# Speaker Recognition With Polynomial Classifiers

William M. Campbell, Khaled T. Assaleh, *Senior Member, IEEE,* and Charles C. Broun

***Abstract*—Modern speaker recognition applications require high accuracy at low complexity. We propose the use of a polynomial-based classifier to achieve these objectives. This approach has several advantages. First, polynomial classifier scoring yields a system which is highly computationally scalable with the number of speakers. Second, a new training algorithm is proposed which is discriminative, handles large data sets, and has low memory usage. Third, the output of the polynomial classifier is easily incorporated into a statistical framework allowing it to be combined with other techniques such as hidden Markov models. Results are given for the application of the new methods to the YOHO speaker recognition database.**

***Index Terms*—Discriminative training, polynomial classification, speaker recognition.**

## I. Introduction

SPEAKER recognition is an attractive biometric for implementation in secure access control situations. Voice verification has the advantage that it requires little custom hardware and is nonintrusive. One of the needs of current systems is to obtain reasonable accuracy at low computational and storage complexity. This feature is essential for embedded devices where memory is at a premium. On server-based systems this property increases capacity to handle multiple recognitions simultaneously.

Many structures have been proposed for speaker recognition. The two most popular techniques are statistical and connectionist-based methods. For the former, Gaussian mixture models (GMM) [1] and HMM systems with cohort normalization [2], [3] or background normalization [4], [5] have been the techniques of choice. For connectionist systems, artificial neural networks have been commonly used; e.g., neural tree networks and multilayer perceptrons [6]. In this paper, we propose a new text-independent speaker recognition system based upon a discriminatively-trained polynomial classifier.

Polynomial networks have been known in the literature for many years [7], [8]. Polynomials have excellent properties as classifiers. Because of the Weierstrass theorem, polynomial classifiers are universal approximators to the optimal Bayes classifier [9].

Typical methods of training polynomial classifiers have either been based on statistical methods or minimizing a mean-squared error criterion. The focus has been on linear or second degree polynomial classifiers. Both of these methods traditionally have had limitations. Classical statistical methods estimate the mean and covariance of the speaker data for a parametric model [7]. The drawback of this approach is that out-of-class data is not used to maximize performance. Discriminative methods usually involve large matrices which leads to large intractable problems [10].

We propose several new methods for using polynomial classifiers for speaker recognition. First, we propose a new training method. The method is discriminative, easily handles large datasets, has low memory usage, and computationally separates the training on a per class basis. Second, we propose a new scoring method. This scoring method makes the recognition problem highly computationally scalable with the number of speakers. Third, we show how the training framework is easily adapted to support new class addition, adaptation, and iterative methods.

Section II describes the structure of the polynomial classifier. Section III illustrates the new training algorithm. We discuss a basic training algorithm and how to reduce computational complexity. Section IV discusses the statistical interpretation of the classifier output, how to perform fast scoring, and prior compensation methods. Section V discusses extensions to the basic methods for adaptation, iterative training, etc. Section VI applies the method to the YOHO database for speaker recognition and gives both identification and verification results.

## II. Classifier Structure

The basic scenario for verification is as follows. An identity claim is made by a speaker (the claimant). The model for the claimant is then retrieved. Speech is obtained from claimant. Feature extraction is performed on the speech. The feature vectors are passed through a classifier to produce a score. A decision is then made based upon whether the score is above or below a threshold.

Similarly, (closed set) identification is performed using the following steps. The speaker is recorded saying an utterance. The utterance is then processed through the system and scores are generated from all speaker models in the system. The speaker whose model scores the highest is the predicted speaker.

For both identification and verification, we perform *text-independent* recognition. That is, we do not assume knowledge of the words being said when we train the classifier or perform recognition. Other methods of recognition are text-prompted (assuming a known prompted text) or text-dependent (e.g., password based).

The basic embodiment of this strategy for a single speaker model is shown in Fig. 1. The classifier consists of several parts. Feature vectors, $\mathbf{x}_1, \dots, \mathbf{x}_N$ are introduced into the classifier. The vector $\mathbf{p}(\mathbf{x})$ is the vector of polynomial basis terms of the

Manuscript received May 22, 2001; revised February 21, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Philip C. Loizou.

W. M. Campbell and C. C. Broun are with Motorola Labs, Tempe, AZ 85284 USA (e-mail: Bill.Campbell@motorola.com).

K. T. Assaleh is with Conexant Systems, Newport Beach, CA 92660 USA.

Publisher Item Identifier S 1063-6676(02)06032-7.

