\[
H(v | \ast D^n) = -\sum_{v=0,1} \sum_{i=0,1} p(v, \ast D^n_i) \log p(v | \ast D^n_i),
\]

(3.12)

and

\[
H(v | f_i, \ast D^n) = -\sum_{f_i=0,1} \sum_{i=0,1} \sum_{v=0,1} p(v, f_i, \ast D^n_i) \log p(v | f_i, \ast D^n_i).\]

(3.13)

During the training, equation 3.11 is empirically evaluated by using training examples.

When the training has been completed, the selected feature set \(\tilde{F}^* = (\tilde{f}_1^*, \tilde{f}_2^*, \ldots, \tilde{f}_m^*)\) and the rule for merging (transition rule) in each layer \(D^n_i \rightarrow \ast D^n_j\) are memorized. These data are used in detecting faces during detection runs as described in the next section. We call the above type of decision tree a modified decision tree, because the number of nodes in each layer is reduced by the merging process.

3.3 Classification of Subwindows. We now write a decision rule for determining the class of a subwindow \(x\). We use the following approximation
for the a posteriori probability of the subwindow \( x \) belonging to the face class

\[
p(v = 1 \mid x) = \sum_{i = 1}^{2l} p(v = 1 \mid \star \text{D}_i) p(\star \text{D}_i \mid x),
\]

(3.14)

where \( p(\star \text{D}_i \mid x) \) denotes the probability of \( x \) being assigned to a terminal node \( \star \text{D}_i \) and \( p(v = 1 \mid \star \text{D}_i) \) denotes the probability that a subwindow, having been assigned to \( \star \text{D}_i \), contains a face. The probability \( p(v = 1 \mid \star \text{D}_i) \) is empirically evaluated by using training examples. The detailed procedure used in calculating \( p(v = 1 \mid \star \text{D}_i) \) is given in appendix D. The probability \( p(\star \text{D}_i \mid x) \) determined in the following way. First, we write

\[
p(\star \text{D}_i \mid x) = \sum_j \sum_{f_m \in [0, 1]} p(\star \text{D}_i \mid \star \text{D}_{j}^{m-1}, \tilde{f}_m) p(\star \text{D}_{j}^{m-1}, \tilde{f}_m \mid x).
\]

(3.15)

Here \( p(\star \text{D}_i \mid \star \text{D}_{j}^{m-1}, \tilde{f}_m) \) is given by the node-merging rule (transition rule) that was obtained during the learning process:

\[
p(\star \text{D}_i \mid \star \text{D}_{j}^{m-1}, \tilde{f}_m) = \begin{cases} 1 & \text{when } \text{D}_{j-1}^{m} \text{ is transferred to } \star \text{D}_i \\ 0 & \text{otherwise} \end{cases}.
\]

(3.16)

The probability \( p(\star \text{D}_i^{m-1}, \tilde{f}_m \mid x) \) is modified to obtain

\[
p(\star \text{D}_{j}^{m-1}, \tilde{f}_m \mid x) = p(\star \text{D}_{j}^{m-1} \mid x)p(\tilde{f}_m \mid \star \text{D}_{j}^{m-1}, x).
\]

(3.17)

Note that \( p(\tilde{f}_m \mid \star \text{D}_{j}^{m-1}, x) \) is determined by equation 3.4 regardless of \( \star \text{D}_{j}^{m-1} \). Thus, equation 3.17 is written as

\[
p(\star \text{D}_{j}^{m-1}, \tilde{f}_m \mid x) = p(\star \text{D}_{j}^{m-1} \mid x)p(\tilde{f}_m \mid x).
\]

(3.18)

Substituting equation 3.18 for equation 3.15, we obtain

\[
p(\star \text{D}_i^{m} \mid x) = \sum_j \sum_{f_m \in [0, 1]} p(\star \text{D}_i^{m} \mid \star \text{D}_{j}^{m-1}, \tilde{f}_m)p(\star \text{D}_{j}^{m-1} \mid x)p(\tilde{f}_m \mid \star \text{D}_{j}^{m-1} \mid x).
\]

(3.19)

Hence, we can iteratively compute values for \( p(\star \text{D}_i^{m} \mid x) \) from the probability \( p(\star \text{D}_{i}^{m-1} \mid x) \). The probability \( p(\star \text{D}_{i}^{1} \mid x) \) is given as

\[
p(\star \text{D}_{i}^{1} \mid x) = \sum_{f_1 \in [0, 1]} p(\star \text{D}_{i}^{1} \mid \tilde{f}_1)p(\tilde{f}_1 \mid x),
\]

(3.20)

where \( p(\star \text{D}_{i}^{1} \mid \tilde{f}_1) \) is given by the node-merging rule. Therefore, the a posteriori probability \( p(v = 1 \mid x) \) is calculated by using the probabilities \( p(\tilde{f}_1 \mid x) \) and the transition probabilities of the modified decision tree,
An Algorithm for the Detection of Faces

\[ p(D_m^m | \hat{f}^m) \]. The decision rule is then given by

\[ p(v = 1 | x) \geq T \Rightarrow x \text{ is a face subwindow} \]
\[ p(v = 1 | x) < T \Rightarrow x \text{ is a nonface subwindow} \]

(3.21)

where \( T \) is a threshold.

