Nonparametric Approaches for Estimating Driver Pose

Paul Watta, *Member, IEEE*, Sridhar Lakshmanan, and Yulin Hou

***Abstract*—To better understand driver behavior, the Federal Highway Administration and the National Highway Traffic Safety Administration have collected several thousands of hours of driver video. There is now an immediate need for devising automated procedures for analyzing the video. In this paper, we look at the problem of estimating driver pose given a video of the driver as he or she drives the vehicle. A complete system is proposed to perform feature extraction and classification of each frame. The system uses a Fisherface representation of video frames and a nearest neighbor and neural network classification scheme. Experimental results show that the system can achieve high accuracy and reliable performance.**

***Index Terms*—Classification, driver pose estimation, eigenfaces, fisherfaces, neural networks, video.**

I. Introduction

In ORDER to design the next generation of automotive safety and comfort devices, the intelligent transportation systems (ITS) community has been very interested in understanding driver behavior [1], [2], [16], [17]. There are a number of reasons why driver behavior is of interest.

1) The way a driver behaves behind the wheel can give an indication as to the driver's attentiveness and/or fatigue. For example, drivers who are falling asleep at the wheel tend to blink at a faster rate than drivers who are fully awake [2], [24].
2) As more and more communication and travel assistance devices make their way into the vehicle, it is important to understand how they impact driver performance. For example, drivers that use cellular telephones are more prone to being inattentive [21].
3) When driving assistance systems such as backup or blind spot collision warning systems are installed in vehicles, their utility is enhanced significantly if the systems are strategically placed. For example, drivers tend to look at the rear-view mirror most often during a lane change [24]. What does that mean in terms of the optimal location for a blind-spot detection system?
4) Even state-of-the-art vision and radar-based forward collision warning systems produce an abundance of false alarms and quickly become a nuisance to the driver. Perhaps one way of integrating these systems into the vehicle is by activating them only in situations where the driver is inattentive.

A number of recent ITS conference papers [16], [17] can be consulted for additional reasons for investigating driver behavior.

One approach to understanding driver behavior is to try to infer what the driver is doing based on vehicle data, such as speed, yaw, and braking. Although these types of vehicle data can certainly be correlated with driver attentiveness and fatigue, they are neither sufficiently accurate/reliable nor do they provide a complete description of the driver's state [1]. Systems that combine vehicle data with human factor information are more accurate/reliable and provide a more complete description of the driver's state.

A commonly used method for measuring human factors information is to use a video camera and record the driver (both face and hands) as he or she drives the vehicle. A massive project initiated by the Federal Highway Administration and the National Highway Traffic Safety Administration (NHTSA) has indeed collected several thousands of hours of such video. There is now an immediate need for analyzing this facial video. Unfortunately, analysis of facial images and video is a nontrivial problem, as the volume of previous studies indicate [5].

The driver pose estimation problem is formulated as follows.

1) Frame Classification Problem: Given a video sequence consisting of a driver as he or she drives a vehicle, determine the pose of the driver for each frame in the sequence.

An example of a single video frame is shown in Fig. 1 and shows the driver looking straight at the road ahead.

Although there are an endless number of possibilities, the NHTSA has formulated seven standard driver pose classes:

Pose 1 looking over the left shoulder;
Pose 2 looking in the left rear view mirror;
Pose 3 looking at the road ahead;
Pose 4 looking down at the radio/instrument panel;
Pose 5 looking at the center rear view mirror;
Pose 6 looking at the right rear view mirror;
Pose 7 looking over the right shoulder.

Although simple to state, the pose estimation problem is rather difficult to solve. Factors that are common to all image processing problems contribute to the difficulty: image

Manuscript received July 24, 2002; revised March 30, 2005 and February 27, 2006. This work was supported in part by a grant from the Department of Transportation under Small Business Innovation Research (SBIR) Contract DTRS57-00-C-10025 and in part by a grant from the University of Michigan Office of the Vice President of Research. The review of this paper was coordinated by Prof. D. Lovell.

The authors are with the Vehicle Electronics Laboratory, University of Michigan-Dearborn, Dearborn, MI 48128 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2007.897634

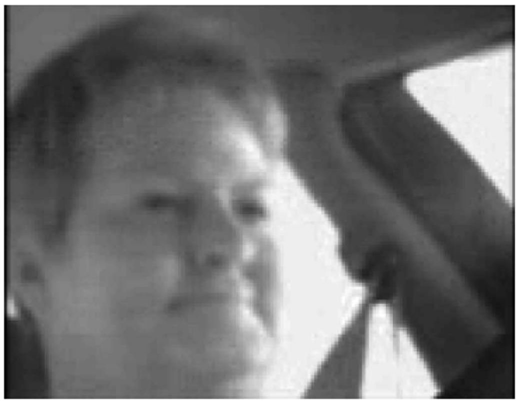

Fig. 1. Example frame from the driver video.

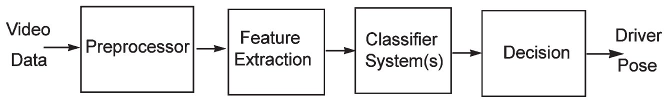

Fig. 2. Complete pose estimation system.

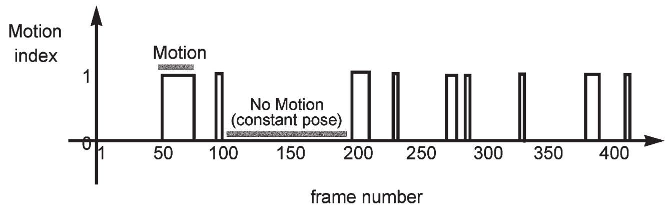

Fig. 3. Output of the motion preprocessor which identifies areas where there is motion. The areas between motion indicators are those frames which contain relatively little motion (i.e., the driver has constant pose).

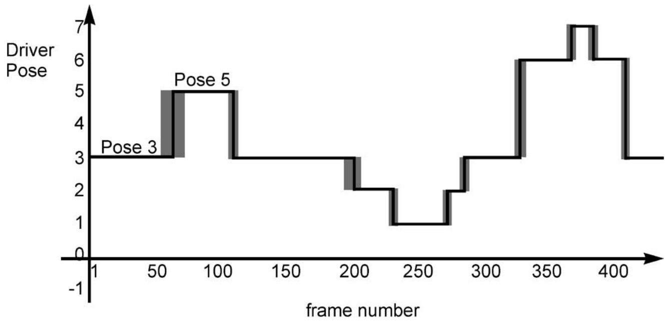

Fig. 4. Typical output of the pose estimation system.

noise, shift, rotation, and changes in illumination. In addition, the problem suffers from several difficulties specific to this application.

1) The driver can change his/her facial expression without any apparent change in pose.
2) The driver's face can be occluded when the driver touches his/her face or hair, or eats food.
3) The driver can tilt his/her head without any apparent change in pose.
4) The background can change due to vehicle motion.

Of course, one way to determine driver pose is by hand, and have a human operator look at the video and mark those frames where the driver changes pose, and then, by visual inspection, classify the resulting pose sequences. This process, however, is very tedious and prone to errors. For example, to partition and classify one 20-min video took a student assistant 6 h! Even so, the resulting ground truth contained a number of missed pose transitions and sequences that were misclassified. Clearly, an automatic and computer-based classification method is needed.

A schematic diagram of an automated pose estimation system is shown in Fig. 2. There are four main processing stages: preprocessing, feature extraction, classification, and decision.

In the preprocessing stage, the signal from the video camera or video data file is converted into a sequence of frames suitable for further processing. In this paper, the original video signal is in the form of an MPEG file. The preprocessor decodes the input MPEG video into the raw pixel values for each frame (8-b RGB grayscale pixel data).

Additional processing can be done in this initial stage as well. For example, we can use motion detection in order to segment the video into chunks of constant pose. Fig. 3 shows a typical output of such a motion detector. Here, the output of the motion detector is 1 if there is significant motion in the video; otherwise, the output is 0. For example, Fig. 3 shows that there is significant motion between frames 50 and 75. One of the primary causes of the motion will be when the subject changes pose. However, there can be motion due to other causes as well, such as if the driver is eating or moving their arms and hands, etc.