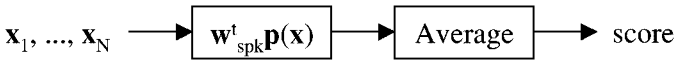

Fig. 1. Polynomial classifier for recognition.

input feature vector; e.g., for a two-dimensional (2-D) feature vector, $\mathbf{x} = [x_1 \; x_2]^t$ and degree two, the vector is given by

$$\mathbf{p}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}^t. \quad (1)$$

In general, the polynomial basis terms of the form

$$x_{i_1}x_{i_2}\dots x_{i_k} \quad (2)$$

are used where $k$ is less than or equal to the polynomial degree $K$, and the $i_j$'s are integers. The speaker model is given by $\mathbf{w}_{\text{spk}}$. For each input feature vector, $\mathbf{x}_i$, a score is produced by the inner product, $\mathbf{w}_{\text{spk}}^t\mathbf{p}(\mathbf{x}_i)$. The score is then averaged over time to produce the final output. The total score is then

$$s = \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}_{\text{spk}}^t\mathbf{p}(\mathbf{x}_i). \quad (3)$$

Viewed in another manner, we are averaging the output of a discriminant function [7] over time. Note that we do not use a sigmoid on the output as is common in higher-order neural networks [11]. Also, note that we use a bold $\mathbf{p}$ to indicate a polynomial expansion; this reduces confusion when we talk about probabilities where a $p$ is used.

For verification, the accept/reject decision for the system in Fig. 1 is performed by comparing the output score (3) to a threshold. If $s < T$, then reject the claim; otherwise, accept the claim. For identification, the operation in Fig. 1 is performed for all speaker models. The highest scoring speaker is selected as the predicted speaker.

## III. Training Method

### A. Basic Algorithm

In order to obtain good separation between classes for the system in Fig. 1, we use discriminative training with a mean-squared error criterion. For the speaker's feature vectors an output of one is desired. For impostor data, an output of zero is desired. For discussion purposes, we consider the two-class problem (i.e., verification). The extension to the multiclass identification will be shown in the algorithm pseudo-code.

Let $\mathbf{w}_{\text{spk}}$ be the optimum speaker model, $\omega$ the class label, and $y(\omega)$ the ideal output, i.e., $y(\text{spk}) = 1$ and $y(\text{imp}) = 0$. The resulting problem using the mean-squared error criterion is

$$\mathbf{w}_{\text{spk}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbf{E}\{(\mathbf{w}^t\mathbf{p}(\mathbf{x}) - y(\omega))^2\} \quad (4)$$

where $\mathbf{E}$ denotes expectation over $\mathbf{x}$ and $\omega$. This criterion can be approximated using the training set as

$$\mathbf{w}_{\text{spk}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{N_{\text{spk}}} |\mathbf{w}^t\mathbf{p}(\mathbf{x}_i) - 1|^2 + \sum_{i=1}^{N_{\text{imp}}} |\mathbf{w}^t\mathbf{p}(\mathbf{y}_i)|^2\right]. \quad (5)$$

Here, the speaker's training data is $\mathbf{x}_1, \dots, \mathbf{x}_{N_{\text{spk}}}$, and the anti-speaker data is $\mathbf{y}_1, \dots, \mathbf{y}_{N_{\text{imp}}}$. (Anti-speakers are a set of speakers randomly selected from the impostor population [6].)

The training method can be written in matrix form. First, define $\mathbf{M}_{\text{spk}}$ as the matrix whose rows are the polynomial expansion of the speaker's data, i.e.,

$$\mathbf{M}_{\text{spk}} = \begin{bmatrix} \mathbf{p}(\mathbf{x}_1)^t \\ \mathbf{p}(\mathbf{x}_2)^t \\ \vdots \\ \mathbf{p}\left(\mathbf{x}_{N_{\text{spk}}}\right)^t \end{bmatrix}. \quad (6)$$

Define a similar matrix for the anti-speaker data, $\mathbf{M}_{\text{imp}}$. Define

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{\text{spk}} \\ \mathbf{M}_{\text{imp}} \end{bmatrix}. \quad (7)$$

The problem (5) then becomes

$$\mathbf{w}_{\text{spk}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, ||\mathbf{M}\mathbf{w} - \mathbf{o}||_2 \quad (8)$$

where $\mathbf{o}$ is the vector consisting of $N_{\text{spk}}$ ones followed by $N_{\text{imp}}$ zeros (i.e., the ideal output).