4 Using the Modified Decision Tree Classifier in a Face Detection Algorithm

We now describe the face detection algorithm that operates on the basis of the decision tree proposed in the previous section; that is, we describe the details of the procedure applied to detect faces.

4.1 Preprocessing. We extract a large number of subwindows from an input image in order to detect faces of various sizes and at various locations. However, the extracted subwindows include variations in overall and local values with illumination. To reduce the effects of such variation, we take the convolution with the difference-of-gaussian (DoG) filter,

\[ I'(\vec{x}) = \sum_{\vec{x}'} D(\vec{x}') I(\vec{x}' - \vec{x}) \]

(4.1)

where

\[ D(\vec{x}) = (1/2\pi \sigma_e^2) \exp(|\vec{x}|^2/2\sigma_e^2) - (1/2\pi \sigma_i^2) \exp(|\vec{x}|^2/2\sigma_i^2). \]

(4.2)

In equation 4.1, \( I(\vec{x}) \) and \( I'(\vec{x}) \) denote subwindow images before and after filtering, respectively, and \( \sigma_e \) and \( \sigma_i \) in equation 4.2 are constants that satisfy \( \sigma_e < \sigma_i \). We set \( \sigma_e^2 = 5/3 \text{(pixels)} \) and \( \sigma_i^2 = 12/3 \text{(pixels)} \). This process of filtering suppresses low-frequency components in the subwindows.

This preprocessing is optional, but we found that it improves the detection performance especially for faces that have high brightness and low contrast. During tests that will be described in section 5, we incorporated the preprocessing. Presumably, other preprocessing, such as mean and variance normalization or histogram equalization, would also improve the detection performance for such low-contrast faces.

We can efficiently compute equation 4.1 by using the Fourier transform technique, that is, the DoG filter outputs, equation 4.1, for subwindows at various locations can be simultaneously computed by one whole image Fourier transform, a filtering in the frequency domain, and an inverse Fourier transform. Since we compute the Fourier transform anyway (see section 4.2), the main additional computation needed is an inverse Fourier transform, which requires us to perform \( \sim N \log_2 N \) multiplications (\( N \): number of pixels), much fewer than required by other possible preprocessing. For example, mean and variance normalization for each subwin-
dow requires us to perform $\sim N \times 1200$ multiplications in total, that is, $\sim 75 \times N \log_2 N$ multiplications when $N = 200 \times 300$ pixels.

**4.2 Prefiltering.** In constructing the prefilter, we prepared 2400 feature candidates. All possible positions in the subwindow are used as central positions ($30 \times 40 = 1200$). The orientation of the Gabor filter is determined as vertical or horizontal (i.e., angle of $\vec{k} = 0, \pi/2$). The size and spatial frequency are fixed at $(\sigma, |\vec{k}|) = (4\sqrt{2}/3, \sqrt{2}\pi/2)$. There are $2l = 52$ intermediate nodes in the application of equation 3.9.

We used only two combinations of $(\sigma, \vec{k})$ because we wanted to calculate the Gabor outputs quickly. Remember that the prefilter has to test all the subwindows. Thus, we need to calculate the Gabor outputs at (almost) every pixel within a whole input image. These can be efficiently computed via Fourier transform, that is, by a Fourier transform of the whole image, a filtering in the frequency domain, and an inverse Fourier transform. We need to carry out such calculations only for two fixed combinations of the filter, $(\sigma, \vec{k})$. The accurate filter computes the Gabor outputs through product-sum operations, since it needs to test only subwindows that are passed by the prefilter.

In training the prefilter, we used 200 face subwindows and 1000 nonface subwindows; the training data were gathered from the World Wide Web. The face images are of various sizes and are against various backgrounds. The positions of the eyes in these images were labeled manually. These positions are used in extracting face subwindows and normalizing their sizes:

1. Let the positions of the right and left eyes be $\vec{z}_r$ and $\vec{z}_l$. We use $\vec{z}_c = (\vec{z}_r + \vec{z}_l)/2$ and $d = |\vec{z}_r - \vec{z}_l|$ to compute the central position and distance between the eyes, respectively.

2. We extract the region with left-hand, right-hand, upper, and lower bounds at $0.65d$, $0.65d$, $1.2d$, and $3.2d$ from $\vec{z}_c$.

3. Resize the region thus extracted to 30 by 40 pixels.

The nonface subwindows are gathered from images of natural objects of various sizes and in various positions. One hundred features were selected from among 2400 feature candidates. The prefilter obtained through the above procedure showed a high detection rate (above 90%) along with a rather high rate of false detections (about 1%).

