Real-Time Face Detection and Lip Feature Extraction Using Field-Programmable Gate Arrays

Duy Nguyen, David Halupka, *Student Member, IEEE*,
Parham Aarabi, *Member, IEEE*, and Ali Sheikholeslami, *Senior Member, IEEE*

***Abstract*—This paper proposes a new technique for face detection and lip feature extraction. A real-time field-programmable gate array (FPGA) implementation of the two proposed techniques is also presented. Face detection is based on a naive Bayes classifier that classifies an edge-extracted representation of an image. Using edge representation significantly reduces the model's size to only 5184 B, which is 2417 times smaller than a comparable statistical modeling technique, while achieving an 86.6% correct detection rate under various lighting conditions. Lip feature extraction uses the contrast around the lip contour to extract the height and width of the mouth, metrics that are useful for speech filtering. The proposed FPGA system occupies only 15 050 logic cells, or about six times less than a current comparable FPGA face detection system.**

***Index Terms*—Edge information, face detection, field-programmable gate array (FPGA), lip feature extraction, lip tracking, naive Bayes classifier, Sobel.**

I. Introduction

STATE-OF-THE-ART speech recognition systems, in a noise-free environment, can achieve word error rates as low as 5% [1]. In noisy environments, such as an office, the word error rate jumps to 58% or higher. Noise, due to reverberations or other speakers, causes mismatches in the models initially trained on clean speech. Clearly, noise suppression is needed to improve the performance of speech recognition systems in practical environments.

Motivated by the fact that humans can use lip motion to comprehend a speaker in a noisy environment, researchers have tried to use lip motion to improve the accuracy of speech recognition systems. Lip motion can be used by speech enhancement systems [3] to remove noise and enhance the speech signal of interest, or by speech recognition systems directly as extra information for recognition [2]. Several audio–visual automatic speech recognition (AV-ASR) systems have been proposed [3]–[5], [7], [25]. These systems can decrease the word error rate significantly over purely acoustic speech recognition [4]–[6].

In addition to audio information, an audio–visual speech processing system also needs to extract visual information, such as lip motion, gestures, or facial expressions, to name a few. We will restrict the focus of this paper to lip motion analysis. For lip motion analysis, the system first needs to locate the speaker's face. The process of locating faces in an image is called face detection. Once the face is found, the lips can be located and their visual features can be extracted. This stage is called lip feature extraction.

Visual information can be helpful for speech processing applications, but such information comes at increased system and computational complexity. In fact, most state-of-the-art face detection systems are computationally complex and cannot run in real time. Moreover, for lip feature extraction, we need to capture the motion of the lips in real time. That is, we need to process sets of images at a rate of 30 frames per second (fps).

Given the processing requirements and the processing capabilities of today's computers, a custom hardware implementation is necessary to be able to localize faces and to detect lip motion in real time. Moreover, a hardware implementation can be integrated directly onto the charge-coupled device (CCD) substrate, which is the heart of digital image acquisition devices. As such, the focus of this paper is on hardware-based lip motion analysis. Specifically, the hardware optimized algorithms proposed for face detection and lip feature extraction, and the hardware implementations thereof.

II. Background and Prior Work

A. Face Detection

Face detection by itself is useful not only for AV-ASR systems but also for such varied applications such as facial expression recognition, teleconferencing, security, and robotics. Indeed, humans can find and identify a face in an image effortlessly. Extenuating circumstances, such as poor lighting conditions, partially obstructed faces, or side profiles of faces, do not seem to hinder our ability to find faces in an image. For machines, the task of face detection is very difficult, since faces exhibit facial feature variations from person to person and for a single person over time. Lighting conditions and the presence or absence of facial hair also compound the detection task.

The face detection techniques proposed in the literature can be divided into two broad categories: facial feature based or machine learning based techniques. Facial feature based techniques use prior knowledge about the face's features, such as its shape or skin color [10]–[14]. These systems are often simple, but do not work well in practice. On the other hand, machine-learned techniques require no prior, or relatively minimal, knowledge of what constitutes a face. Indeed, the system itself learns to detect faces. This is accomplished

Manuscript received February 3, 2005; revised August 12, 2005. This paper was recommended by Associate Editor S. Sarkar.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 2E4, Canada (e-mail: halupka@eecg.toronto.edu).

Digital Object Identifier 10.1109/TSMCB.2005.862728

by training the system with a large database of preclassified face samples and nonface samples. Several successful learning systems have been reported, which use support vector machines [15], [16], neural networks [17]–[19], and statistical modeling [20]–[22].

Machine learning techniques, in particular, can achieve high detection accuracies since they can adaptively change their understanding of what constitutes a face during training (learning). For example, the system in [20] achieves a detection rate of over 90%. However, most machine learning techniques are complex and computationally expensive. That is, real-time face detection on a video stream is infeasible. For example, the neural network based system reported in [18] takes 140 s to process a 320 $\times$ 240 image on a 175-MHz R10000 SGI O2 workstation, equivalent to more than 20 s on a 3-GHz Pentium IV. The statistical modeling approach in [21] takes 90 s on a 420-MHz Pentium II, equivalent to more than 13 s on a 3-GHz Pentium IV.

For a 30-fps video stream, face detection needs to be performed in approximately 33 ms for each frame. Clearly, a hardware implementation is necessary, where the speed-up can be achieved by parallelizing the algorithm, rather than increasing the system's clock frequency. However, most face detection techniques proposed have focused on improving face detection accuracy. As such, these implementations often do not have an efficient hardware implementation. For example, the hardware system proposed by [26]—implementing the neural network based technique proposed by [18]—takes more than 30 mm^2 of silicon area in a 160-nm technology and consumes more than 7 W of power. In Section III, we propose a hardware-efficient statistical modeling technique that uses image gradient information and a naive Bayes classifier to perform face detection.

B. Lip Feature Extraction

Lip feature extraction, or lip tracking, is complicated by the same problems that are encountered with face detection, such as variation among persons, lighting variations, etc. However, lip feature extraction tends to be more sensitive to adverse conditions. A moustache, for example, can be easily confused to be an upper lip. The teeth, tongue, and lack of a sharp contrast between the lips and face can further complicate lip feature extraction.

Recent techniques use knowledge about the lip's color or shape to identify and track the lips. Indeed, color differentiation is an effective technique for locating the lips. A study by [5] showed that, in the hue saturation value color space, the hue component provides a high degree of discrimination. Thus, the lips can be found by isolating the connected area with the same lip color. Obviously, color discriminating techniques will not work for grayscale images. Techniques that use information about the lip's shape include active contour models [23], shape models [24], and active appearance models [25]. Unfortunately, these techniques also require a large amount of storage, which is unattractive from a hardware perspective. In Section IV, we propose a lip feature extraction technique, which makes use of the contrast at the contour of the lips. This technique works well on grayscale images and can be easily implemented on hardware.