The problem (8) can be solved using the method of normal equations [12]

$$\mathbf{M}^t\mathbf{M}\mathbf{w}_{\text{spk}} = \mathbf{M}^t\mathbf{o}. \quad (9)$$

We rearrange (9) to

$$\left(\mathbf{M}_{\text{spk}}^t\mathbf{M}_{\text{spk}} + \mathbf{M}_{\text{imp}}^t\mathbf{M}_{\text{imp}}\right)\mathbf{w}_{\text{spk}} = \mathbf{M}_{\text{spk}}^t\mathbf{1} \quad (10)$$

where $\mathbf{1}$ is the vector of all ones. If we define $\mathbf{R}_{\text{spk}} = \mathbf{M}_{\text{spk}}^t\mathbf{M}_{\text{spk}}$ and define $\mathbf{R}_{\text{imp}}$ similarly, then (10) becomes

$$(\mathbf{R}_{\text{spk}} + \mathbf{R}_{\text{imp}})\mathbf{w}_{\text{spk}} = \mathbf{M}_{\text{spk}}^t\mathbf{1}. \quad (11)$$

We also define $\mathbf{R} = \mathbf{R}_{\text{spk}} + \mathbf{R}_{\text{imp}}$.

Our new training method is based on (11). There are several advantages to this approach. First, the matrices $\mathbf{R}_{\text{imp}}$ and $\mathbf{R}_{\text{spk}}$ are fixed in size, i.e., they do not vary with the size of the training data set. Let $M_{\text{terms}}$ equal the number of terms in $\mathbf{p}(\mathbf{x})$, then $\mathbf{R}_{\text{imp}}$ and $\mathbf{R}_{\text{spk}}$ are matrices of size $M_{\text{terms}} \times M_{\text{terms}}$. Second, the computation is partitioned. We can calculate $\mathbf{R}_{\text{spk}}$ and $\mathbf{R}_{\text{imp}}$ separately at costs of $O(N_{\text{spk}}M_{\text{terms}}^2)$ and $O(N_{\text{imp}}M_{\text{terms}}^2)$, respectively. The calculation of these two matrices is the most significant part of computation. Note that $\mathbf{R}_{\text{imp}}$ can be precomputed and stored. Third, the terms in the right-hand side of (11) do not need to be calculated; the terms can be obtained as entries of the matrix $\mathbf{R}_{\text{spk}}$.

Note that using the process in (4) to (11), we could also define a model for the anti-speaker set, $\mathbf{w}_{\text{imp}}$, by defining $y(\text{spk}) = 0$ and $y(\text{imp}) = 1$. Intuitively, this model does not appear to be useful, since we have already incorporated the anti-speaker population into training for the speaker model. In Section IV, we show that this is indeed the case.

One objection to the normal equation method might be that it squares the condition number of the matrix $\mathbf{M}$. Because of this property, it might be thought there would be instability in solving (10). In [13], we show that this is not a difficulty—the

TABLE I
TERM REDUNDANCIES FOR THE MATRIX $\mathbf{R}_{\text{spk}}$

| Degree | Terms in $\mathbf{R}_{\text{spk}}$ | Unique Terms | Ratio |
|---|---|---|---|
| 2 | 8,281 | 1,820 | 4.55 |
| 3 | 207,025 | 18,564 | 11.15 |
| 4 | 3,312,400 | 125,970 | 26.30 |

TABLE II
TRAINING ALGORITHM

```
1) For $i = 1$ to $N_{\text{classes}}$
   2) Let $\mathbf{r}_i = \mathbf{0}$.
   3) For $j = 1$ to $N_i$
      4) Retrieve training vector $j$, $\mathbf{x}_{i,j}$, from class $i$'s training set.
      5) Let $\mathbf{r}_i = \mathbf{r}_i + \mathbf{p}_2(\mathbf{x}_{i,j})$.
   6) Next j
7) Next i
8) Compute $\mathbf{r} = \sum_{i=1}^{N_{\text{classes}}} \mathbf{r}_i$.
9) Expand $\mathbf{r}$ to $\mathbf{R}$. Derive $\mathbf{M}_i^t\mathbf{1}$ from $\mathbf{R}_i$.
10) For all i, solve $\mathbf{R}\mathbf{w}_i = \mathbf{M}_i^t\mathbf{1}$ for $\mathbf{w}_i$.
```

condition number of $\mathbf{R}$ can be reduced several orders of magnitude with a simple diagonal scaling operation.