**4.3 Accurate Filtering.** We construct a modified decision tree classifier according to the method described in section 3. There are 196,800 feature candidates for a subwindow, and the number of intermediate nodes $2l$ in the application of equation 3.9 is 52. We used 1500 examples of faces gathered from the World Wide Web. The nonface subwindows are extracted
An Algorithm for the Detection of Faces

from images of natural scenes that are also gathered from the Web. From these images, the nonface subwindows are gathered in a bootstrap manner (Rowley et al., 1998) according to the following procedure:

1. Prepare 200 nonface subwindows by randomly extracting them from the images.
2. Construct a modified decision tree classifier, and test new nonface subwindows that are randomly extracted from the images. Choose 100 nonface subwindows that have been misclassified as face during the test. Add them to the training set of nonface subwindows.
3. Reconstruct the modified decision tree with the enlarged set of nonface subwindows.

This process is repeated until the number of nonface subwindows reaches 3000. In this bootstrap method, missed nonface subwindows are learned repeatedly so that the incidence of false detections gradually decreases. We finally succeeded in reducing the rate of false detections to $\sim 10^{-6}$. In the final decision tree, the number of selected features was about 400. Figure 5 shows the first to ninth selected features, $f_1^*, f_2^*, \ldots, f_9^*$ in the final accurate filter. In Figure 5, the left-hand image shows the central position of the selected Gabor filter superimposed on an averaged face, and the right-hand image shows a density plot of the real component of the Gabor filter. Figure 6 shows a distribution of the output of each Gabor filter. In each plot, the horizontal axis represents the output value of the Gabor filter, and the vertical axis represents the number of subwindows for each bin. The white bars indicate the histogram for face subwindows, and the black bars indicate that for nonface subwindows. The thin line represents the threshold $t_i$, and the dotted lines represent the value of $t_i \pm \sigma_n$, where $\sigma_n$ is the noise variance. From Figure 5, we notice that features that seem reasonable are automatically selected through the process of learning. For example, the second feature corresponds to a horizontal edge detected at the position of an eye. Detection of this feature is expected in most of face subwindows (see Figure 6). The first feature corresponds to an edge near the nose. Since the orientation of this edge is different from the orientation of the nose, this edge is not detected in most of face subwindows (see Figure 6).

Figure 7A shows the distribution of selected spatial frequencies (A-1), the distribution of selected orientations (A-2), and the distribution of selected positions in the $30 \times 40$ image plane (A-3). We also plotted in Figure 7B the information gain at each frequency (B-1), each orientation (B-2), and each position (B-3). For example, in Figure 7B-2, the information gain for selected features, $\sum_{f_n^* \in \tilde{F}} I(v; f_n^* | f_1^*, f_2^*, \ldots, f_{n-1}^*)$, is plotted at each orientation $\theta$. Here, $\tilde{F}$ denotes the set of selected features, and $\theta_f$ denotes the orientation of a feature $f$. We notice that the information gain is large around $\theta = 0$ (vertical
Figure 5: The first to ninth selected features in the final accurate filter. The left-hand image shows the central position of the selected Gabor filter superimposed on the averaged face, and the right-hand image shows the density plot of the real component of the Gabor filter.

orientation) and around $\theta = 90$ (horizontal orientation). From Figure 7B-3, we see that the information gain is large around the eyes, the nose, and the mouth.

4.4 Heuristics for the Precise Estimation of the Positions and Sizes of Faces. The accurate filter will usually detect several face subwindows around a face. In order to remove such overlapping detections and to estimate the size and position of the face correctly, we include the following heuristics after the stage of accurate filtering:

1. Select the subwindow $x^*_1$ with the highest a posteriori probability of containing an image of a face $p(v = 1 | x)$.

2. Extract the set of subwindows $y = \{y_1, y_2, \ldots, y_n\}$, which are located within 5 pixels of $x^*_1$ and have been passed by the prefilter.

3. Average the a posteriori probabilities $p(v = 1 | y_i)$ over the above subwindows: $p(v = 1 | x^*_1) = (1/n) \sum_{i} p(v = 1 | y_i)$. If the average is above a threshold ($= 0.25$), the subwindow $x^*_1$ is determined as belonging to
Figure 6: Distributions of the outputs of the Gabor filters for the first to ninth features. In each plot, the horizontal axis represents the output value of the Gabor filter, and the vertical axis represents the number of subwindows for each bin. The white bars indicate the histogram for face subwindows, and the black bars indicate that for nonface subwindows. The thin line represents the threshold $t_i$, and the dotted lines represent the value of $t_i \pm \sigma_n$, where $\sigma_n$ is the noise variance.
Figure 7: (A) The distribution of selected spatial frequencies (A-1), selected orientations (A-2), and selected positions in a 30 × 40 image plane (A-3). (B) Information gain at each spatial frequency (B-1), at each orientation (B-2), and at each position (B-3).
An Algorithm for the Detection of Faces

The a posteriori probabilities for the subwindows that overlap with \( x_i^* \) are set to 0 in order to prevent overlapping detection.

4. Select the subwindow with the second highest a posteriori probability \( p(\nu = 1 | x) \). Steps 2 and 3 are applied to determine whether the selected subwindow contains a face.