III. Face Detection

Our proposed algorithm for hardware-based face detection is described below. The basis for this algorithm is a simple naive Bayes classifier. We do not claim that this classifier is optimal in terms of hardware complexity and performance. We chose this classifier in order to reduce the number of model parameters that would need to be stored in memory. A classifier comparison would be a fruitful research endeavor, and we are currently investigating this matter.

Given an input image, an exhaustive search is performed using a window scanning technique. Each 20 $\times$ 20 search window is preprocessed and then classified as a face $(\mathcal{F})$ or nonface $(\mathcal{F}^c)$. In order to cope with faces occupying different fractions of the given image, we also classify down-sampled versions of the given image: we use a recursive scaling factor (β) of $\sqrt[4]{2}$, or 1.189 as follows:

```
**Algorithm 1** Proposed face detection algorithm
**Require:** $\mathcal{I}$, an image of a scene
  **repeat**
    **for all** $\Psi_{(20\times20)} \subseteq \mathcal{I}$ **do**
      Equalize grayscale histogram of $\Psi$
      Compute $\nabla\Psi$, magnitude, and direction of edges
      Classify $\Psi \in \{\mathcal{F}, \mathcal{F}^c\}$ based on $\nabla\Psi$
    **end for**
    Down-sample $\mathcal{I}$ by $\beta$
  **until** size of $\mathcal{I}$ is $< 20 \times 20$
  **return** position of face in $\mathcal{I}$, if exists, determined by where cluster of $\Psi \in \mathcal{F}$ exceeds a threshold
```

A. Preprocessing

In our proposed technique, images are classified based on the amplitude and direction of edges in the image (gradient of the image) because of several intrinsic advantages of an edge-based algorithm. First, an image of a face usually has discernable patterns of strong edges around the facial features, such as the mouth and eyes, whereas images of nonfacial objects usually do not have such strong edges or are in random locations. Secondly, the image gradient can be quantized more aggressively, without much loss in image information, especially if the image's contrast is maximized first. Third, using edge representation allows us to concentrate on the shape of the object instead of its appearance. This minimizes the effect of different lighting conditions.

The employed naive Bayes classifier is greatly simplified since we classify images based on the gradient of the image. However, in order for such a technique to be robust in various lighting conditions, we need to ensure that any image edge can be successfully identified. That is, we need to maximize the contrast for each 20 $\times$ 20 pixel search window. This is accomplished by using histogram equalization.

Many simple edge extraction techniques exist, such as the Sobel [27] filter or Roberts cross. These techniques are much

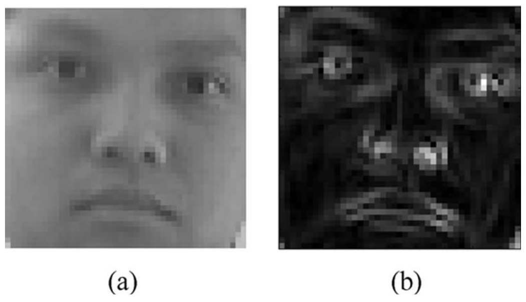

Fig. 1. A face has strong edges around facial features. These edges can be extracted using a Sobel filter [brighter pixels in (b) indicate strong edges]. (a) Original image. (b) Edge magnitudes.

simpler than principal component analysis (PCA) in terms of the calculations required. We chose the Sobel filter because of its proven performance and because of its ease of hardware implementation.

The Sobel filter uses two convolution kernels to compute the magnitude and direction of an edge at each pixel, i.e.,

$$\mathbf{S_x} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \mathbf{S_y} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}. \tag{1}$$

Each 3 × 3 piece of the 20 × 20 search window is convolved with $\mathbf{S_x}$ and $\mathbf{S_y}$ to give the magnitude of the edges in the horizontal and vertical directions, $\mathbf{E_x}$ and $\mathbf{E_y}$, respectively. Given $\mathbf{E_x}$ and $\mathbf{E_y}$, the magnitude and direction of an edge at each pixel can be computed by using

$$|e_{i,j}| = \sqrt{e^2_{\mathrm{x}_{i,j}} + e^2_{\mathrm{y}_{i,j}}} \qquad \angle e_{i,j} = \arctan\left(\frac{e_{\mathrm{y}_{i,j}}}{e_{\mathrm{x}_{i,j}}}\right) \tag{2}$$

where $\mathbf{E_x}$ and $\mathbf{E_y}$ are obtained by filtering the 20 × 20 search window using $\mathbf{S_x}$ and $\mathbf{S_y}$. That is, given $\psi_{i,j}$, 3 × 3 submatrix of Ψ centered on the (i, j)th entry

$$e_{\mathrm{x}_{i,j}} = \mathrm{Tr}\left(\mathbf{S_x}\psi^{\mathrm{T}}_{i,j}\right) \quad e_{\mathrm{y}_{i,j}} = \mathrm{Tr}\left(\mathbf{S_y}\psi^{\mathrm{T}}_{i,j}\right) \tag{3}$$

where Tr indicates the matrix trace. Edge extraction reduces the dimensions of the search window from 20 × 20 to 18 × 18.

Fig. 1 shows an example image before and after being processed by the Sobel filter. All edges with amplitudes below a certain threshold are eliminated; this threshold was chosen to be the mean edge magnitude as computed over all training images. The direction of each edge is quantized to seven directions; weak edges are denoted as a special eighth direction. We form an edge vector $\xi = [\angle e_{2,2}, \angle e_{2,3}, \ldots, \angle e_{19,19}]$ for each 20 × 20 search window, which is then classified.

B. Naive Bayes Classifier

Given the edge vector ξ, the naive Bayes classifier evaluates the likelihood that ξ is representative of a face. Let $\mathcal{F}$ indicate the occurrence of a face and $\mathcal{F}^c$ be the converse. To classify whether ξ is a face, we need to evaluate whether $P(\mathcal{F}|\xi)$ or $P(\mathcal{F}^c|\xi)$ is more likely. Clearly, $P(\mathcal{F}|\xi) > P(\mathcal{F}^c|\xi)$ for ξ to be a face. This classification rule can be rewritten as

$$\frac{P(\xi|\mathcal{F})}{P(\xi|\mathcal{F}^c)} > \lambda \tag{4}$$

where $\lambda = P(\mathcal{F}^c)/P(\mathcal{F})$. The bias (sensitivity) of the classifier can be controlled by varying λ. Without prior information, the naive choice for λ would be 1/2.

If it is naively assumed that there is no relationship between neighboring image pixels, then the edge directions are independent. Hence

$$P(\xi|\mathcal{F}) = \prod_i \prod_j P(\angle e_{i,j}|\mathcal{F}) \tag{5a}$$

$$P(\xi|\mathcal{F}^c) = \prod_i \prod_j P(\angle e_{i,j}|\mathcal{F}^c). \tag{5b}$$

In the log-likelihood domain, the classification criteria of (4), taking into account the independence of pixels, can be calculated as

$$\sum_i \sum_j \left[\log P(\angle e_{i,j}|\mathcal{F}) - \log P(\angle e_{i,j}|\mathcal{F}^c)\right] > \log \lambda. \tag{6}$$

Using a large database of preclassified images (faces and nonfaces) allows the naive Bayes classifier to train itself within the probabilistic framework described above. That is, by analyzing training data, the model estimates the values of $P(\angle e_{i,j}|\mathcal{F})$ and $P(\angle e_{i,j}|\mathcal{F}^c)$, which will be used to classify images during normal operation.