The next stage of the pose estimation system involves feature extraction. Here, the problem is one of representation, and the task is to choose an efficient way to extract salient features from the video data. We desire features which will minimize the amount of storage and processing time required. Of course, the chosen representation must maintain enough information so that the classifier can adequately discriminate between the target pose classes.

Once a data representation is chosen, the classifier design problem involves designing a system to map each feature vector to the proper pose class. Typically, a training set of randomly selected patterns is used to design the classifier. After training, the system is tested with another data set called the test set, which contains novel patterns.

Note that several different classifiers may be used in the classification stage. Hence, a final decision stage is needed in order to combine the classifier outputs and make a final decision as to the driver pose. Sometimes, postprocessing is useful for smoothing the classifier outputs. Such postprocessing is also included as part of the decision stage.

Given an input video file, the complete system should produce an output such as the one shown in Fig. 4. Here, a pose is assigned to each frame of the video. The transition regions are shown highlighted in gray. It is expected that the system will make errors in these transition regions, as the driver is between poses.

It is desired that the system be invariant to changes in image illumination, facial movement, occlusions, rotations, etc. For example, the system should work just as well when driving under a tunnel (dark video images) as when driving out in the open on a sunny day (bright video images). The design of systems that are invariant to such distortion effects presents a major obstacle for feature extraction and tracking [5].

Driver pose estimation is a special case of the more general problem of pose estimation in image processing. Gee and Cipolla [7] proposed a technique using projective geometry to estimate the pose of a human face from five features: the far corners of the eyes and mouth and the tip of the nose. Since estimating the position of the tip of the nose is not an easy task, Ho and Huang [9] proposed a technique that just relied on the corners of the eyes and the mouth. For the problem of eye gaze detection, it is possible to use information from just one eye [26].

Nikolaidis and Pitas [15] proposed a technique for pose estimation that used both feature extraction (using the Hough transform) and template matching. Sherrah and Gong [19] proposed a head tracking system that relied on a data fusion technique to combine head position information with face pose information.

Kruger *et al.* [12] proposed a Gabor wavelet-based technique. This approach requires a computationally intensive elastic graph matching technique and requires the user to select (by hand) a suitable grid of matching points for each pose. Using specialized hardware, McKenna and Gong [14] developed a Gabor-based features pose estimation system that ran in real time. Huang *et al.* [10] proposed a technique based on support vector machines. Like Gabor wavelets, although, this method is extremely computationally intensive.

In this project, we propose a nonparametric technique to determine driver pose from video data. The technique is based on computing an eigenface [20], [22], [23] or Fisherface representation [3] of the video data, and it completely avoids the computation of (and the associated pitfalls of) explicit geometric facial features (for example, eye positions, etc.). The set of Fisherfaces is used to compactly represent each standard pose of the driver. The pose of an input image can be determined by first computing the Fisherface representation of the input and then comparing it to the representation of the set of stored prototypes. We also introduce a novel neural network which can be used to optimize a nearest neighbor-type classification. Note that many neural network-based approaches for the processing of face images can be found in the literature [4], [6], [8], [11], [18].

The remainder of this paper is organized as follows. In Section II, we review the representation problem and outline how features can be efficiently extracted from the video data. In Section III, we review the operation of the nearest neighbor and the max–min classifiers. We also propose a hybrid nearest neighbor-neural network classifier. In Section IV, we discuss the decision problem and how the classifier output can be biased to improve performance. In Section V, we introduce various measures to assess pose estimation system performance. In Section VI, we describe the database of driver videos that we used in order to design and test our system, and in Section VII, we present experimental results on these videos. Finally, Section VIII gives an analysis of the results and outlines future work.

II. Representation and Feature Extraction

We will assume that we have a database or memory set containing sample images of each of the classes of interest. In this problem, there are seven classes of interest. Let M denote the total number of training samples, and let m_i represent the number of samples of class i, $i = 1, 2, \ldots, 7$. Hence, we have $M = m_1 + m_2 + \cdots + m_7$. The training images will be denoted $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_M$, and each training image is assumed to be an N-dimensional vector of grayscale pixel data. In addition, let t_j represent the facial pose for each memory image $\mathbf{I}_j$ so the database has the following form: $\{(\mathbf{I}_1, t_1), (\mathbf{I}_2, t_2), \ldots, (\mathbf{I}_M, t_M)\}$.

A. Eigenfaces

The eigenface algorithm employed here is based on a direct implementation of the ones in [20], [22], and [23]. Some of our preliminary results in applying eigenfaces to pose estimation can be found in [27].

An $N \times M$ dimensional matrix of zero mean images $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]$ can be constructed with column entries $\mathbf{x}_j = \mathbf{I}_j - \mathbf{I}_{\text{avg}}$, where $\mathbf{I}_{\text{avg}}$ is the average of all the images: $\mathbf{I}_{\text{avg}} = (1/M)\sum_{j=1}^{M} \mathbf{I}_i$.

The covariance matrix of $\mathbf{X}$ is then given by $\mathbf{C} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$. In general, $M \ll N$, and so, rather than compute the eigenvectors of the $N \times N$ matrix $\mathbf{C}$, we compute the eigenvectors of the $M \times M$ matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. The eigenvectors (or eigenfaces) $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M$ of the covariance matrix $\mathbf{C}$ are then computed as $\mathbf{X} \bullet \mathbf{eig}(\mathbf{X}^{\mathrm{T}}\mathbf{X})$. For each training image $\mathbf{I}_j$, an eigen representation $w_{kj} = \mathbf{e}_k^{\mathrm{T}}\mathbf{x}_j$, $j = 1, 2, \ldots M$ is obtained by projecting the corresponding $\mathbf{x}_j$ onto each of the eigenfaces $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M$. Therefore, each database image $\mathbf{I}_j$ is now represented by an M-dimensional vector of eigen coefficients $\mathbf{w}_j$. For $M \ll N$, this amounts to a significant dimensionality reduction. In addition, the dimension can be further reduced by only considering the eigenvectors which have sufficiently large eigenvalues. We will denote the number of eigenvectors used by M'.

B. Fisherfaces

While the eigen approach allows for a low-dimensional representation of the training images, it implicitly reduces the discrimination between training images. While this is a useful property for compactly coding images, it may have some undesirable consequences when it comes to classification. The Fisher representation, on the other hand, provides better discrimination between classes while preserving the reduced discrimination property of the eigen representation within each class.

In the Fisherfaces algorithm [3], [13], we compute the average image within each of the seven classes. That is, for each $i = 1, \ldots, 7$

$$\mu_i = \frac{1}{m_i} \sum_{\mathbf{x}_j \text{ in class } i} \mathbf{x}_j$$

is the average of all the memory set images which are in class i.

The within-class scatter matrix $\mathbf{S}_\mathrm{W}$ is an $N \times N$ matrix which provides a measure of the amount of variation among the memory set images within each class and is given by

$$S_\mathrm{W} = \sum_{i=1}^{7} \sum_{\mathbf{x}_k \in \mathrm{Class}_i} (\mathbf{x}_k - \mu_i)(\mathbf{x}_k - \mu_i)^\mathrm{T} = \sum_{i=1}^{7} \mathbf{A}_i \mathbf{A}_i^\mathrm{T}.$$

The between-class scatter matrix $\mathbf{S}_\mathrm{B}$ is also $N \times N$ and measures the amount of variance between the means of each class and is computed by

$$\mathbf{S}_\mathrm{B} = \sum_{i=1}^{7} m_i(\mu_i - \mathbf{I}_\mathrm{avg})(\mu_i - \mathbf{I}_\mathrm{avg})^T.$$

The Fisher criterion is to choose a linear projection such that the distance between the projected class means is maximized, but the selection is normalized by the within-class scatter. This amounts to computing the eigenvectors of the matrix $\mathbf{S}_\mathrm{W}^{-1}\mathbf{S}_\mathrm{B}$. Again, since this is an $N \times N$ matrix, the computation of the eigenvalues of $\mathbf{S}_\mathrm{W}^{-1}\mathbf{S}_\mathrm{B}$ is computationally inefficient. Hence, rather than directly computing the eigenvectors, we first project the memory set data onto a lower dimensional space using eigenfaces.