5. Repeat processes 3 and 4 for all subwindows.

Through these procedures, we eliminate overlapping detection and determine the precise sizes and positions of faces.

4.5 Combination of Two Decision Trees. We combine two decision trees to reduce further the rate of false detections. The first decision tree is that generated in the above manner. The second is generated by changing the examples of nonface subwindows. These trees are combined by (1) using two filters separately to detect regions that contain faces and (2) in cases where almost the same region has been detected as containing a face by both filters, the region is considered to contain a face. In step 2 in particular, the following ratio is used as a criterion in judgment: \( (\text{area of overlap between } S_1 \text{ and } S_2) / (\text{the averaged area of } S_1 \text{ and } S_2) \), where \( S_1 \) and \( S_2 \) are the areas of the facial regions detected by the first and second filters, respectively. If the ratio is above a threshold (in this trial, we used 0.8), the region is selected as one that contains an image of a face. The region \( S_i (i = 1 \text{ or } 2) \) that has the larger a posteriori probability is then determined as the correct region.

5 Experimental Results

To verify the ability of the modified decision tree face detector, we investigated its results on 42 gray-scale images that have complex backgrounds, which were collected by the computer vision group at Carnegie Mellon University (CMU) (Rowley et al., 1995, 1998). The set is called CMU test set A and contains 169 faces. In this investigation, 4,836,713 subwindows were tested. To search for relatively small faces (smaller than that for 30 x 40 subwindow), the original images were enlarged by a factor of 1.5. In Figure 8, we give some examples of our system's results in detection. The results in the image at the upper-left corner show one false detection and two missed faces. Table 2 shows the levels of performance of our single and double decision tree methods, along with some other algorithms of state-of-the-art performance (Rowley et al., 1995; Féraud et al., 2001). The result for the double decision tree is the high detection rate of 87.0%, with six false detections. The structure of our system used for the test is summarized in Table 3.

Table 4 lists the detection rate of our double decision tree system evaluated by using another data set, (CMU+MIT) test set (A+B+C), or CMU-130 (Rowley et al., 1998; Sung & Poggio, 1998). This set consists of 130 images with 507 frontal faces. The CMU test set A is a subset of this test set.
Figure 8: Examples of detection results.

Table 2: Results on CMU Test Set A.

<table>
<thead>
<tr>
<th>Model</th>
<th>Detection Rate (%)</th>
<th>False Detections</th>
<th>False Detection Rate</th>
<th>Computational Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single decision tree</td>
<td>87.6</td>
<td>25</td>
<td>$5.17 \times 10^{-6}$</td>
<td>18.1</td>
</tr>
<tr>
<td>Double decision trees</td>
<td>87.0</td>
<td>6</td>
<td>$1.24 \times 10^{-6}$</td>
<td>25.3</td>
</tr>
<tr>
<td>Constrained generative model (Féraud et al., 2001)</td>
<td>87</td>
<td>39</td>
<td>$1.15 \times 10^{-6}$</td>
<td>–</td>
</tr>
<tr>
<td>Neural network model (Rowley et al., 1995)</td>
<td>85.2</td>
<td>47</td>
<td>$2.13 \times 10^{-6}$</td>
<td>–</td>
</tr>
</tbody>
</table>

Notes: This test set consists of 42 images and contains 169 faces. We tested 4,836,713 subwindows. The computational times are calculated for an Intel Pentium 1.0 GHz processor, operating on 200-by-300 pixel images.
Table 3: Structure of Decision Trees Used for Performance Evaluation.

<table>
<thead>
<tr>
<th></th>
<th>Number of Selected Features</th>
<th>Number of Intermediate Nodes, 2l</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single decision tree</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefilter 100</td>
<td>100</td>
<td>52</td>
</tr>
<tr>
<td>Accurate filter</td>
<td>402</td>
<td>52</td>
</tr>
<tr>
<td>Double decision tree</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefilter 100</td>
<td>100</td>
<td>52</td>
</tr>
<tr>
<td>Accurate filter 1</td>
<td>402</td>
<td>52</td>
</tr>
<tr>
<td>Accurate filter 2</td>
<td>397</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 4: Detection Rates for Various Numbers of False Detections on CMU-130 Test Set.