C. Training

The edge vector ξ represents 324 image edges, quantized to eight levels: seven of which represent direction and one to denote a weak edge. Each edge needs to be assigned a probability value for $P(\angle e_{i,j}|\mathcal{F})$ and $P(\angle e_{i,j}|\mathcal{F}^c)$, which are calculated, based on the training data $\{D\} = \{\mathcal{F}\} \cup \{\mathcal{F}^c\}$, as

$$P(\angle e_{i,j}|\mathcal{F}) \approx f(\angle e_{i,j} \wedge \mathcal{F}) = \sum_{\{\mathcal{F}\}} \frac{\left[\angle e_{i,j} = \angle e'_{i,j}\right]}{|\{\mathcal{F}\}|}$$

$$P(\angle e_{i,j}|\mathcal{F}^c) \approx f(\angle e_{i,j} \wedge \mathcal{F}^c) = \sum_{\{\mathcal{F}^c\}} \frac{\left[\angle e_{i,j} = \angle e'_{i,j}\right]}{|\{\mathcal{F}^c\}|}$$

where $\angle e'_{i,j}$ is the measured and quantized edge angle at position (i, j) in the training data image. The square bracket operator is a counting operator, that is,

$$[a = b] = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{otherwise.} \end{cases}$$

The training set is supplied by MIT's Center for Biological and Computation Learning [30]. This set consists of a 6977-image training set (2429 faces and 4548 nonfaces) and a 24 045-image test set (472 faces and 23 573 nonfaces). The training set we utilized consists of 2400 faces and 15 000 nonfaces all from the training set. To further enrich this set, we also used the horizontally flipped versions of these 2400 faces.

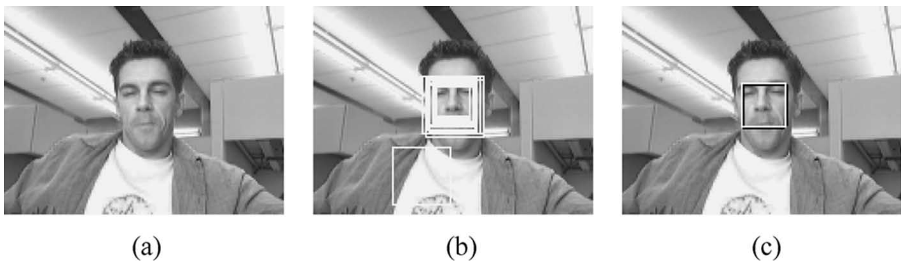

Fig. 2. Multiple face detections are used to remove false positives. High concentrations of detected faces are used to indicate the region where actual face exits. (a) Original. (b) Classifier results. (c) No false positives.

Hence, there are a total of 4800 faces and 15 000 nonfaces. These images are histogram equalized and converted into edge vectors in order to train the naive Bayes classifier.

To reduce the training time, we use bootstrapping [28]: the system is trained on a subset of the training set and then retraining it incrementally on the rest of the set. The model parameters $P(\angle e_{i,j}|\mathcal{F})$ and $P(\angle e_{i,j}|\mathcal{F}^c)$ are initially set by counting the number of times a specific edge appears at every location in the first 2000 faces and 5000 nonfaces. The system is then retrained on the entire training set as follows.

Algorithm 2 Training via bootstrapping

```
while false detection on training set $>$ tolerance level do
  for all misclassified faces do
    increment $P(\angle e_{i,j}|\mathcal{F})\ \forall e_{i,j}$ in misclassified image
  end for
  for all misclassified nonfaces do
    increment $P(\angle e_{i,j}|\mathcal{F}^c)\ \forall e_{i,j}$ in misclassified image
  end for
end while
```

By increasing $P(\angle e_{i,j}|\mathcal{F})$ or $P(\angle e_{i,j}|\mathcal{F}^c)$ for the misclassified images, we effectively increase the likelihood that these faces will be classified correctly by the new model. The magnitude of the increment can be varied to train the system faster. Initially, a high increment is used so that the desired performance (high detection rate and low number of false positives) is reached quickly. Once close to the desired performance, the magnitude of the increment is decreased to avoid overshooting our performance goal.

Fifteen thousand nonface sample images are just a small subset of an image space consisting of $256^{(20\times 20)} \approx 10^{960}$ elements, and they are clearly not enough to achieve a low false-positive rate. Therefore, the system is further trained on about 500 large nonface images collected from the Internet using the same bootstrapping approach. The images are pictures of buildings, clothes, scenery, etc.

D. Issues With Overlapping Search Windows

In this work, a large image is raster scanned for faces using a 20×20 search window. Hence, there is a possibility that a face can be detected at multiple locations in an image and possibly at different scalings of the original image. This side effect can actually be exploited to reduce the number of false positives. Fig. 2 shows an example of how this can be done. The input image to the system is shown in Fig. 2(a). Detected faces for different search windows and scaling levels are indicated in Fig. 2(b) as white squares. Note that in this example the false positive is detected only once. Thus, clusters of detected faces in one region can be used to indicate where a face exists in the overall image.

Using clusters of detected faces was first proposed in [17]. In this work, a cluster is defined as a collection of detected faces with more than 70% overlap. The weight of a cluster is defined as the number of members in the cluster, but can also be defined as the sum of classification scores for each search window in the cluster. The representative for a cluster is defined as the search window with the largest classification score, but can also be defined as the search window with the largest area.

Two or more clusters can also overlap. In this case, it is likely that only one of the groups is a true face while the other is a false positive. In this case, the cluster with a higher total classification score is chosen and the other clusters are discarded.

Once all overlapping and false clusters are removed, the positions of all remaining cluster representatives are reported. Our lip feature extraction assumes that only one face will ever be present in a given scene; in this case, only the cluster representative with the highest classification score is reported.

IV. Lip Feature Extraction

The proposed lip feature extraction technique uses the contrast between the lips and the face to locate the four corners of the mouth. The position of the four corners in turn gives an estimate of the mouth's height and width. The operation of the proposed technique is shown in Fig. 3. Contrast, in the context of our lip feature extraction scheme, is defined as the average difference between pixels in a 2×2 pixel region.