If we let $\bar{\mu}_i = \mu_i - \mathbf{I}_\mathrm{avg}$, $\mathbf{A} = [\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_7]$, $\mathbf{B} = [m_1\bar{\mu}_1, m_2\bar{\mu}_2, \ldots, m_7\bar{\mu}_7]$, and $\mathbf{S}_\mathrm{B} = \mathbf{B}\mathbf{A}^\mathrm{T}$, then we can give an effective measure of $\mathbf{S}_\mathrm{W}$ and $\mathbf{S}_\mathrm{B}$ in the eigenspace coordinates:

$$\bar{\mathbf{S}}_\mathrm{W} = \sum_{i=1}^{7} \mathbf{W}_\mathrm{pca}^T \mathbf{A}_i \mathbf{A}_i^\mathrm{T} \mathbf{W}_\mathrm{pca}$$

and

$$\bar{\mathbf{S}}_\mathrm{B} = \mathbf{W}_\mathrm{pca}^T \mathbf{B}\mathbf{A}^\mathrm{T} \mathbf{W}_\mathrm{pca}$$

where $\mathbf{W}_\mathrm{pca}$ is the eigenfaces linear projection operator. Finally, the Fisher projection is computed as the eigenvectors of $\bar{\mathbf{S}}_\mathrm{W}^{-1}\bar{\mathbf{S}}_\mathrm{B}$.

$$\mathbf{W}_\mathrm{fld} = \mathrm{eig}(\mathbf{S}_\mathrm{W}^{-1}\mathbf{S}_\mathrm{B})$$

Once the projection matrix is computed and just like in the eigenfaces method, the Fisher projection matrix is used to project each of the database images $\mathbf{I}_j$ onto an M' -dimensional vector of $\mathbf{w}_j$ Fisher coefficients.

III. Classifiers

In this paper, we use two methods for mapping the feature vector into a pose class: nearest neighbor and a hybrid nearest neighbor-neural network approach. The hybrid approach uses a nearest neighbor (and max–min) computation to determine the best matching patterns and, then, a neural network to map these distances to the appropriate class.

A. Nearest Neighbor

Here, we assume that a representation has been chosen and applied to all database images, yielding a database of feature vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M$. In the nearest neighbor classifier, for a given input image $\mathbf{x}$, we first compute the corresponding representation: $\mathbf{w}$ (Fisher or eigen coefficient vector). Then, we compute the distance between $\mathbf{w}$ and each of the stored database feature vectors:

$$\begin{aligned} \delta_1 &= \|\mathbf{w} - \mathbf{w}_1\| \\ \delta_2 &= \|\mathbf{w} - \mathbf{w}_2\| \\ &\cdots \\ \delta_M &= \|\mathbf{w} - \mathbf{w}_M\|. \end{aligned}$$

Finally, we choose the minimum such distance and output the pose associated with the closest database vector.

For convenience, we normalize the resulting distances as follows. First, we compute the minimum distance between $\mathbf{w}$ and each pose class $i = 1, 2, \ldots, 7$:

$$d_i = \min\{\delta(\mathbf{w}, \mathbf{w}_j) : \mathbf{w}_j \text{ is in class } i\}.$$

Then, we normalize these minimum values to be in the range $0 \le d_i \le 1$ by dividing by the sum:

$$\bar{d}_i = \frac{d_i}{d_\mathrm{sum}}$$

where $d_\mathrm{sum} = d_1 + d_2 + \cdots + d_7$. Of course, in the nearest neighbor classifier, choosing the smallest distance is the equivalent to choosing the smallest $\bar{d}_i$.

B. Max–Min Classifier

The min–max classifier works as follows: We compute the maximum distance to each class:

$$D_i = \max\{\delta(\mathbf{w}, \mathbf{w}_j) : \mathbf{w}_j \text{ is in class } i\}$$

and then select the minimum among these maximum distances. Note that the maximum distances can be normalized in the same way: $\overline{D}_1 = D_i/D_\mathrm{sum}$ and $D_\mathrm{sum} = D_1 + D_2 + \cdots + D_7$. Preliminary results showed that the max–min classifier gave poor results for the pose estimation problem. However, the next section shows that a neural network can be used to combine both the smallest $\bar{d}_i$ and largest $\overline{D}_i$ distances to achieve a high-performance decision rule.

C. Hybrid Neural Network Classifier

The nearest-neighbor classifier chooses the smallest distance among all the $\bar{d}_i$ normalized distances, and the max–min classifier chooses the smallest among all the $\overline{D}_i$ distances. It is conceivable, although, that there may be other and more optimal strategies of combining all of the $\bar{d}_i$ and $\overline{D}_i$ information to determine the output pose. To this end, we propose using a feedforward neural network to learn such a mapping.

The structure of the proposed neural network is shown in Fig. 5. In this case, the input to the neural net consists of all the $\bar{d}_i$ and $\overline{D}_i$ values for all seven classes. The output layer consists of seven nodes, and each node signifies one of the classes. Therefore, for example, **Pose 3** would be indicated by the output: $\mathbf{y} = [0, 0, 1, 0, 0, 0, 0]^\mathrm{T}$.

In order to determine the network weights ν_{ji} and w_{kj}, a separate training set of data is needed (separate from the

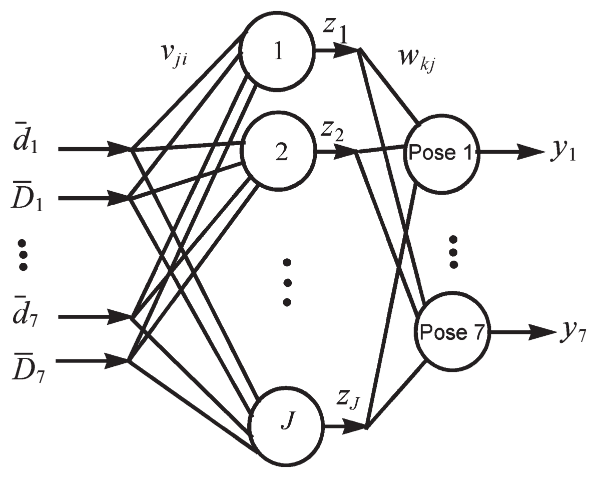

Fig. 5. Architecture of the neural net classifier which maps minimum and maximum distances to output pose.

data used to compute the eigen or Fisher representation). The training phase for the neural network proceeds as follows. First, before training begins, we compute the eigen or Fisher representation for each image in the given database. Then, for each pattern in the training set, we compute the following:

1) the appropriate representation $\mathbf{w}$;
2) normalized min and max distances $\bar{d}_i$ and $\overline{D}_i$, $i = 1, 2, \ldots, 7$;
3) the neural network output $\mathbf{y}$.

The backpropagation algorithm is then used to update all the network weights so that sum-squared error between the neural net output $\mathbf{y}$ and the target output $\mathbf{t}$ is minimized. The adjustable parameters associated with the neural net design are the number of hidden units J and the number of training cycles. J must be chosen large enough so that the network can learn the required mapping; however, choosing J too large can result in overfitting problems where the network achieves low error on the training set but is not able to generalize well for new patterns.

IV. Decision and Postprocessing Stage

A. Biasing the Classifier Output

For the driver pose estimation problem, we expect (and hope!) the driver to be in **Pose 3** most of the time (looking straight at the road ahead). Hence, most of the frames in the training and test set will have truth of **Pose 3**.