The matrix $\mathbf{R}_{\text{spk}}$ (and its anti-speaker counterpart) in (11) has many redundant terms. In order to reduce storage and computation, only the unique terms should be calculated. The terms in $\mathbf{R}_{\text{spk}}$ consist exactly of sums of the polynomial basis terms of degree less than or equal to $2K$. We denote the vector of terms of degree less than or equal to $2K$ as $\mathbf{p}_2(\mathbf{x})$. Table I shows the number of unique terms in $\mathbf{R}_{\text{spk}}$ for a twelve-dimensional (12-D) feature vector and different polynomial degrees. As an example, from the table we see that by computing only the *unique* terms of $\mathbf{R}_{\text{spk}}$, we save a factor of approximately 11 in training computation and storage for a degree three polynomial.

Combining all of the above methods results in the training algorithm in Table II. (The expansion in Step 9 is discussed in detail in Section III-C.) Note that we have extended to the general multi-class problem. This training algorithm is not limited to polynomials. One key enabling property is that the basis elements form a semigroup, i.e., the product of two basis elements is again a basis element. This property allows one to compute only the unique elements in the matrix $\mathbf{R}_i$. Another key property is partitioning the problem. If a linear approximation space is used, then we obtain the same problem as in (8). The resulting matrix can be partitioned, and the problem can be broken up to ease memory usage.

Our method of using normal equations to fit data is a natural extension of older methods used for function fitting [12] and radial basis function training [14]. We have extended these methods in two ways. First, we partition the problem by class, i.e., we calculate and store the $\mathbf{r}_i$ for each class separately. This is useful since in many cases we can use the $\mathbf{r}_i$ for adaptation, new class addition, or on-line processing. We also can obtain the right hand side of (11) as a subvector of $\mathbf{r}_i$ which reduces computation significantly as we find each $\mathbf{w}_i$. Second, we have used the semigroup property of the monomials to reduce computation and storage dramatically.

## B. Computing the Polynomial Expansion

Suppose we have the polynomial basis terms of degree $k$, and we wish to calculate the terms of degree $k+1$. Assume that every term is of the form (2) where $i_1 \le i_2 \le \cdots \le i_k$. If we have the $k$th degree terms with end term having $i_k = l$ as a vector $\mathbf{u}_l$, we can obtain the $(k+1)$-th degree terms ending with $i_{k+1} = l$ as

$$\begin{bmatrix} x_l\mathbf{u}_1 \\ x_l\mathbf{u}_2 \\ \vdots \\ x_l\mathbf{u}_l \end{bmatrix}. \tag{12}$$

We can then construct $\mathbf{p}(\mathbf{x})$ as follows. Initialize a vector of one and the first degree terms. Then recursively calculate the $(k+1)$-th degree terms from the $k$th using (12). Concatenate the different degrees to get the final result. As an example, suppose $\mathbf{x} = [x_1 \ \ x_2]^t$ and the degree is three, then

$$\mathbf{p}(\mathbf{x}) = \begin{bmatrix}1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3\end{bmatrix}^t. \tag{13}$$

## C. Structure in $\mathbf{R}$

We now explore more the mapping Step 9 in Table II. The redundancy in $\mathbf{R}$ is very structured. One can plot the index of terms in the vector $\mathbf{p}_2(\mathbf{x})$ versus the index of the terms in the matrix $\mathbf{R}$. We compute the vector $\mathbf{p}(\mathbf{x})$ and the vector $\mathbf{p}_2(\mathbf{x})$ as in Section III-B. We index the elements in $\mathbf{R}$ as a one-dimensional (1-D) array using column major form, i.e., element number one is matrix entry (1, 1), element 2 is matrix entry (2, 1), etc. The resulting plot is shown in Fig. 2 for an eight-dimensional (8-D) feature vector and a polynomial degree of 3. Note the interesting self-similar structure. This structure becomes more detailed as the polynomial degree is increased.