Given the location of the face, the proposed technique searches for the mouth in the lower half of the face, Fig. 3(a). Fig. 3(b) and (c) shows the contrast and negative contrasts that exist in this image. In Fig. 3(c), a higher contrast is represented by white. Notice that the left and right corners of the mouth are where the contrast is highest. The left corner of the mouth is located by searching from the leftmost column of the search area toward the middle. In each column, we take the pixel with the highest contrast and compare its contrast with a threshold. If the contrast is greater than the threshold, that pixel is considered the left corner and we stop the search. If it is not, we

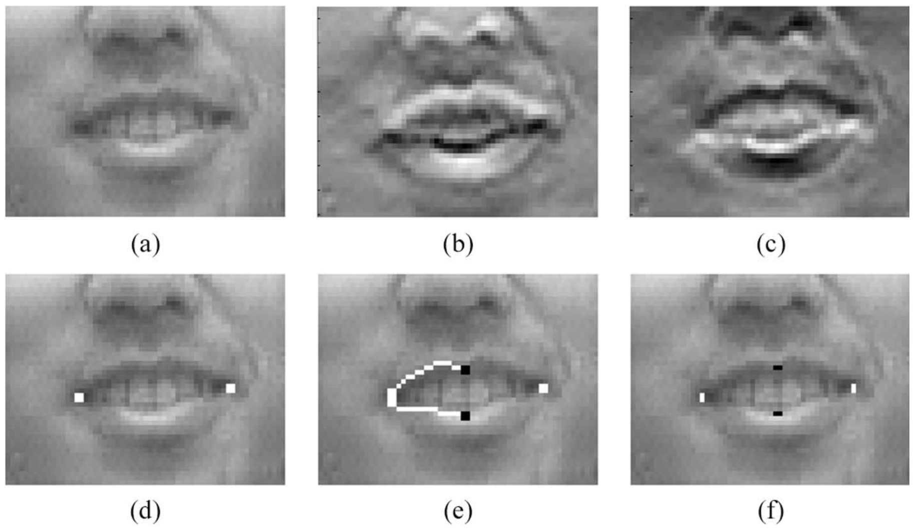

Fig. 3. Example of lip detection scheme. A contrast image (b) is constructed from the original image (a). Once the corners of the mouth are found (d), the contrast level is followed (e) to maximal top/bottom locations (f). (a) Lip search area. (b) Contrast. (c) Negated contrast. (d) Mouth corners. (e) Top/bottom search path. (f) Height/width points.

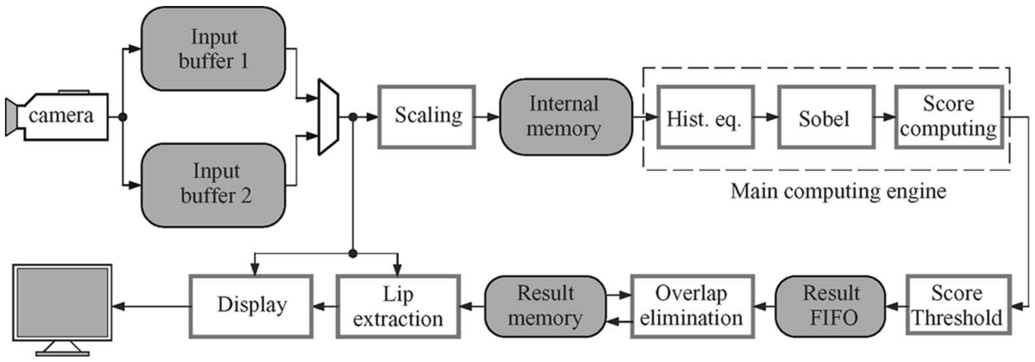

Fig. 4. Overall architecture of the system.

continue to the next column. The threshold can be made a tunable parameter to compensate for different lighting conditions. The right corner is located in a similar way, resulting in the two points shown in Fig. 3(d).

To locate the top of the lips, the proposed technique traces along the edge of the mouth by, starting at the left corner of the mouth, following neighboring points with the highest contrast. The search is terminated midway between the left and right corners of the mouth. The bottom of the lips can be found in a similar manner. An example of the search paths traced is shown in Fig. 3(d), the resulting points denoting the width and height of the lips are shown in Fig. 3(e). Note that the top of the lips indicated falls on the outside of the upper lip and the bottom of the lips indicated falls on the inside of the lower lip. This does in no way affect the ability for other systems to make use of the lip motion information provided as only the motion of the lips is important and not the lips' absolute position.

We found that this technique works better on faces that are larger than 20 × 20 pixels. We found that the face must be at least 80 × 80 pixels for this technique to work well. As such, the hardware implementation detects faces using a 20 × 20 search window, but performs lip motion extraction on faces that are at least 80 × 80 pixels.

TABLE I
IMAGE DOWN-SAMPLING FACTORS

	Factor	Image dimension
1	4.0	60×80
2	4.756	50×67
3	5.655	42×57
4	6.724	37×48
5	8.0	30×40
6	9.512	25×34
7	11.310	21×28

V. FIELD-PROGRAMMABLE GATE ARRAY (FPGA) IMPLEMENTATION

Fig. 4 shows the block diagram of the proposed hardware system. Video input is captured by a camera and stored in one of the two input buffer memories, each of which is 128 kB and can hold one 320 × 240 frame of video. Two input buffers are needed in order to be able to process a frame, while another is being buffered.

To detect faces in a frame of video, the original image is down-sampled by the scaling unit and stored in internal memory. Each down-sampled image is raster scanned using a 20 × 20 pixel search window. Each search window is histogram

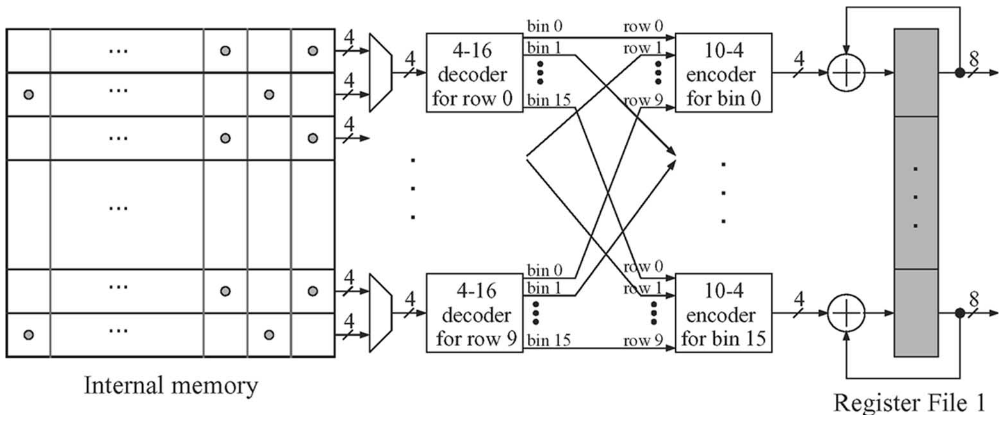

Fig. 5. Histogram equalization. Hardware to build pixel intensity histogram.