In order to account for the fact that not all of the poses are equally probable, we modify the distances computed in the nearest neighbor classifier by including a bias term in the computation of the normalized distances:

$$\bar{d}_i = \frac{d_1}{d_{\text{sum}}} - \alpha_1$$
$$\bar{d}_2 = \frac{d_2}{d_{\text{sum}}} - \alpha_2$$
$$\cdots$$
$$\bar{d}_7 = \frac{d_7}{d_{\text{sum}}} - \alpha_7.$$

Each bias term can be used to decrease the distance according to the *a priori* probability of each pose class. For this application, the most critical bias is that of **Pose 3**. Hence, we will set all other bias terms to zero: $\alpha_i = 0$ for all $i \neq 3$. The remaining bias parameter $\alpha_3 = \alpha$ must be determined based on the available training data.

Clearly, when $\alpha = 0$, no bias is used, and we just have the nearest neighbor rule. Let us denote the correct classification rate of the nearest neighbor rule by P_{NN}. When $\alpha = \infty$, the classifier output will be constant and give **Pose 3** for every test image. In this case, the classification rate will be determined by the probability of a test image being in **Pose 3**. Let us denote this probability by P_3.

What happens for intermediate values of α? We have found empirically that intermediate values of α can yield larger values than both P_{NN} and P_3.

A bias term can also be used for the neural network classifier. In this case, we apply the bias at the output of the neural network rather than at the input. Hence, with the bias term, the output of each unit is modified as follows:

$$\bar{y}_i = y_i + \alpha_i, \quad i = 1, 2 \ldots, 7.$$

B. Sequence Classification Problem

The previous discussion has ignored the temporal aspect of the problem and focused on how to classify each frame of the video (individually and independently). However, the frames of the video are not independent, and the temporal aspect of the problem may be used in order to design more efficient and practical pose estimation systems. As mentioned earlier, one way to incorporate temporal information is to use motion detection or some other preprocessing to segment the video into regions of constant pose. Once this preprocessing is done, then the classification becomes one of mapping the sequence to the proper pose.

1) Sequence Classification Problem: Given constant pose video sequences, classify each sequence into one of the seven standard pose classes.

Designing a system to map the sequence directly to pose class is problematic because each sequence may contain a different number of frames. One simple way to classify a sequence is to first apply a frame-by-frame classification using the previous nearest neighbor or neural net classifier and then combine the outputs using some decision rule to obtain the class of the sequence. The simplest decision rule is to use a majority rule and simply choose the pose which is most prevalent among all the classified frames.

Of course, the performance of the sequence classifier now depends on the performance of the motion detection preprocessor. In this paper, we will assume that the preprocessor operates perfectly and produces the precise sequences, as indicated in the ground truth.

V. Measures of System Performance

There are several measures which can be used to assess the performance of the pose estimation system. For example,

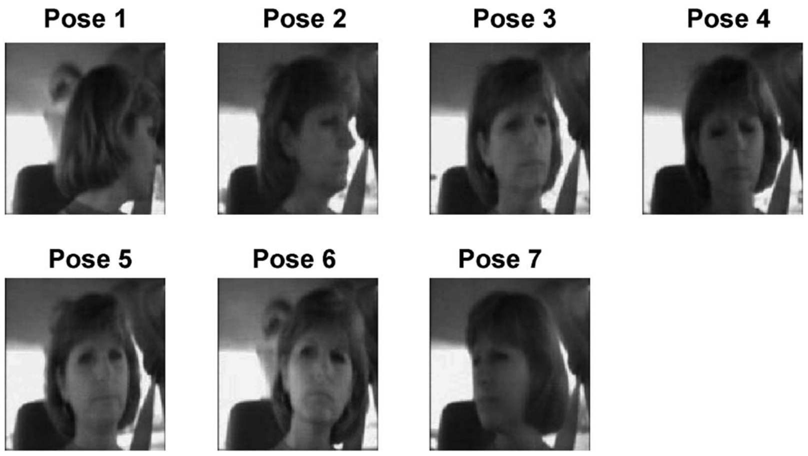

Fig. 6. Sample poses from Subject 11.

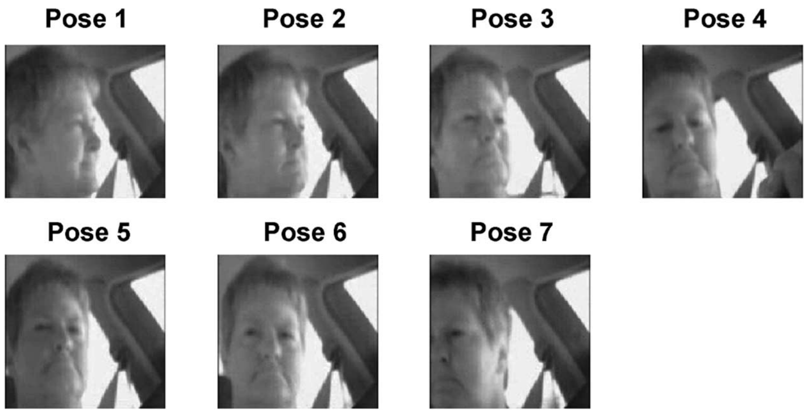

Fig. 7. Sample poses from Subject 46.

we can consider the percent correct classification for each individual pose:

$$P_i = \frac{\#\text{Correct Pose}_i \text{ frames}}{\#\text{Total Pose}_i \text{ frames}}.$$

In addition, we can look at the overall correct classification rate of the individual frames over all the poses:

$$P_{\text{frame}} = \frac{\#\text{Correct frames}}{\#\text{Frames total}}.$$

It is expected that the system will make errors in the transition regions between poses. This is not surprising since even human operators have difficulty choosing a precise demarcation between poses when truthing the video. Hence, we will compute the P_i and P_{frame} measures in three ways:

1) over all the available testing data;
2) over the testing data without any transition regions;
3) just the transition regions.

We assume that the transition region consists of five frames on each side of the pose change. For example, if the pose changes from straight to looking down at frame 100, then the transition region is taken to be frames 95–105. These frames would be excluded in the calculation of P_i and P_{frame} when evaluating system performance in the nontransition regions.

The previous measures can be used for both the frame classification problem and the sequence classification problem. For the sequence classification problem, we can also look at the percent of the segments that were correctly classified:

$$P_{\text{seq}} = \frac{\#\text{sequences correct}}{\#\text{sequences total}}.$$

Oftentimes, the performance of an algorithm depends on several factors, including the number of training samples used, quality of training samples, number of hidden units (for neural net classifiers), etc. Hence, to get a reasonable estimate of how the system performs, we perform several runs of each algorithm and average the results. In addition, to average performance, it is also important to consider the variance of the results. For example, it may be preferable to use a system which offers slightly lower average performance but has a low variance of results, yielding a more reliable system.

VI. Video Data

MPEG videos were obtained from the Transportation Research Center, East Liberty, Ohio. Individual frames from the MPEG video were extracted and cropped to a size of 150×115. Sample poses from the video are shown in Figs. 6 (subject 11) and 7 (subject 46).

TABLE I
NUMBER OF FRAMES IN EACH DATABASE

Database	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8	Total
11-1	190	516	29,782	1,606	1,570	340	316	1,680	36,000
11-2	309	344	24,768	1,513	365	320	290	11,091	39,000
46-1	138	823	22,391	108	899	598	533	265	25,755
46-2	235	556	17,186	102	509	183	492	446	19,710

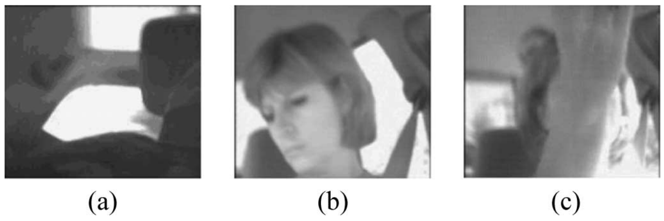

Fig. 8. Sample pose-8 frames.

Two databases for each subject were used, and the number of frames in each database is shown in Table I.

The ground truth was obtained by hand-classifying each frame of the video. Some of the frames in the video were difficult to classify into any of the standard seven poses. In this case, the frame was labeled Pose 8 (none of the above). All the Pose-8 cases were excluded from the training and test sets. Examples of frames that were classified as Pose 8 are shown in Fig. 8. In Fig. 8(a), the driver is leaning so far forward that she is no longer in the camera's view. In Fig. 8(b), the driver has a clearly discernible pose—looking down at the passenger seat—but this is not one of the seven standard poses. In Fig. 8(c), the driver's face is completely occluded by her hand and arm.