To exploit the "redundancy" in the matrix $\mathbf{R}$, we give a simple method for finding entries in $\mathbf{R}$. We specialize to the case when we map a class's vector $\mathbf{r}_k$ to a matrix structure as in Step 9 in Table II. The matrix $\mathbf{R}_k$ is obtained from a sum of outer products

$$\mathbf{R}_k = \sum_{i=1}^{N_k} \mathbf{p}(\mathbf{x}_{k,i})\mathbf{p}(\mathbf{x}_{k,i})^t. \tag{14}$$

The mapping in Step 9 is based upon the fact that it is not necessary to compute the sum of outer products (14) directly. Instead one can compute the subset of unique entries (i.e., the vector $\mathbf{p}_2(\mathbf{x})$), and then map this result to the final matrix.

A straightforward way to implement the mapping is to precompute an index function. We first label each entry in the matrix $\mathbf{R}_k$ in column major form from 1 to the number of entries. The structure of $\mathbf{R}_k$ is determined by one outer product term from (14), $\mathbf{p}(\mathbf{x})\mathbf{p}(\mathbf{x})^t$. Using a computer algebra package, one

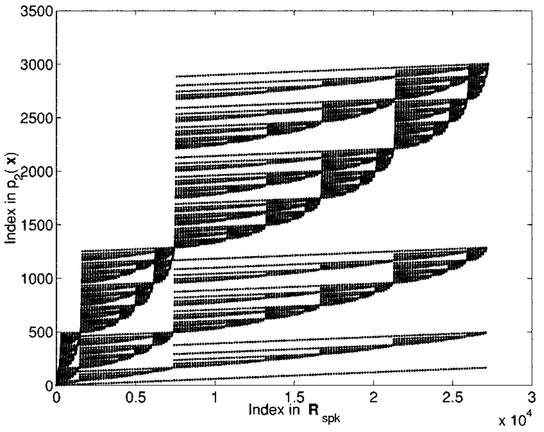

Fig. 2. Plot of index of term in $\mathbf{p}_2(\mathbf{x})$ versus index in $\mathbf{R}_{\text{spk}}$.

can compute the outer product and $\mathbf{p}_2(\mathbf{x})$ with symbolic variables. Then an exhaustive search for each entry of $\mathbf{R}_k$ in $\mathbf{p}_2(\mathbf{x})$ yields the required index map.

The difficulty in using an index function is that the index map must be stored. To avoid this problem, we propose an alternate method based upon a simple property–the semigroup structure of the monomials. Suppose we have an input vector with $n$ variables, $x_1, \ldots, x_n$. The mapping

$$x_{i_1} x_{i_2} \ldots x_{i_k} \mapsto q_{i_1} q_{i_2} \ldots q_{i_k} \tag{15}$$

where $q_{i_j}$ is the $i_j$'th prime, defines a semigroup isomorphism between the monomials and the natural numbers (we map 1 to 1). For example, we map $x_1^2 x_2 = x_1 x_1 x_2$ to $q_1 q_1 q_2 = 2^2 3 = 12$ (2 is the first prime). We can implement the mapping in Table II efficiently using this semigroup isomorphism since it transforms symbol manipulation (monomials) into number manipulation. An algorithm for this mapping is given in Table III. Note that in the table, $[\mathbf{r}]_{i_m}$ is the $i_m$th entry of the vector $\mathbf{r}$, and $[\mathbf{R}]_{i,j}$ is the $(i,j)$th entry of the matrix $\mathbf{R}$.

## IV. Statistical Framework

### A. Statistical Interpretation of Scoring

A polynomial discriminant function $\mathbf{w}_{\text{spk}}^t \mathbf{p}(\mathbf{x})$ with the model from the training process (11) approximates the *a posteriori* probability $p(\text{spk} \,|\, \mathbf{x})$ [10]. In this section, we show how to put this in an optimal decision rule (similar to the artificial neural network literature [15]). We also show how a simple approximation to the standard rule leads to a system that is highly computationally scalable with the number of speakers (for speaker identification).