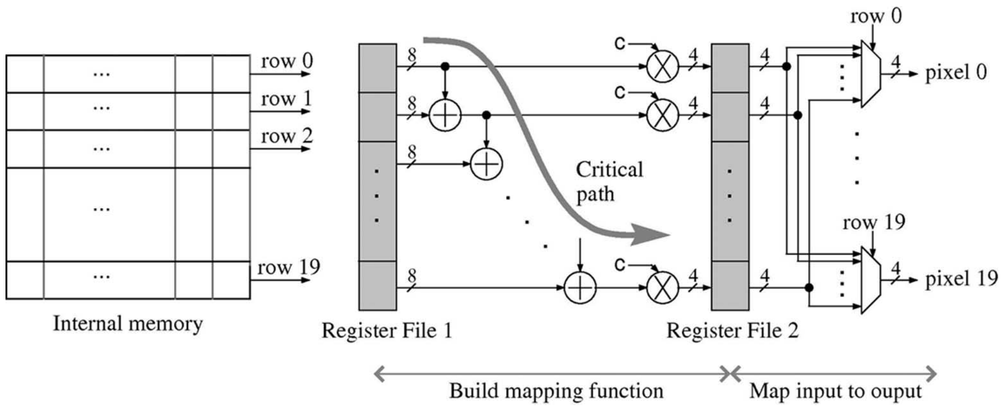

Fig. 6. Histogram equalization. Parts 2 and 3.

equalized and classified. If a search window's likelihood (score) is greater than a threshold, then the search window's information (its location, scale, and score) is stored in a resulting FIFO, to be processed by the overlap elimination (OE) block.

The OE block organizes overlapped faces into clusters and stores the locations of the detected faces into the resulting memory block. Since the OE block has to scan through each face cluster in the resulting memory block and the number of groups can vary during operation, the processing time of the OE block can vary. Hence, the resulting FIFO is needed to ensure that the main computing block does not have to wait for the OE block. The lip extraction block picks the face, from the resulting memory block, that has the highest score and searches for the four lip parameters. Finally, the display unit takes the output from the lip extraction block and displays the results on a monitor. To achieve real-time operation, all processing units were designed to process one column of data (20 pixels) in parallel per cycle. Let us now look at each processing block in more detail.

A. Scaling Unit

The scaling unit takes a 320×240 pixel input video frame from one of the two input buffers and produces a selectively down-sampled image. There are seven down-sampling factors that are used (shown in Table I). This scaled image is then stored in the internal memory block.

Image down-sampling is performed using the simple nearest neighbor technique. That is, pixel (i', j') in the scaled image is assigned the value of the pixel closes to $\beta(i, j)$ in the original image, where β is the scale factor used. Scaling is performed by simply coping the corresponding image pixels from the input buffer into the internal memory block. The scaling factors and the corresponding image sizes are fixed parameters in this implementation and are stored in a read-only memory (ROM).

B. Organization of the Internal Memory Unit

Memory access proved to be a substantial bottleneck, as reading a 20×20 search window would take 400 clock cycles from a single ported memory. Hence, to reduce access time, we use 20 smaller memories that are accessed in parallel. Each memory block is 256 B and stores a single row of the scaled image such that one column of the search window can be read from the memory subsystem in parallel.

C. Histogram Equalization Unit

Histogram equalization allows the system to work well in different lighting conditions. This is the most complex block in the system because it has to perform three operations. First, it has to build the intensity histogram of the input window. Next, it has to build the cumulative density function (CDF) of the pixels. Finally, it has to build the new image based on the input and the scaled CDF.

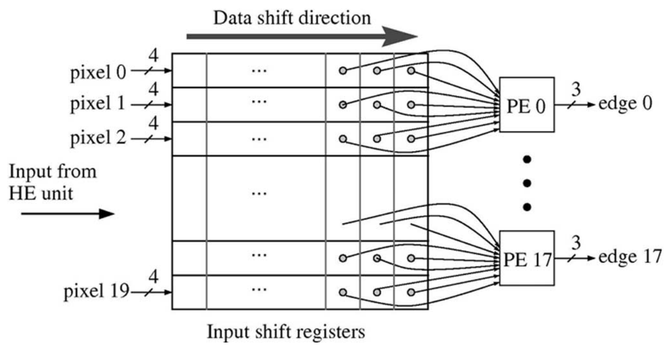

Fig. 7. Sobel filter unit.

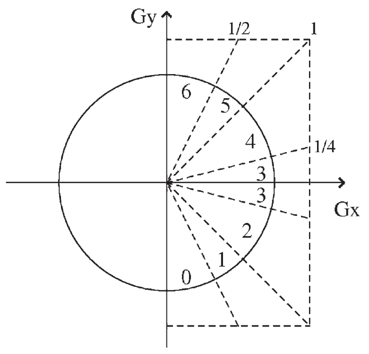

Fig. 8. Nonlinear edge direction quantization chart.

The histogram builder counts the appearance of intensity values in the search window. This is a sequential task, but the proposed system processes the pixels in one column of the image in parallel. Hence, we first assign pixel intensities to histogram bins and then count the number of pixels assigned to each bin. To further speed up histogram equalization, the dynamic range of pixel intensities is down-sampled by 16 and only every other pixel intensity is examined. Compressing the dynamic range of pixel intensities reduces the number of histogram bins required: whereas, reducing the number of pixels examined reduces the number of parallel processing elements required. The resulting architecture is shown in Fig. 5.

For each 20-pixel column, only the ten odd or even pixels intensities are examined (ones marked with a gray dot in Fig. 5). These are selected using ten 2-to-1 multiplexers. The down-sampled intensity of the ten pixels are one-hot encoded using the 4-to-16 decoders. Each bit corresponds to the histogram bin that should increment its value. Each histogram bin examines the one-hot encoded bits it receives that are active and generates a binary increment using the 10-to-4 encoders. Each bin's decoded binary increment is added to the bin's running total, stored in register file 1 (RF1). Thus, a possible 400-clock cycle histogram operation now requires only 20 clock cycles.

To build the scaled CDF, we need to calculate the cumulative frequency of each intensity value starting at the lowest intensity. This is easily accomplished with a ladder of adders, as shown in Fig. 6, which cumulatively sum up the outputs of RF1. The output of each addition is scaled by a constant factor c such that the result falls within the range 0–15. The results of these operations are stored in RF2.

The equalized image is constructed a column at a time using the original image's down-sampled pixel intensities to select a corresponding normalized intensity from RF2. This is done using twenty 16-to-1 multiplexers. As each column is processed, it is passed over to the Sobel filter.

D. Sobel Filter Unit

The Sobel filter unit extracts and quantizes edges from the equalized image, the search window. The block diagram for this block is shown in Fig. 7.

This block uses 20 FIFO memory blocks to store the search window, one FIFO block per row of the search window. Each FIFO is 20 entries deep, deep enough to store a complete row of the search window; each entry is 4 bits wide. The search window is shifted into this FIFO after having its histogram equalized. The last three entries in each FIFO are connected to the 18 Sobel processing elements (PE), which extract the quantized direction of an edge per 18 of the rows. Data are continuously shifted toward the right in Fig. 7, thereby processing a column of edges per clock cycle.