Sometimes occlusions occur that are not as drastic as that shown in Fig. 8(c). It is desired that the system be able to produce reasonable classifications when such minor occlusions occur. In order to study the effect that occlusions have on overall classification, we performed a very conservative hand-classification for one of the databases: the 11-2 database. In this case, all frames in which there was an occlusion—even a minor occlusion—were classified as Pose 8. Using this conservative truthing strategy, the human operator assigned 28% of the frames to Pose 8. In the other three databases, a much more liberal truthing strategy was used, and minor occlusions were included. In these three databases, the number of Pose-8 frames is less than 5%.

VII. EXPERIMENTAL RESULTS

There are a number of parameters which affect the performance of the proposed system: number of Fisherfaces, the bias parameter, memory set selection, number of images per pose in the memory set, etc. In this section, we look at the effect of these parameters on system performance and compare the performance of the nearest-neighbor algorithm with the hybrid neural network.

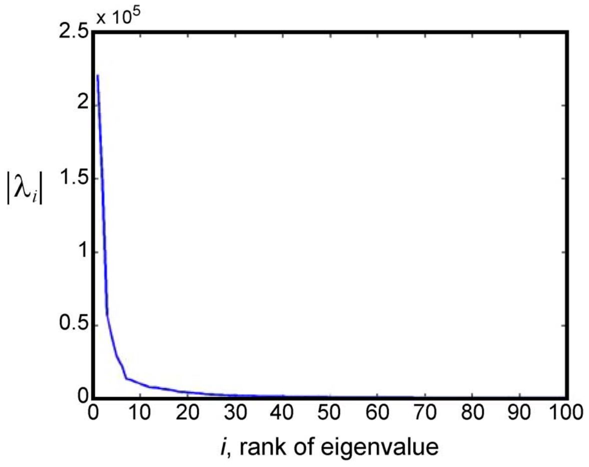

Fig. 9. Plot of the magnitude of the 100 largest eigenvalues.

TABLE II
OVERALL PERCENT CORRECT CLASSIFICATION AS A FUNCTION OF M'

Database	M'	% Correct	Classification Time (msec/frame)
11-1	10	86.3	5.2
	25	87.0	6.6
	50	88.7	9.4
	60	88.8	10.3
	450	89.2	64.5

TABLE III
OVERALL PERCENT CORRECT CLASSIFICATION AS A FUNCTION OF m_i

m_i / m_3	% Correct			Classification Time (msec/frame)
	Overall	Only Transitions	Exclude Transitions	
30 / 90	84.1	65.7	86.1	38.5
40 / 120	87.1	66.1	89.3	56.9
50 / 150	87.4	65.9	89.6	67.4
60 / 180	90.0	66.3	92.5	78.3

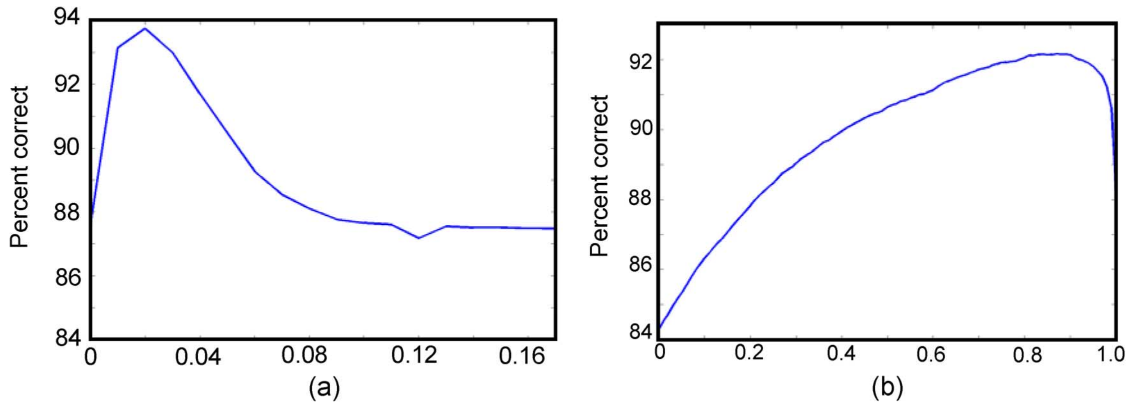

Fig. 10. Effect of the bias parameter α on percent correct classification for (a) the nearest neighbor and (b) the hybrid neural network classifier.

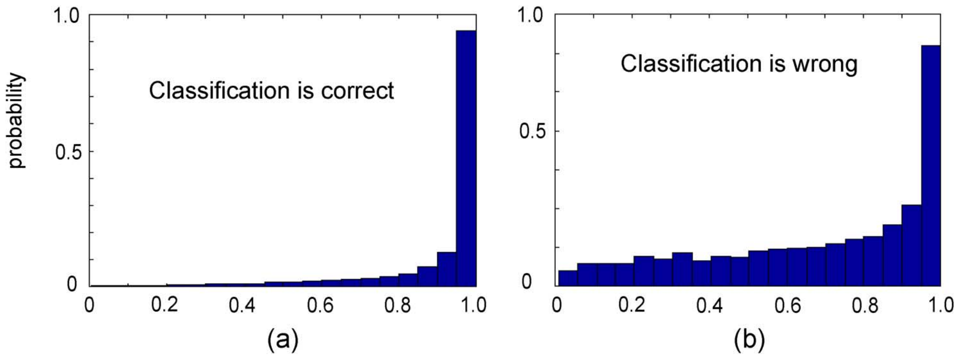

Fig. 11. Histogram of neural net outputs for (a) the winning unit when the classification is correct and (b) the winning unit when the classification is wrong.

A. Number of Fisherfaces

One of the advantages of the Fisher representation is that a small number of Fisherfaces can be used to represent the data set. In this section, we examine the question: How many Fisherfaces should be retained?

Given a database of driver video frames to be classified, suppose we start with a memory set consisting of 50 patterns for each pose, except for **Pose 3**, which contains 150 patterns. These memory set patterns are selected randomly from the given database. In this case, there are a total of 450 different memory set images, and hence, there are 450 different eigenvectors and eigenvalues. Fig. 9 shows a plot of the magnitude of the 100 largest eigenvalues (in descending order). The plot shows an exponential-type decay in magnitude. Since each eigenvalue is related to the amount of variance captured by the corresponding eigenvector (or eigenface), the plot suggests that it is possible to capture much of the variance of the training set by retaining just a few of the eigenfaces with the highest eigenvalue, for example, the top 50.

Classification experiments were run on the 11-1 database for different values of $M' : M' = 10, 25, 50, 75, \text{and} 450$, and the results are shown in Table II. The results show that typically a reduction from $M' = 450$ to $M' = 10$ or 25 reduces the overall percent classification by a few percentage points. The big advantage of using a smaller number of eigenvectors is that the classification time is greatly reduced. For example, using $M' = 10$ requires 5.2 ms to classify a single frame, while using $M' = 450$ requires 64.5 ms. Note that these timing results were

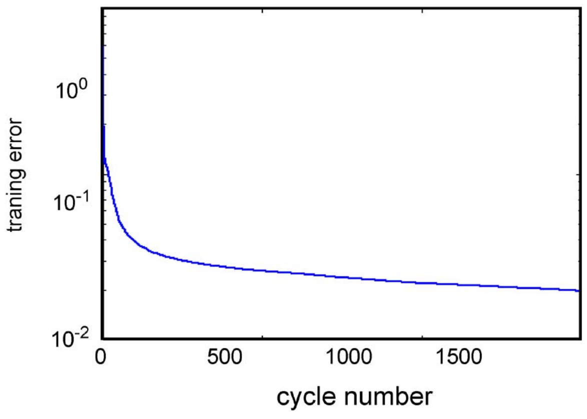

Fig. 12. Neural net training error versus cycle number.

obtained from running the algorithm in Matlab on a 3-GHz PC with a Pentium 4 processor. For a practical system, we want to use as small a value for M' as possible.