<table>
<thead>
<tr>
<th>Detector</th>
<th>Correct Detection Rate/False Detections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imaoka and Okajima (2001)</td>
<td>82.2%/9 86.6%/32 88.6%/65 89.3%/94 90.3%/165 91.9%/405</td>
</tr>
<tr>
<td>Viola &amp; Jones (2001)</td>
<td>78.3%/10 85.2%/31 89.8%/65 90.8%/95 91.8%/167 93.7%/422</td>
</tr>
<tr>
<td>Rowley et al. (1995, 1998)</td>
<td>83.2%/10 86.2%/23 86.0%/31 89.2%/95 90.1%/167 89.9%/422</td>
</tr>
<tr>
<td>Féraud et al. (2001)</td>
<td>86% /8</td>
</tr>
<tr>
<td>Schneiderman &amp; Kanade (2000)</td>
<td>(94.4%/65)</td>
</tr>
<tr>
<td>Yang et al. (2000)</td>
<td>(93.6% /74)</td>
</tr>
<tr>
<td>Roth et al. (2000)</td>
<td>(94.8% /78)</td>
</tr>
</tbody>
</table>

*aWork in press.*

also lists the detection rate for other published systems compiled by Hjelmas and Low (2001) and by Viola and Jones (2001). The system of Schneiderman and Kanade, that of Yang, Ahuja, and Kriegman (2000) and that of Roth, Yang, and Ahuja (2000) have high detection rates with a relatively small number of false detections. These results are on the CMU-130 test set but excluding some hand-drawn faces, so their results are for a subset of the CMU-130 containing 125 images with 483 faces. If the full test set was used, their detection rate would be lower and their number of false detections increased. The result for Féraud et al. (2001) is obtained by using a system different from that which they used for test set A. Their result for test set A (see Table 2) was obtained by a system composed of two detectors, while they used a system of four detectors to obtain the result in Table 4. Also, the results for Rowley et al. (1995, 1998) are for different versions of their detector (summarized in Table 3 of Viola and Jones, 2001). As for the processing speed, the Viola-Jones detector is fast. Their detector can process a 384 × 288 pixel image in about 0.067 second on a 700 MHz Pentium III processor (Viola & Jones, 2001). At present, the processing speed of our system (see Table 2) is about
1182 H. Imaoka and K. Okajima

Figure 9: ROC curves for our face detection system on the CMU-130 test set.

1000 times slower than their system. Reducing the processing time of our system is left for future studies. (At present, our algorithm is implemented on a MATLAB platform.) Figure 9 shows a receiver operating characteristic (ROC) curve for the double decision tree system on the CMU-130 test set.

In Figure 9, ROC curves for each single decision tree are also shown. To obtain these curves, single decision trees were tested without the heuristics described in section 4.4 (overlapping detections were eliminated).

Next, to investigate a robustness of the algorithm against variation in the angle of view, we tested images of faces from various angles. For this test, we used a data set collected at Sussex University (Howell & Buxton, 1995), which is composed of 100 images of 10 people with a 90 degree range of angle of view (10 views for each person). Table 5 shows the result of the double decision tree and of one other method (Féraud et al., 2001). Although we had used frontal views of faces as the training data, we found that our algorithm detected the faces in 40 degree side views at the rate of 80% and with no false detections. This indicates that our face detector is robust against variation in the angle of view.

6 Discussion

In this letter, we have proposed a modified decision tree classifier and applied it to the problem of detecting faces. Since our method is not specialized
Table 5: Results on the Sussex Database.

<table>
<thead>
<tr>
<th>Face View (degrees)</th>
<th>Double Decision Tree</th>
<th>Féraud et al. (2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Detection Rate (%)</td>
<td>False Detections</td>
</tr>
<tr>
<td>0</td>
<td>100.0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>100.0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>100.0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>100.0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>80.0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>60.0</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>40.0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>20.0</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>10.0</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0.0</td>
<td>0</td>
</tr>
</tbody>
</table>

Notes: The database is composed of 100 images of 10 people with a 90 degree range of views. Here, the results under Féraud et al. (2001) are for a detector that was trained by using frontal views of faces.

for the detection of faces, we expect it to be applicable to other object detection problems.

The operation of the modified decision tree resembles the ID3 method proposed by Quinlan (1979). However, the merging of nodes in our method leads to markedly lower computational times than for ID3. Furthermore, our algorithm realizes improved detection ability for new examples that were not used in training by introducing noise to the feature output.

AdaBoost (for an overview, see, e.g., Schapire, 2002) can be used as a method for feature selection if a weak learner in AdaBoost is constrained to return a weak classifier that can depend on only a single feature (Viola & Jones, 2001). Both our method and the AdaBoost-based method select features in a sequential way. Our infomax-based method selects each feature by maximizing the additional (or conditional) information gain. Thus, in our method, the importance of each selected feature can be evaluated by using the information gain, and the stopping criterion is rather simple: stop selecting a new feature when the maximal additional information gain becomes (almost) zero. The AdaBoost-based method also computes a weight for each feature (for each weak classifier). However, the sum of the weights is not bounded, and thus determining the stopping criterion is not straightforward. For example, Shapire (2002) describes a case in which the test error continues to decrease with the number of boosting rounds even after the training error becomes zero.

Viola and Jones (2001) used the AdaBoost-based feature selection method in their face detection system. They reported that the first and second features selected were located around the eyes. Our experiments showed, consistent with their result, that the information gain around the eyes is large (see Figure 7). The first feature selected in their system is a two-rectangle.
feature (one rectangle for an excitatory subfield, and the other for an inhibitory subfield) whose width is larger than the distance between the eyes. This feature size is larger than the largest feature size possible in our system (3σ of the largest Gabor feature in our system was set to 8 pixels, 0.44 times the distance between the eyes). By considering their result, the detection performance of our system might be improved if we add features of larger size to the ensemble of possible features.