Each PE computes the horizontal and vertical edges present ($e_{\mathrm{x}_{i,j}}$ and $e_{\mathrm{y}_{i,j}}$) using eight adders. The net direction of each corresponding pair of edges is then evaluated, as per (2), and quantized. Edge direction is not computed using an implementation of an inverse tangent function, as such an implementation is typically iterative or requires substantial resources. Instead, the quantized direction is computed directly based on the values of $e_{\mathrm{x}_{i,j}}$ and $e_{\mathrm{y}_{i,j}}$ using the chart shown in Fig. 8. These quantization levels are not divided evenly over the half-circle, but in such a way that the quantized direction level can be easily evaluated based on $e_{\mathrm{x}_{i,j}}$ and $e_{\mathrm{y}_{i,j}}$. This allows for the computation of the quantized direction using only three comparators and a small logic element.

The magnitude of an edge is computed as the sum of $e_{\mathrm{x}_{i,j}}$ and $e_{\mathrm{y}_{i,j}}$ using two multipliers and one adder. The square root of this sum is not computed, as it is easier to just square the threshold value *a priori*. If the squared sum is less than the threshold, then the edge is characterized as a weak edge.

E. Score Computing Unit

The score (likelihood that a search window is a face) is obtained by using the quantized edge directions to look-up the precomputed likelihood $(P(\angle e_{i,j}|\mathcal{F}))$ associated with that edge. A precomputed likelihood associated with all possible eight edge directions needs to be stored. These likelihoods are stored in a look-up table (LUT). The 18 LUTs for a row of search window pixels are in turn grouped into a single ROM. That is, each row of the search window has an associated ROM in which 18 LUTs are stored. This allows for parallel processing of a column of the search window, as shown in Fig. 9.

As a column of edge directions is received from the Sobel filter unit, they are buffered into the input shift registers. Each three-bit quantized edge direction is used to look-up the likelihood value associated with that edge direction. This is done in parallel for each row of the column being analyzed. A counter

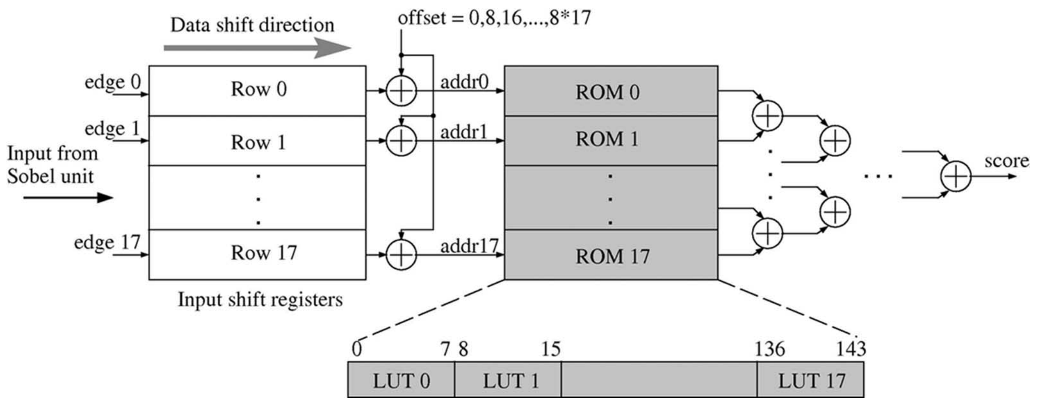

Fig. 9. Score (face likelihood) computing unit.

is used to offset each column to point to its respective LUT in the ROM in order to obtain the correct likelihoods for each of the 18 × 18 pixels in the search window. Hence, the counter points to an LUT in the ROM and the edge direction indexes a single entry in the LUT. The outputs of each row of ROMs are summed for all search window columns processed.

Each edge direction likelihood (LUT entry) is represented by a 16-bit fixed-point number. One bit is reserved for the sign of the number and the decimal point is to the right of the first mantissa bit, representing values in the range $[-2, 2)$. The precision of this representation is 2^{-14}.

F. Overlap Elimination Unit

This block organizes detected faces into clusters. The locations and scores of detected faces, if any, are read from the resulting FIFO. For each detected face in the resulting FIFO, we search the resulting memory for a cluster that the detected face belongs to. If a corresponding cluster exists, information about that cluster is updated to include the current detected face. Otherwise, a new group is created. This unit is the main performance bottleneck in this system.

G. Lip Tracking Unit

The face that has the highest score from the resulting memory is used for lip feature extraction. The video frame is read from the active input buffer and the left and right corners of the mouth are located using the contrast information for the lower half of the face. The top and bottom of the mouth are found by tracing along the lip border.

The design can be broken into two main blocks: one to search for the left and right corners of the mouth and the other one to search for the top and bottom of the mouth. Both blocks use a common block to compute the contrast of the image, which uses several adders. The left–right search block accepts the starting point and search direction as its input. The top–bottom search block takes the search direction (up or down) and the starting and ending points, which were found previously using the left–right search block. The result from the lip tracking block can be either displayed on a monitor or sent to a host computer.

TABLE II
IMPLEMENTATION CHARACTERISTICS

RAM usage	268.2 KB
Area	15,050 Logic cells
Max clock frequency	41 MHz
Max frame rate	136.6 frames/sec

VI. Results

A. Implementation Characteristics

The system was implemented on a Microtronix Stratix Development Kit that houses an Altera Stratix 40 000 logic element FPGA. The characteristics of this implementation are summarized in Table II.

The design uses 268 kB of memory, most of which (256 kB) is devoted to the two input buffers. The design takes up 15 050 (38%) logic elements. Approximately 300 000 clock cycles are required to process each frame through the face detection subsystem. Lip feature extraction takes only a few thousand clock cycles. Synthesis results indicated that the design can operate at a maximum clock frequency of 41 MHz. Hence, this implementation can potentially process 136.6 fps, 30 fps operation can be achieved using a 9-MHz clock frequency.

B. Face Detection Accuracy

The proposed face detection was tested on the Yale test set [31] to measure the effectiveness of the system under various lighting conditions. The test set consists of 164 images, each face with different expressions (angry, surprised, etc.) under different lighting conditions. The proposed system achieved a detection rate of 86.6%, with no false positives. A sample of images from this test case is shown in Fig. 10. This demonstrates that our system is able to detect faces over a wide range of lighting conditions and only fails under extreme lighting conditions.

The targeted application of our proposed system is audio–visual speech processing, specifically for teleconference applications. To gauge the performance our system with respect to the intended target application, we collected approximately 1600 images of different speakers sitting in front of a camera and talking. That is, each image only has one face in the frame. The speakers were free to move their head naturally while they

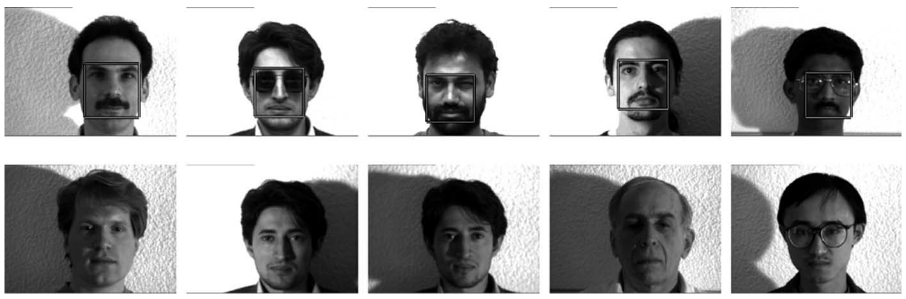

Fig. 10. Results from the Yale test set. The upper row shows correctly detected faces, and the bottom row shows missed faces.