B. Number of Memory Images

The question addressed in this section is: How many training samples are needed in the memory set in order to achieve good classification performance? Remember that it is expected that the driver will be in **Pose 3** most of the time. Hence, it is natural to use more training samples for **Pose 3** than the other poses. We will use a uniform number of memory images for each of

TABLE IV
PERCENT CORRECT CLASSIFICATION RATE FOR THE HYBRID NEURAL NETWORK CLASSIFIER ON THE FOUR DATABASES. THE RESULTS WERE AVERAGED OVER TEN RUNS

Database	Test Set	Overall frame rate					
		Nearest neighbor			*Hybrid Neural Net*		
		ave	**std**	**best**	**ave**	**std**	**best**
11-1	Overall	87.8	0.97	89.3	93.4	0.67	94.6
	Transition Regions	66.8	1.2	68.5	68.5	1.4	70.2
	Exclude Transitions	90.1	1.0	91.6	96.1	0.64	97.2
11-2	Overall	94.0	0.42	94.7	95.8	0.41	96.1
	Transition Regions	76.0	1.4	77.7	76.3	1.78	78.4
	Exclude Transitions	95.6	3.9	96.3	97.6	0.41	98.0
46-1	Overall	85.1	1.1	86.9	92.6	0.80	93.6
	Transition Regions	61.4	1.7	63.1	62.0	1.4	63.8
	Exclude Transitions	87.6	1.1	89.4	95.7	0.89	97.0
46-2	Overall	90.8	0.76	91.9	94.9	0.79	95.8
	Transition Regions	66.1	0.99	67.3	67.4	1.1	69.6
	Exclude Transitions	93.3	0.75	94.3	97.6	0.79	98.6

the nonstraight poses: $m = m_i$, $i \neq 3$, and triple that amount for **Pose 3**: $m_3 = 3$ m.

Table III gives the classification performance when m varies from 30 to 60 (and, hence, m_3 varies from 90 to 180). As expected, as the size of the memory set increases, classification performance increases as well. The disadvantage of using a large memory set is that classification time becomes slower. As shown in Table III, the amount of time (on a 3-GHz PC) to classify a single frame increases linearly from 38.5 ms when the memory set contains 30/90 images per pose to 78.3 ms when the memory set contains 60/180 images per pose. It is desired that the system operates as close to real time as possible $(30 \text{ fps} = 33.3 \text{ ms/frame})$. Hence, even though better classification results can be obtained by using the largest memory set possible, we will use the 50/150 memory set for all remaining simulations.

C. Bias Parameter

Fig. 10(a) shows how the probability of correct classification varies with the bias parameter α for the nearest neighbor classifier on the 11-1 database. From this figure, it is possible to determine an optimal bias value of $\alpha^* = 0.02$. With this value of the bias, the classification rate is raised from about 88% (the simple nearest-neighbor rule with $\alpha = 0$) to about 94%.

Fig. 10(b) shows how the correct classification rate varies with α for the hybrid neural network. In this case, there is a steady increase in performance and a rapid falloff near $\alpha = 0.95$. An optimal bias here would be about $\alpha^* = 0.92$.

Recall that each neural net output is bounded between 0 and 1. Hence, at first glance, it seems odd that such a large α is needed for the neural network. The reason for this can be seen in the plots shown in Fig. 11. Fig. 11(a) shows a plot of a histogram of the neural net output for the winning (largest) unit when the winning unit correctly classifies the input image. As expected, the output is near 1. Fig. 11(b) shows a histogram of the output of the winning unit when the classification was not correct (i.e., the pattern was misclassified). Notice that these values are also tightly clustered around 1. Hence, to correct these errors requires a large bias.

D. Neural Net Versus Nearest Neighbor

To compare the performance of the nearest-neighbor and neural net classifier, we proceeded as follows. First, we selected a training set with 50 patterns for each pose, except for **Pose 3**, which contained 150 patterns. Then, we computed the Fisherface representation (which was used by both the nearest neighbor and neural net). Note that only the top 50 Fisherface coefficients were retained; hence, the feature vector representing each image was 50-D.

The neural net required a separate training set, and we used the same size for the training set as the memory set

TABLE V
PERCENT CORRECT CLASSIFICATION RATE FOR THE NEAREST-NEIGHBOR CLASSIFIER ON THE FOUR DATABASES. THE RESULTS WERE AVERAGED OVER TEN RUNS

Database	**Test Set**	**% Correct Classification on Individual Poses (Ave)**						
		P_1	P_2	P_3	P_4	P_5	P_6	P_7
11-1	Overall	81.6	78.7	89.6	78.5	67.3	78.8	76.7
	Transition Regions	69.3	53.1	74.9	60.7	47.3	64.1	49.1
	Exclude Transitions	99.2	91.4	90.5	86.3	77.1	90.8	92.3
11-2	Overall	57.9	89.7	95.9	85.2	88.3	34.7	82.9
	Transition Regions	27.9	80.5	85.1	65.9	81.9	17.2	46.9
	Exclude Transitions	70.1	99.1	96.5	90.7	98.1	42.1	96.7
46-1	Overall	68.1	63.4	87.0	69.8	67.0	75.8	71.5
	Transition Regions	65.8	45.6	69.9	64.3	55.8	53.3	51.1
	Exclude Transitions	100	78.0	88.0	100	78.6	85.4	85.0
46-2	Overall	77.0	71.4	92.4	65.0	71.5	79.0	74.8
	Transition Regions	60.1	52.1	73.8	63.3	53.4	75.8	55.3
	Exclude Transitions	93.3	90.8	93.5	100	86.3	98.3	86.7

(50 samples per pose and 150 samples of **Pose 3**). We found that setting $J = 30$ provided sufficiently many hidden neurons to learn the desired mapping. Also, we limited training to 1500 cycles because we found that the training error was typically sufficiently small at that point. A typical plot of how the training error decreases as training proceeds is shown in Fig. 12. Note that, ideally, one would like to monitor neural net training with a separate validation set, which would provide more information as to when to stop training. The driver video databases that we used, however, did not have enough training data (other than **Pose 3**) to use a separate validation set. Also, note that both the neural net and the nearest neighbor were tested on the same test set and that the test set excluded all of the training and memory patterns.

From the trial simulations on the 11-1 database and as indicated in Fig. 10, the following bias parameters were used: $\alpha = 0.02$ for the nearest-neighbor classifier and $\alpha = 0.90$ for the hybrid neural network. The same bias parameters were used for all the other databases without any further refinement or training.

The classification experiment was repeated ten different times for each database, each time with a different (randomly selected) memory set. The overall classification results P_{frame} are shown in Table IV for both the nearest-neighbor algorithm (with bias) and the hybrid neural network (with bias). The results are reported both with and without the transition regions in the test set. As expected, the system performs rather poorly in the transition regions.

In general, the hybrid neural network outperforms the nearest neighbor classifier because it achieves a higher classification rate. Both algorithms offer relatively stable performance in that the standard deviation is typically less than 1% over the ten runs.

The classification results pertaining to each individual pose P_i are shown in Table V for the nearest-neighbor classifier and Table VI for the hybrid neural network classifier. As before, each result is averaged over ten trials.

For the 11-1 database, the best result over the ten runs yielded a 94.6% correct classification rate. In this case, 5.4% of the frames were misclassified. For this best result, Table VII gives an indication as to what types of errors were made. The first column of the table shows how the system classified all **Pose 1** frames. Here, 65.6% were correctly classified as **Pose 1** (the main diagonal gives the percent correct classification for each pose); however, 5.6% were misclassified as **Pose 2**, 28.8% were misclassified as **Pose 3**, etc.