Appendix A: Procedure for Determining the Threshold and Noise Variance for the Gabor Filter’s Output

In this appendix, we give a procedure for determining the threshold $t_i$ and noise variance $\sigma^2_{n_i}$ for use in equations 3.4 and 3.5. Let $x = \{x_1, x_2, \ldots, x_n\}$ be a training set of subwindows, each of which is assumed to belong to the face class ($\nu = 1$) or the nonface class ($\nu = 0$). A threshold for each feature, $t_i$, is determined by maximizing the empirical value for mutual information between the variable $\nu$ and each feature $f_i$ ($i = 1, 2, \ldots, 196,800$). Let $f_i(t'_i)$ be a feature with a threshold of $t'_i$ and the mutual information be $I(\nu; f_i(t'_i))$. The threshold is then determined as the $t'_i$ that produces the maximum value for mutual information:

$$t_i = \arg \max_{t'_i \in \Phi_i} I(\nu; f_i(t'_i)),$$

(A.1)

where $\Phi_i = \{g_{i}^{\nu} + (3 \sqrt{g_{i}^{\nu}/20}) \times i \mid i = -20, -19, \ldots, 19, 20\}$, and $g_{i}^{\nu}$ and $g_{i}^{\sigma}$ are the average and variance of the Gabor filter’s output over training subwindows respectively. The average and variance are given as

$$g_{i}^{\nu} = (1/n) \sum_{j=1}^{n} g_{i,j,k,\sigma},$$

(A.2)

$$g_{i}^{\sigma} = (1/n) \sum_{j=1}^{n} (g_{i,j,k,\sigma})^2 - (g_{i}^{\nu})^2,$$

(A.3)

where $g_{i,j,k,\sigma}$ is the Gabor filter’s output for the $j$th subwindow and $n$ is the number of training subwindows. (The noise variance is assumed to be zero in the calculation A.1). A procedure for calculating the mutual information is given in appendix B.

For the noise variance, we assume that it is proportional to the variance of the Gabor filter’s outputs as

$$\sigma^2_{n_i} = \alpha g_{i}^{\sigma},$$

(A.4)

where $\alpha$ is a fixed constant. We used $\alpha = 0.2$ in constructing the classifier and in testing images.

The noise is introduced to select features that can separate training examples with large margins. Suppose, for simplicity, we have two training
Figure 10: Effect of the noise on the training error (schematic representations.)

examples, \(x_{\text{face}}\) and \(x_{\text{nonface}}\), and these can be successfully classified by using a feature (filter output) \(g_1\) or by using \(g_2\) (see Figure 10a). The training error is zero for both cases, but we expect that the generalization error will be smaller for a feature that can separate these data with a larger margin. By introducing the noise, the training error will increase for both cases (see Figure 10b), but the increase will depend on the margin (presumably it also depends on the generalization error), and thus we will be able to select a better feature. In this sense, the noise in our algorithm plays a similar role as that of a complexity penalty term introduced in model selection problems.

Equation 3.4 mean that for each training example, we generate an ensemble of data according to a certain data-generation model, and we use these ensembles for the training. Suppose we generate an ensemble \(S_x\) for a training example \(x\). Then the filter output \(g_i(x')\) for \(x' \in S_x\) is determined by a probability density that is set in our model to \(p(g_i(x')) = 1/(\sqrt{2\pi} \sigma_{n,i}) \exp(-(g_i(x') - g_i(x))^2/2\sigma_{n,i}^2)\), where \(g_i(x)\) denotes the filter output for \(x\). We assumed that the parameter \(\sigma_{n,i}^2\) is proportional to the variance \(\sigma_i^2\), that is, \(\sigma_{n,i}^2 = \alpha \sigma_i^2\), and experimentally optimized the value of \(\alpha\) as \(\alpha = 0.2\). From the data-generation model viewpoint, the noise has the effect of enlarging the effective training data size. We hoped that this will reduce...
the generalization error. Actually, we found that the noise is effective in reducing the generalization error in our system.

**Appendix B: Procedure for Calculating the Empirical Mutual Information**

In this appendix, we give a procedure for calculating the empirical mutual information $I(\nu; F^*)$ between the variable $\nu$, which represents a subwindow label, and a set of features $F^* = (f_1^*, f_2^*, \ldots, f_m^*)$.

We let $x_1, x_2, \ldots, x_{l_{\text{face}}}$ denote the face subwindows used in training and $x_{l_{\text{face}}+1}, x_{l_{\text{face}}+2}, \ldots, x_{l_{\text{face}}+l_{\text{nonface}}}$ denote the nonface subwindows used in training, where $l_{\text{face}}$ and $l_{\text{nonface}} = l - l_{\text{face}}$ denote the numbers of training subwindows that belong to the respective classes. We write the probability that a feature $f$ takes state $i \in \{0, 1\}$ for a subwindow $x$ as $p_f(i \mid x)$. We now derive a procedure for calculating the mutual information $I(\nu; F^*)$ by using $p_f(i \mid x), l_{\text{face}}, l_{\text{nonface}}$, and $l$.