We use an optimum Bayes approach to recognition. Suppose we have input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$. We first calculate $p(\mathbf{x}_1, \ldots, \mathbf{x}_N \,|\, \text{spk}_j)$, i.e., the probability of the sequence given speaker $j$'s model. We abbreviate this as $p(\mathbf{x}_1^N \,|\, \text{spk}_j)$. A standard assumption in the speech literature is to assume independence of feature vectors [16]; we have

$$p\left(\mathbf{x}_1^N \mid \text{spk}_j\right) = \prod_{i=1}^{N} p(\mathbf{x}_i \,|\, \text{spk}_j). \tag{16}$$

We now use the relation

$$p(\mathbf{x}_i \,|\, \text{spk}_j) = \frac{p(\text{spk}_j \,|\, \mathbf{x}_i) p(\mathbf{x}_i)}{p(\text{spk}_j)} \tag{17}$$

along with the fact that the factor $p(\mathbf{x}_i)$ cancels out of the likelihood decision ratio, to obtain the discriminant function

$$d'\left(\mathbf{x}_1^N, \text{spk}_j\right) = \prod_{i=1}^{N} \frac{p(\text{spk}_j \,|\, \mathbf{x}_i)}{p(\text{spk}_j)}. \tag{18}$$

Note that the prior probability in (18) is the *training* prior. If we equalize the priors in training as in (30), then the quantity $p(\text{spk}_j)$ does not need to be included in the discriminant function (18).

We now perform two simplifications. First, we consider the logarithm of the discriminant function

$$\log\left(d'\left(\mathbf{x}_1^N, \text{spk}_j\right)\right) = \sum_{i=1}^{N} \log\left(\frac{p(\text{spk}_j \,|\, \mathbf{x}_i)}{p(\text{spk}_j)}\right). \tag{19}$$

Second, using Taylor series, a linear approximation of $\log(x)$ around $x = 1$ is $x - 1$. Thus, we can approximate $\log(d'(\mathbf{x}_1^N, \text{spk}_j))$ as

$$\log\left(d'\left(\mathbf{x}_1^N, \text{spk}_j\right)\right) \approx \sum_{i=1}^{N} \left(\frac{p(\text{spk}_j \,|\, \mathbf{x}_i)}{p(\text{spk}_j)} - 1\right). \tag{20}$$

The approximation (20) is especially good when the scaled *a posteriori* is near one. Since $x - 1$ goes to $-1$ as $x$ goes to zero and $\log(x)$ goes to $-\infty$, using $x - 1$ is approximately equivalent to replacing $\log(x)$ by $\max(\log(x), -1)$. The approximation $\max(\log(x), -1)$ is equivalent to ensuring that the probability is not allowed to go below a certain value. The discriminant function with all of the above approximations is

$$d\left(\mathbf{x}_1^N, \text{spk}_j\right) = \frac{1}{N} \sum_{i=1}^{N} \frac{p(\text{spk}_j \,|\, \mathbf{x}_i)}{p(\text{spk}_j)} \tag{21}$$

where we have dropped the $-1$ since a constant offset will be eliminated in a likelihood ratio function. In (21), we have normalized by the number of frames to ensure that a constant threshold can be used in verification. The approximation (21) is the basis for our scalable scoring technique.

A critical feature of our approximation is that it maintains the ordering of top speaker scores, i.e., if $d'(\mathbf{x}_1^N, \text{spk}_j) > d'(\mathbf{x}_1^N, \text{spk}_k)$, then we want $d(\mathbf{x}_1^N, \text{spk}_j) > d(\mathbf{x}_1^N, \text{spk}_k)$ to ensure correct classification. We demonstrate this empirically in the experiments.

Equation (21) gives a computationally scalable system. If we substitute in the model for each speaker, $\mathbf{w}_j$, and assume all priors are equal in (21), we get

$$\begin{aligned} d\left(\mathbf{x}_1^N, \text{spk}_j\right) &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}_j^t \mathbf{p}(\mathbf{x}_i) \\ &= \mathbf{w}_j^t \bar{\mathbf{p}} \end{aligned} \tag{22}$$

TABLE III
MAPPING $\mathbf{r}$ TO $\mathbf{R}$

```
1) Let $\mathbf{q}$ be the vector of the first $n$ primes (where $n$ is the number of features).
2) Let $\mathbf{v} = \mathbf{p}(\mathbf{q})$ and $\mathbf{v}_2 = \mathbf{p}_2(\mathbf{q})$.
3) Sort $\mathbf{v}_2$ into a numerically increasing vector, $\mathbf{v}_2'$. Store the permutation, $\pi$, which maps $\mathbf{v}_2'$ to $\mathbf{v}_2$.
4) For $i = 1$ to (Number of rows of $\mathbf{R}$)
  5) For $j = 1$ to (Number of columns of $\mathbf{R}$)
    6) Compute $m = v_i v_j$.
    7) Perform a binary search for $m$ in $\mathbf{v}_2'$; call the index of the resulting location $i_m'$.
    8) Using the permutation $\pi$, find the index, $i_m$, in $\mathbf{v}_2$ corresponding to the index, $i_m'$ in $\mathbf{v}_2'$.
    9) $[\mathbf{R}]_{i,j} = [\mathbf{r}]_{i_m}$
  10) Next j
11) Next i
```

where

$$\bar{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}(\mathbf{x}_i). \tag{23}$$