E. Sequence Classification Problem

The goal here is to classify sequences of data at a time rather than individual frames. Here, we assume that a preprocessing is done to the video data in order to partition the frames into

TABLE VI
PERCENT CORRECT CLASSIFICATION RATE FOR THE HYBRID NEURAL NETWORK CLASSIFIER ON THE FOUR DATABASES. THE RESULTS WERE AVERAGED OVER TEN RUNS

Database	**Test Set**	**% Correct Classification on Individual Poses (Ave)**						
		P_1	P_2	P_3	P_4	P_5	P_6	P_7
11-1	Overall	69.4	66.0	98.2	63.5	46.9	65.6	66.4
	Transition Regions	53.1	29.5	93.8	38.7	23.8	44.0	33.8
	Exclude Transitions	93.0	84.0	98.5	74.5	58.4	83.3	84.8
11-2	Overall	56.7	78.1	98.6	80.2	70.5	36.5	75.6
	Transition Regions	28.4	59.6	92.4	58.8	55.0	15.4	37.6
	Exclude Transitions	68.1	97.1	99.0	86.3	94.3	45.4	90.2
46-1	Overall	38.1	49.6	97.1	46.5	49.9	54.9	64.8
	Transition Regions	33.8	29.4	89.5	37.3	35.5	27.7	40.5
	Exclude Transitions	96.7	66.1	97.6	97.3	64.8	66.5	81.7
46-2	Overall	60.1	63.0	97.6	50.0	63.3	51.7	71.1
	Transition Regions	33.5	41.2	87.7	47.6	41.6	44.7	46.2
	Exclude Transitions	85.7	84.8	98.2	100	80.9	93.8	86.3

TABLE VII
RESULTS OF THE HYBRID NEURAL NET ON THE SUBJECT 11 VIDEO. OVERALL CORRECT CLASSIFICATION RATE: 94.6%

Database 11-1		**True Pose**						
		#1	#2	#3	#4	#5	#6	#7
Classified Pose	#1	65.6	0	0	0	0	0	0
	#2	5.6	74.5	0.2	0	0.1	2.1	4.6
	#3	28.8	25.5	98.8	25.4	48.8	38.3	24.1
	#4	0	0	0.4	73.4	1.1	0	0
	#5	0	0	0.5	1.1	50.0	2.1	1.4
	#6	0	0	0.1	0.1	0	56.7	2.3
	#7	0	0	00	0	0	0.8	67.6

sequences of constant pose. Since we did not implement such a motion preprocessor, we used a best-case scenario and assumed that an ideal motion detector was able to partition the given database into exactly the same sequences as our ground truth. We assume that the motion occurs in the five frames on either side of the pose transition.

For example, for the 11-1 database, the ground truth indicates that for frames 1–50, the driver is in **Pose 3**, and for frames 51–84, the driver is in **Pose 4**. Hence, the ideal motion detector would indicate that in frames 5–45, the pose was constant (no motion) and that motion occurred between frames 46–54 (as the driver changed pose from 3 to 4). In the next segment, frames 55–79 would be considered constant pose, and frames 80–89 would be marked as motion frames.

As described in Section IV-B, the sequence classification strategy involves classifying each frame in the given sequence and then combining the results in order to assign a pose class characterizing the entire sequence. Let n_i be the number of frames in the sequence that were classified as pose i, $i = 1, 2, \ldots, 7$. We use a simple plurality rule combination

TABLE VIII
PROBABILITY OF CORRECT CLASSIFICATION FOR INDIVIDUAL FRAMES AND FOR SEGMENTS

Database	**Test Set**	**Overall frame rate**					
		Nearest neighbor			*Hybrid Neural Net*		
		ave	**std**	**best**	**ave**	**std**	**best**
11-1	P_{seq}	88.3	9.7	95.0	90.3	11.3	98.3
	P_{frame}	92.4	9.8	97.4	92.0	9.5	98.7
11-2	P_{seq}	92.7	8.2	98.5	92.8	8.2	98.6
	P_{frame}	91.3	10.5	98.8	91.6	10.1	98.7
46-1	P_{seq}	89.3	9.5	96.0	92.2	0.4	98.8
	P_{frame}	89.1	7.7	94.5	89.4	12.5	98.2
46-2	P_{seq}	90.6	10.2	97.8	94.4	7.2	99.1
	P_{frame}	90.9	9.7	97.8	93.3	8.3	99.2

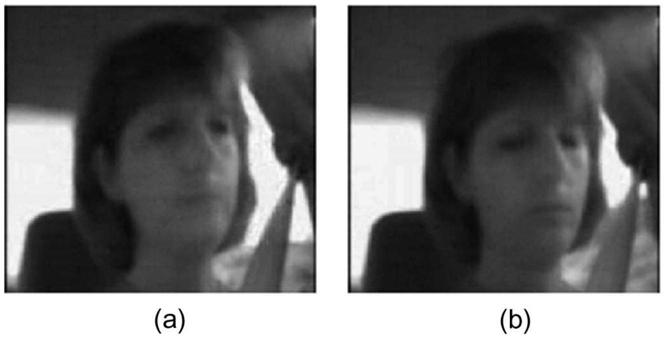

Fig. 13. Two frames of subject 11 showing (a) **Pose 3** and (b) **Pose 4**. These two poses are very difficult to distinguish even for a human operator.

strategy. That is, the sequence is classified as class i^* when that class appeared most often in the individual frames: $n_{i^*} = \max\{n_1, n_2, \ldots, n_7\}$.

Table VIII gives the constant pose sequence classification rate P_{seq} and the frame classification rate P_{frame} for the four databases after such an ideal motion detector is used. Notice that the variance of results is much higher here than the frame classification results.

VIII. DISCUSSION

A thorough analysis of the classifier outputs reveals that the major sources of error are the following.

1) Indistinguishable poses—In some instances, not all seven poses are clearly distinguishable from each other. Fig. 13 shows two different facial poses (**Poses 3** and **4**) of subject 11. Notice the visual similarity between the two images, and indeed, they are both classified as belonging to **Pose 3**. We believe a feasible solution to eliminate such errors would be to more carefully select the training samples and/or select a region of interest within the image to do the training and the classification.

2) Transition poses—When a subject makes a transition from one pose to another, there are frames in the interim that may really not belong to either one of the poses, or it may not even belong to any of the seven standard poses. A case in point is the transition from **Pose 3** to **Pose 7**, where one or two of the intermediate frames could belong to **Pose 6**. Fig. 14 shows a **Pose 3** to **Pose 7** transition sequence. This problem is further compounded by the fact that very few of the images used to train the classifier are in these transition regions. A possible solution to this problem is explicit inclusion of the so-called transition poses in both the training and classification phases.

3) Incorrect true pose—When videos are truthed by a human operator, certain types of errors are introduced. Going back to the transition pose problem, the classifier error rate is dependent on where (on which frame) the human operator chose to separate the two poses. In addition, pose transitions that are of very small duration can (and often are) be completely missed. Such errors and others were found to be present in the ground truth produced by two different people. Using the existing ground truth as a starting point, the video can be truthed again to eliminate errors.

4) Image noise—Template matching approaches are sensitive to changes in illumination and background. Preprocessing measures (for example, histogram equalization, cropping, etc.) can be used to try to mitigate these types of noise. In addition, one can populate the memory set with a rich set of prototypes, which are covering, as much as possible, instances of the input noise that the system is expected to handle. As mentioned previously,

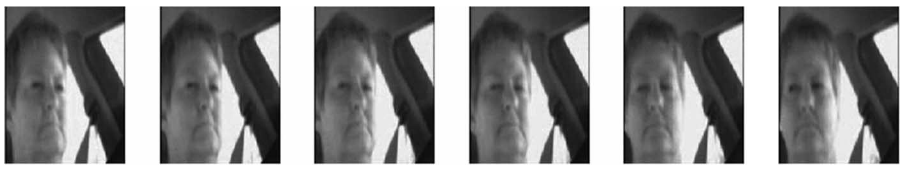

Fig. 14. Sequence of frames which shows subject 46 moving from **Pose 3** to **Pose 7**. The ground truth for this sequence was determined to be [3, 3, 3, 3, 7, 7], and the classifier output for this sequence was [3, 3, 3, 6, 7, 7].