TABLE III
FACE DETECTION RESULTS

Test set	Detection rate	False-positives
Yale	86.6%	0%
Single speaker	99.2%	0.125%

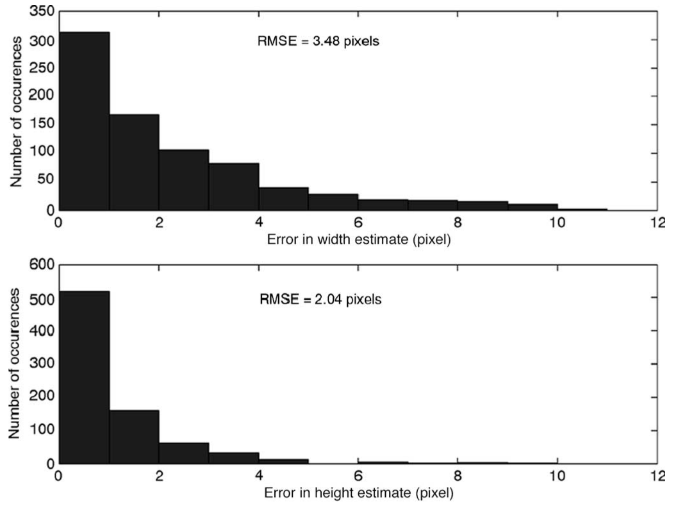

Fig. 11. Error distributions for the automatic height and width measurements compared to manual measurements.

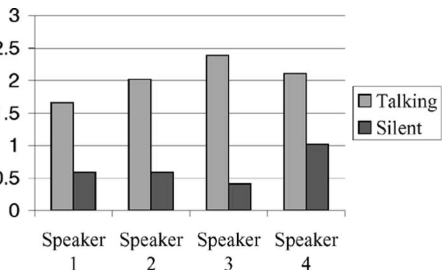

Fig. 12. Standard deviations of the height estimates in the speaking and silent cases.

TABLE IV
COMPARISON TO PREVIOUS FPGA IMPLEMENTATION

	Logic elements	Video rate (frames/second)[1]	Detection accuracy
McCready system	89,856	30 fps	80.9%
Our system	15,050	41.7 fps	86.6%

[1]Using a 320×240 frame size with a processing clock frequency of 12.5 MHz.

conversed. The system achieved a 99.2% detection rate, with only two false positives. Table III summarizes the performance of our system with respect to the Yale test case and our own test scenario.

C. Lip Feature Extraction Accuracy

The lip feature extraction technique was tested on four speakers. A video of 200 frames was recorded for each speaker. The videos were processed by the lip extraction system, from which the lip widths and heights were calculated. We compared these automatically extracted measurements against manual measurements. Our system achieved a root mean squared (rms) error of 2.04 pixels for the height measurements and an rms of 3.48 pixels for the width measurements. The error distributions are shown in Fig. 11.

In order to use lip motion analysis for speech processing, the system needs to detect periods where the speaker of interest is silent, where the lips are not moving [3]. To test how the proposed system is able to detect silent speakers, we recorded another set of videos where the speakers are silent and do not move their lips. The standard deviations of lip height measurements for speaking and nonspeaking speakers are plotted in Fig. 12. Note that there is a clear distinction between talking and silent cases.

The results shown above are derived from a small test sample and are intended that the simple lip detection proposed does indeed work and the results are thus far promising.

D. Comparison to Alternative FPGA-Based Systems

Table IV compares the proposed system to the FPGA-based face detection system proposed in [29]. Note that our system requires six times less logic elements. There might exist a slight discrepancy between the number of logic elements between our proposed implementation and the implementation in [29] as the latter is based on an older FPGA.

Our system performs better in various lighting conditions because we use a histogram equalization technique that improves our system's ability to detect edges in an image. Our system can run at a higher clock frequency and can therefore process a higher rate of frames per second. However, for this comparison, we used the same clock frequency as the implementation in [29] (12.5 MHz), and our system was still able to process 41.7 fps, as opposed to 30 fps.

VII. Conclusion

This paper proposes a new face detection technique that utilizes a naive Bayes classifier to detect faces in an image based on only image edge direction information. This technique allows for a compact FPGA-based implementation that can operate in real-time at frame rates of over 30 fps. A simple lip feature extraction technique is also proposed, which uses contrast information in the image to locate the corners, top, and bottom of the mouth. This implementation was also implemented on the same FPGA. We showed that our combined face detection and lip feature extraction system could discern whether a speaker was speaking or silent. Information on when a speaker of interest is silent can be used to gather information about the interfering noise(s); this information can be used to improve the fidelity of the signal originating from the speaker of interest.

References

[1] L. Deng, A. Acero, M. Plumpe, and X. Huang, "Large-vocabulary speech recognition under adverse acoustic environments," in *Proc. Int. Conf. Spoken Language Processing (ICSLP)*, Beijing, China, 2000, vol. 3, pp. 806–809.

[2] P. Aarabi and B. V. Dasarathy, "Robust speech processing using multi-sensor multi-source information fusion—An overview of the state of the art," *Inf. Fusion*, vol. 5, no. 2, pp. 77–80, Jun. 2004.

[3] P. Aarabi and B. Mungamuru, "The fusion of visual lip movements and mixed speech signals for robust speech separation," *Inf. Fusion*, vol. 5, no. 2, pp. 103–117, Jun. 2004.

[4] G. Potamianos, G. Gravier, A. Garg, and A. W. Senior, "Recent advances in the automatic recognition of audiovisual speech," *Proc. IEEE*, vol. 91, no. 9, pp. 1306–1326, Sep. 2003.

[5] X. Zhang, C. C. Broun, R. M. Mersereau, and M. A. Clements, "Automatic speechreading with applications to human-computer interfaces," *EURASIP J. Appl. Signal Process., Special Issue on Audio-Visual Speech Processing*, vol. 1, pp. 1228–1247, 2002.

[6] K. Nakamura, N. Murakami, K. Takagi, and N. Takagi, "A real time lipreading LSI for word recognition," in *Proc. IEEE Asia-Pacific Conf. ASIC*, Taipei, Taiwan, 2002, pp. 303–306.

[7] I. Mathews, G. Potasmianos, C. Neti, and J. Luettin, "A comparison of model and transform-based visual features for audio-visual LVCSR," in *Proc. IEEE Int. Conf. Multimedia and Expo.*, Tokyo, Japan, 2001, pp. 1032–1035.