By definition, the mutual information is given by

$$I(\nu; F^*) = H(\nu) - H(\nu \mid F^*), \quad (B.1)$$

where the first term denotes the entropy of $\nu$ and the second term denotes the conditional entropy. The first term is given by

$$H(\nu) = -p(\nu = 1) \log p(\nu = 1) - p(\nu = 0) \log p(\nu = 0), \quad (B.2)$$

where $p(\nu = 1)$ and $p(\nu = 0)$ are the probabilities that a subwindow belongs to the face and nonface class, respectively. These are given by

$$p(\nu = 1) = l_{\text{face}}/l, \quad (B.3)$$
$$p(\nu = 0) = l_{\text{nonface}}/l. \quad (B.4)$$

The second term on the right-hand side of equation B.1 is given by

$$H(\nu \mid F^*) = - \sum_{i_1 \in f_1^*} \sum_{i_2 \in f_2^*} \cdots \sum_{i_m \in f_m^*} \sum_{\nu = 0, 1} p(\nu, i_1, i_2, \ldots, i_m) \times \log p(\nu \mid i_1, i_2, \ldots, i_m). \quad (B.5)$$

Note that the probability of a subwindow $x$ having a feature $f_j^* = 0$ or 1, that is, $p_f(i_j \mid x)$ is determined by equation 3.4. Thus, the probability is written as

$$p(i_1, i_2, \ldots, i_m \mid x) = \prod_{j=1}^m p_f(i_j \mid x). \quad (B.6)$$
Using equation B.6, the elements of equation B.5 are rewritten as

\[ p(v, i_1, i_2, \ldots, i_m) = \sum_x p(i_1, i_2, \ldots, i_m | x)p(x, v) \]

\[ = (1/l) \sum_{x \in V} \prod_{j=1}^{m} p_{j}^{*}(i_j | x). \]  \hspace{1cm} (B.7)

\[ p(v | i_1, i_2, \ldots, i_m) = p(v, i_1, i_2, \ldots, i_m)/p(i_1, i_2, \ldots, i_m) \]

\[ = \sum_{x \in V} \prod_{j=1}^{m} p_{j}^{*}(i_j | x)/\sum_{x} \prod_{j=1}^{m} p_{j}^{*}(i_j | x). \]  \hspace{1cm} (B.8)

We then use equations B.2, B.5, B.7, and B.8 in empirically obtaining the mutual information \( I(\nu; F^*) \) by using \( p_f(i | x), l_{\text{face}}, l_{\text{nonface}}, \) and \( l \).

### Appendix C: Proof of the Invariance of Mutual Information Through Node Merging

We consider a case where two nodes \( D_1 \) and \( D_2 \) are merged into a single node \( D \). We show that if \( p(\nu | D_1) \) is equal to \( p(\nu | D_2) \) for \( \forall \nu (= 0 \text{ or } 1) \), the mutual information before and after merging is invariant: \( I(\nu; D_{\text{before}}) = I(\nu; D_{\text{after}}) \).

Here, the variables \( D_{\text{before}} \) and \( D_{\text{after}} \) are, respectively, defined as

\[ D_{\text{before}}(x) = D_i \quad \text{when the subwindow } x \text{ belongs to } D_i, \text{ and} \]

\[ D_{\text{after}}(x) = \begin{cases} 
D_i & \text{when } x \in D_i, i \neq 1, 2 \\
D & \text{when } x \in D_1 \text{ or } x \in D_2.
\end{cases} \]

First, we write \( I(\nu; D_{\text{before}}) \) as

\[ I(\nu; D_{\text{before}}) = H(\nu) - H(\nu | D_{\text{before}}). \]  \hspace{1cm} (C.1)

The second term on the right-hand side of equation C.1 is written as

\[ H(\nu | D_{\text{before}}) = -\sum_{i=1,2} p(D_i) \sum_{\nu=0,1} p(\nu | D_i) \log p(\nu | D_i) - \sum_{i \neq 1,2} p(D_i) \sum_{\nu=0,1} p(\nu | D_i) \log p(\nu | D_i) \]  \hspace{1cm} (C.2)

Our hypothesis is

\[ p(\nu | D) = p(\nu | D_1) = p(\nu | D_2). \]  \hspace{1cm} (C.3)

The following equation holds when nodes are merged:

\[ p(D) = p(D_1) + p(D_2). \]  \hspace{1cm} (C.4)
Substituting equations C.3 and C.4 into C.2,
\[ H(v \mid D_{\text{before}}) = -p(D) \sum_{v=0,1} p(v \mid D) \log p(v \mid D) \]
\[ - \sum_{i \neq 1,2} p(D_i) \sum_{v=0,1} p(v \mid D_i) \log p(v \mid D_i) \]
\[ = H(v \mid D_{\text{after}}). \]  (C.5)