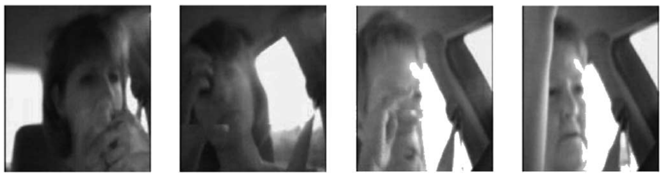

Fig. 15. Sample images of subjects 11 and 46 showing various types of occlusions.

the size of the memory set has to be chosen as the result of a compromise between accuracy and computation time.

5) Wrong pose—Movement of the subject's head can cause large changes in both the position and/or the brightness values of facial pixels. In addition, movements of the subject's hands to the face cause occlusions. Fig. 15 shows several examples. There is no obvious method to overcome such errors.

IX. Summary

The results in this paper show that the Fisherface representation shows much promise in terms of solving the difficult problem of driver facial pose classification. Both the nearest neighbor and the hybrid neural net classifiers provide favorable results across a wide spectrum of images.

In a practical implementation of a pose estimation system, a troubling problem to be solved is "How is the training data to be collected?" One approach is for a human user to classify the first few frames by hand. However, some of the poses are not encountered until 1000 of frames into the video. What is needed is a system confidence measure, and when the confidence falls lower than a specified threshold, then the system will just add that pattern to the database and ask the user to provide the needed ground truth. In future work, we will devise schemes for such a practical implementation.

In the present procedure, the training images are randomly selected from all available video frames. Naturally, the more training images used in the training set, the better the classifier will perform; however, the higher dimensional feature vector increases the classification time. A method is needed whereby the training images are chosen—not randomly—but to maximize the discrimination ability of the classifier [25]. In future work, we will investigate incremental and adaptive strategies to systematically choose just the "right" number of training images.

Finally, the present system requires that both the memory and test set contain images of the same driver. An obvious needed extension is the design of a system that is driver independent. We plan to develop such systems and test whether the system trained with the images of one driver can be used to reliably classify test sets of many different drivers.

Acknowledgment

The authors would like to thank Prof. N. Natarajan for his suggestions and help with this project.

References

[1] F. Barickman and M. Goodman, "Micro-DAS: In vehicle portable data acquisition system," *Transp. Res. Rec.*, 1999. Paper 99-0611.
[2] P. Batavia, "Driver adaptive warning system," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep.-CMU-RI-TR-98-07, 1998.
[3] P. Belhumeur, J. Hespanha, and D. Kreigman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.
[4] J. Chang and J. Chen, "A facial expression recognition system using neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1999, vol. 5, pp. 113–120.
[5] R. Chellappa, C. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proc. IEEE*, vol. 83, no. 5, pp. 705–740, May 1995.
[6] M. Er, S. Wu, J. Lu, and H. Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 697–710, May 2000.
[7] A. Gee and R. Cipolla, "Determining the gaze of faces in images," *Image Vis. Comput.*, vol. 12, no. 10, pp. 639–647, 1994.
[8] S. Gutta and H. Wechsler, "Network ensembles for facial analysis tasks," in *Proc. IEEE-INNS-ENNS IJCNN*, 2000, vol. 3, pp. 305–310.
[9] S. Ho and H. Huang, "An analytic solution for the pose determination of human faces from a monocular image," *Pattern Recognit. Lett.*, vol. 19, no. 11, pp. 1045–1054, 1998.
[10] J. Huang, X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines," in *Proc. 14th Int. Conf. Pattern Recog.*, 1998, pp. 154–156.
[11] L. Koh, S. Ranganath, and Y. Venkatesth, "An integrated face detection and recognition system," *Pattern Recognit.*, vol. 35, no. 6, pp. 1259–1273, 2002.
[12] N. Kruger, M. Potzsch, and C. von der Malsburg, "Determination of face position and pose with a learned representation based on labelled graphs," in *Proc. Image Vis. Conf.*, 1997, pp. 665–673.
[13] S. Lakshmanan, P. Watta, Y. Hou, and N. Gandhi, "Comparison between eigenfaces and fisherfaces for estimating driver pose," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Oakland, CA, Aug. 25–29, 2001, pp. 889–894.
[14] S. McKenna and S. Gong, "Real-time face pose estimation," *Real-Time Imaging*, vol. 4, no. 5, pp. 333–347, 1998.
[15] A. Nikolaidis and I. Pitas, "Facial feature extraction and pose determination," *Pattern Recognit.*, vol. 33, no. 11, pp. 1783–1791, 2000.
[16] in *Proc. 1998 and 2000 IEEE Intell. Vehicles Symp.*
[17] in *Proc. 1997, 1999, and 2000 IEEE Intell. Transp. Syst. Conf.*
[18] R. Rae and H. Ritter, "Recognition of human head orientation based on artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 2, pp. 257–265, Mar. 1998.
[19] J. Sherrah and S. Gong, "Fusion of perceptual cues for robust tracking of head pose and position," *Pattern Recognit.*, vol. 34, no. 8, pp. 1565–1572, 2001.
[20] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 4, no. 3, pp. 519–524, 1987.
[21] L. Tijerina, S. Johnston, E. Parmer, M. Winterbottom, and M. Goodman, "Driver distraction with wireless telecommunications and route guidance

systems," Nat. Highway Traffic Safety Admin., Washington, DC, Rep. DOT HS 809-069, 2000.

[22] M. Turk and A. Pentland, "Face processing: Models for recognition," in *Proc. SPIE Intell. Robots Comput. Vis. VIII: Algorithms Tech.*, 1989, vol. 1192, pp. 22–32.

[23] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[24] L. Tyeruna, M. Gleckler, D. Stoltzfus, S. Johnson, M. Goodman, and W. Wierville, "A preliminary assessment of algorithms for drowsy and inattentive driver detection on the road," Tech. Rep. DOT HS 808, Mar. 1999.

[25] J. Wang, K. Plataniotis, and A. Venetsanopoulos, "Selecting discriminant eigenfaces for face recognition," *Pattern Recognit. Lett.*, vol. 26, no. 10, pp. 1470–1482, 2005.

[26] J. Wang, E. Sung, and R. Venkateswarlu, "Estimating the eye gaze from one eye," *Comput. Vis. Image Underst.*, vol. 98, no. 1, pp. 83–103, 2005.

[27] P. Watta, N. Gandhi, and S. Lakshmanan, "An eigenface approach for estimating driver pose," in *Proc. 3rd Annu. IEEE Conf. Intell. Transp. Syst.*, Dearborn, MI, Oct. 1–3, 2000, pp. 376–381.

Paul Watta (S'91–M'93) received the Bachelor's, Master's, and Ph.D. degrees in electrical engineering from Wayne State University, Detroit, MI, in 1987, 1988, and 1994, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Michigan-Dearborn, Dearborn. His research interests include associative memory, image processing, face recognition, pattern recognition, and computer music.

Sridhar Lakshmanan received the B.S. degree in electronics and communications engineering from the Birla Institute of Technology, Mesra, India, in 1985 and the M.S. and Ph.D. degrees in electrical engineering from the University of Massachusetts, Amherst, in 1987 and 1991, respectively.

In 1991, he joined the faculty at the University of Michigan-Dearborn, Dearborn, as an Assistant Professor of electrical and computer engineering. In May 1997, he was promoted to an Associate Professor, with tenure. His current research interests are in the areas of computer vision, image processing, and pattern recognition. He is also a recipient of several research contracts that address problems in intelligent transportation systems and serves as a consultant to industry and the government on these problems. In 1996, he founded M-Vision, Inc.: a company specializing in the application of computer vision to automotive problems.

Yulin Hou received the B.S. degree in electrical engineering from the National University of Defense Technology, Changsha, China, and the M.S. degree in electrical engineering from the University of Michigan-Dearborn, Dearborn.

He worked in the telecommunications industry for five years in Beijing, China, before joining the University of Michigan-Dearborn as a Graduate Research Assistant. After graduating in 2002, he joined Thermal Waving Imaging, Inc. as an Engineer. His main research interests are data processing and software development.