[8] G. Potamianos and C. Neti, "Improved ROI and within frame discriminant features for lipreading," in *Proc. Int. Conf. Image Processing*, Thessaloniki, Greece, 2001, pp. 250–253.

[9] G. Potamianos, H. P. Graf, and E. Cosatto, "An image transform approach for HMM based automatic lipreading," in *Proc. Int. Conf. Image Processing*, Chicago, IL, 1998, pp. 173–177.

[10] J. Yang and A. Waibel, "A real-time face tracker," in *Proc. 3rd IEEE Workshop Applications Computer Vision*, Sarasota, FL, 1996, pp. 142–147.

[11] M. Hunke and A. Waibel, "Face locating and tracking for human-computer interaction," in *Proc. 28th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, 1994, pp. 1277–1281.

[12] R. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 696–706, May 2002.

[13] D. Mario and D. Maltoni, "Real-time face location on gray-scale static images," *Pattern Recognit.*, vol. 33, no. 9, pp. 1525–1539, Sep. 2000.

[14] S. Paschalakis and M. Bober, "A low cost FPGA system for high speed face detection and tracking," in *Proc. IEEE Int. Conf. Field-Programmable Technology*, Tokyo, Japan, Dec. 2003, pp. 214–221.

[15] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proc. Int. Conf. Computer Vision*, Bombay, India, 1998, pp. 555–562.

[16] H. Sahbi, D. Geman, and N. Boujemaa, "Face detection using coarse-to-fine support vector classifiers," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Rochester, NY, 2002, pp. 925–928.

[17] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–28, Jan. 1998.

[18] ——, "Rotation invariant neural network-based face detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Santa Barbara, CA, Jun. 1998, pp. 38–44.

[19] R. Feraud, O. J. Bernier, J. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 42–53, Jan. 2001.

[20] H. Schneiderman and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition," in *Proc. Conf. Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998, pp. 45–51.

[21] ——, "A statistical model for 3D object detection applied to faces and cars," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Hilton Head Island, SC, 2000, pp. 746–751.

[22] H. Schneiderman, "Learning statistical structure for object detection," in *Proc. 10th Int. Conf. Computer Analysis Images and Patterns*, Groningen, The Netherlands, 2003, pp. 434–441.

[23] M. Kass, A. Witkin, and D. Terzopoulos, "SNAKES: Active contour models," in *Proc. 1st Int. Conf. Computer Vision*, London, U.K., 1987, pp. 259–268.

[24] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models—Their training and application," *Comput. Vis. Image Underst.*, vol. 12, no. 1, pp. 321–332, Jan. 1995.

[25] I. Mathews, T. F. Cootes, J. A. Bangham, S. Cox, and R. Harvey, "Extraction of visual features for lipreading," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 198–213, Feb. 2002.

[26] T. Theocharides *et al.*, "Embedded hardware face detection," in *Proc. Int. Conf. VLSI Design*, Mumbai, India, 2004, pp. 133–138.

[27] R. Gonzalez and R. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1992.

[28] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 39–51, Jan. 1998.

[29] R. McCready, "Real-time face detection on a configurable hardware platform," M.S. thesis, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2000.

[30] M. P. Fitzgerald. (2002). *CBCL Face Database #1*, Center for Biological and Computational Learning at MIT. [Online]. Available: http://cbcl.mit.edu/software-datasets/FaceData2.html

[31] A. Georghiades. (1997). *Yale Face Database*, Center for Computational Vision and Control at Yale University. [Online]. Available: http://cvc.yale.edu/projects/yalefaces/yalefaces.html

Duy Nguyen received the M.A.Sc. and B.A.Sc. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2005 and 2003, respectively.

His area of research involved hardware-based signal processing, human–computer interaction, and the hardware implementation of complex algorithms.

David Halupka (S'99) received the B.A.Sc. degree (with honors) in computer engineering and the M.A.Sc. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2002 and 2004, respectively, and is currently working toward the Ph.D. degree at the University of Toronto.

He holds the prestigious Natural Sciences and Engineering Research Council of Canada's Canadian Graduate Scholarship for the term of his Ph.D. degree. He has also held the Ontario Graduate Scholarship for the term of his M.A.Sc. He was the recipient of the IEEE Canadian Foundation's McNaughton Scholarship in 2001.

Parham Aarabi (M'02) received the B.A.Sc. degree in engineering science (electrical option) and the M.A.Sc. degree in computer engineering from the University of Toronto, Toronto, ON, Canada, in 1998 and 1999, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2001.

He holds a Canada Research Chair in Multi-Sensor Information Systems, a tenured Associate Professor in The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, and the Founder and Director of the Artificial Perception Laboratory. His current research, which includes multisensor information fusion, human–computer interactions, and hardware implementation of sensor fusion algorithms, has appeared in over 50 peer-reviewed publications and covered by media such as *The New York Times*, MIT's *Technology Review Magazine*, *Scientific American*, *Popular Mechanics*, the Discovery Channel, CBC Newsworld, Tech TV, Space TV, and City TV.

Dr. Aarabi's recent award include the 2002–2004 Professor of the Year Award, the 2003 Faculty of Engineering Early Career Teaching Award, the 2004 IEEE Mac Van Valkenburg Early Career Teaching Award, the 2005 Gordon Slemon Award, the 2005 TVO Best Lecturer (Top 30) selection, the Premier's Research Excellence Award, as well as MIT Technology Review's 2005 TR35 "World's Top Young Innovator" Award.

Ali Sheikholeslami (S'98–M'99–SM'02) received the B.Sc. degree from Shiraz University, Shiraz, Iran, in 1990, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1994 and 1999, respectively, all in electrical and computer engineering.

He has collaborated with industry on various very large scale integrated (VLSI) design projects in the past few years, including work with Nortel, Canada, in 1994, with Mosaid, Canada, since 1996, and with Fujitsu Labs, Japan, since 1998. In 1999, he joined the Department of Electrical and Computer Engineering, University of Toronto, where he is currently an Associate Professor. He is currently spending the first half of his sabbatical year with Fujitsu Labs in Japan. His research interests are in the areas of analog and digital integrated circuits, high-speed signaling, VLSI memory design (including SRAM, DRAM, and CAM), and ferroelectric memories. He presently supervises three active research groups in the areas of ferroelectric memory, CAM, and high-speed signaling. He has coauthored several journal and conference papers (in all three areas), in addition to two U.S. patents on CAM and one U.S. patent on ferroelectric memories.

Dr. Sheikholeslami has served on the Memory Subcommittee of the IEEE International Solid-State Circuits Conference (ISSCC) from 2001 to 2004, and on the Technology Directions Subcommittee of the same conference from 2002 to 2005. He presented a tutorial on ferroelectric memory design at the ISSCC 2002. He was the Program Chair for the 34th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2004), Toronto. He is a Registered Professional Engineer in the province of Ontario, Canada. He received the Best Professor of the Year Award in 2000, 2002, and 2005 by the popular vote of the undergraduate students at the Department of Electrical and Computer Engineering, University of Toronto.