From equations C.1 and C.5, we obtain
\[ I(v; D_{\text{before}}) = I(v; D_{\text{after}}). \]

Appendix D: Calculation of \( p(v = 1 \mid *D^m) \) in Equation 3.14

We give a procedure for calculating \( p(v = 1 \mid *D^m) \) on the basis of training examples. Let \( Y = \{y_1, y_2, \ldots, y_n\} \) be a set of training examples. We determine \( p(v = 1 \mid *D^m) \) by
\[ p(v = 1 \mid *D^m) = \sum_{y \in Y} p(v = 1 \mid y)p(y \mid *D^m). \]  (D.1)

Here, \( p(v = 1 \mid y) \) is given as
\[ p(v = 1 \mid y) = \begin{cases} 1 & \text{when } y \text{ is a face subwindow} \\ 0 & \text{when } y \text{ is a nonface subwindow} \end{cases}. \]  (D.2)

On the other hand, \( p(y \mid *D^m) \) is given as
\[ p(y \mid *D^m) = p(*D^m \mid y)p(y) / \sum_{y \in Y} p(*D^m \mid y)p(y), \]  (D.3)

where \( p(y) = 1/n \) for all \( y \) and \( p(*D^m \mid y) \) is written as
\[ p(*D^m \mid y) = \sum_i \sum_{f_m \in \{0,1\}} p(*D^m \mid *D^{m-1}_j, \tilde{f}_m)p(*D^{m-1}_j, \tilde{f}_m \mid y). \]  (D.4)

Here, \( p(*D^m \mid *D^{m-1}_j, \tilde{f}_m) \) is obtained by applying the node-merging rule:
\[ p(*D^m \mid *D^{m-1}_j, \tilde{f}_m) = \begin{cases} 1 & \text{when } D^{m-1}_{j-1} + f_m \text{ is transferred to } *D^m_j \\ 0 & \text{otherwise} \end{cases}. \]  (D.5)

Note that a subwindow \( x \in *D^{m-1}_j \) for which \( \tilde{f}_m(x) = 0(1) \) is assigned to the node \( D^m_{2j-1}(D^m_{2j}) \) (see section 3.2). The probability \( p(*D^{m-1}_j, \tilde{f}_m(y)) \) is modified to obtain
\[ p(*D^{m-1}_j, \tilde{f}_m \mid y) = p(*D^{m-1}_j \mid y)p(\tilde{f}_m \mid *D^{m-1}_j, y). \]  (D.6)
An Algorithm for the Detection of Faces

Note that \( p(\tilde{f}_m \mid \ast D_{j-1}^m, y) \) is determined by equation 3.4, regardless of \( \ast D_{j-1}^m \). Thus, equation D.6 is written as

\[
p(\ast D_{j-1}^m, \tilde{f}_m \mid y) = p(\ast D_{j-1}^m \mid y)p(\tilde{f}_m \mid y).
\]

(D.7)

Substituting equation D.7 into equation D.4, we obtain

\[
p(\ast D_{i}^m \mid y) = \sum_{j} \sum_{f_{m}^{\prime} = \{0,1\}} p(\ast D_{i}^m \mid \ast D_{j}^{m-1}, \tilde{f}_m^{\prime})p(\tilde{f}_m^{\prime} \mid y)p(\ast D_{j}^{m-1} \mid y).
\]

(D.8)

Hence, we can iteratively compute the probability \( p(\ast D_{i}^m \mid y) \) from the probability \( p(\ast D_{i}^{m-1} \mid y) \). The probability \( p(\ast D_{i}^1 \mid y) \) is given as

\[
p(\ast D_{i}^1 \mid y) = \sum_{f_{i}^{\prime} = \{0,1\}} p(\ast D_{i}^1 \mid f_{i}^{\prime})p(f_{i}^{\prime} \mid y).
\]

(D.9)

where \( p(\ast D_{i}^1 \mid f_{i}^{\prime}) \) is given by the node-merging rule. Therefore, the probability \( p(y \mid D_{i}^{m}) \) in equation D.3 is calculated by using the probabilities \( p(f_{i}^{\prime} \mid y) \) (\( i = 1, 2, \ldots, m \)) and the transitional probabilities \( p(\ast D_{i}^{k} \mid \ast D_{j}^{k-1}, \tilde{f}_k^{\prime}) \) (\( i, j = 1, 2, \ldots, 2l, k = 1, 2, \ldots, m \)). Consequently, by substituting equations D.2 and D.3 into equation D.1, we obtain

\[
p(\nu = 1 \mid \ast D_{i}^{m}) = \frac{\sum_{y \in \text{face}} p(\ast D_{i}^{m} \mid y)}{\sum_{y \in Y} p(\ast D_{i}^{m} \mid y)}.
\]

(D.10)

where \( p(\ast D_{i}^{m} \mid y) \) is given by equations D.8 and D.9.

References


Received October 4, 2002; accepted December 2, 2003.