We first compute $\bar{\mathbf{p}}$ from the utterance (independent of the speaker model). Scoring for each speaker model is accomplished by one inner product—a low-complexity operation.

For optimal Bayes identification with equal class priors, we choose $j^*$ so that $j^* = \operatorname{argmax}_j d(\mathbf{x}_1^N, \mathrm{spk}_j)$. For optimal Bayes verification, we compare the log likelihood ratio to a threshold. For our discriminant function, this corresponds to a threshold on $d(\mathbf{x}_1^N, \mathrm{spk}) - d(\mathbf{x}_1^N, \mathrm{imp})$. Since $\mathbf{w}_{\mathrm{imp}}^t \bar{\mathbf{p}} = 1 - \mathbf{w}_{\mathrm{spk}}^t \bar{\mathbf{p}}$ by virtue of the training method, see [10], the decision rule is based on a threshold for

$$2\mathbf{w}_{\mathrm{spk}}^t \bar{\mathbf{p}} - 1. \tag{24}$$

That is, we only need to compute $\mathbf{w}_{\mathrm{spk}}^t \bar{\mathbf{p}}$ and compare this to a threshold (i.e., we do not have to calculate a cohort score).

In the case of a polynomial of degree two, the methods derived can be related to a standard HMM with full covariance Gaussian emission probabilities. In this case, the log of the emission probability is a quadratic function (a special case of our system). We can thus apply our computation reduction method. In addition, cohort normalization can be simplified (after cohort selection) by noting that

$$\mathbf{w}_{\mathrm{spk}}^t \bar{\mathbf{p}} - \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_{\mathrm{cohort},k}^t \bar{\mathbf{p}} \tag{25}$$

is the same as

$$\left[\mathbf{w}_{\mathrm{spk}} - \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_{\mathrm{cohort},k}\right]^t \bar{\mathbf{p}}. \tag{26}$$

As a final point, we note that our scoring method extends to any linear approximation space. The key property that allows simplification of the scoring is that the quantity $\sum_{i=1}^{N} \mathbf{w}_j^t \mathbf{p}(\mathbf{x}_i)$ can be written as $\mathbf{w}_j^t (\sum_{i=1}^{N} \mathbf{p}(\mathbf{x}_i))$. This property is dependent only on the fact that we are using a linear basis, i.e., in our case, the components of $\mathbf{p}(\mathbf{x}_i)$.

## *B. Prior Compensation*

The prior probabilities of the speaker training set and anti-speaker training set are usually imbalanced because significantly more anti-speaker data is available. There are several approaches to this problem including balancing the amount of data that the training algorithm sees per epoch [15], division of the classifier output by training priors [15], and weighting the training criterion [17].

For the polynomial training approach we use, only the latter two approaches are viable. To incorporate priors into recognition scoring, note that the training process (11) produces an approximation of

$$p(\omega \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \omega)\pi_{\mathrm{spk,\,training}}}{p(\mathbf{x})} \tag{27}$$

where $\pi_{\mathrm{spk,\,training}}$ is the prior from the *training* set for the speaker. To adjust the prior for recognition, we scale as follows:

$$\frac{\pi_{\mathrm{spk}}}{\pi_{\mathrm{spk,\,training}}} p(\omega \,|\, \mathbf{x}) \tag{28}$$

where $\pi_{\mathrm{spk}}$ is the desired prior. Typically, we assume all speakers in the training set have equal desired priors (for identification), so we ignore the desired prior and only divide out the training prior in (28). Note that we can ignore the dependency of $p(\mathbf{x})$ on priors, since $p(\mathbf{x})$ cancels out of the decision process. Dividing priors out can be absorbed into the speaker model, $\mathbf{w}_{\mathrm{spk}}$, using (22), i.e., we just multiply $\pi_{\mathrm{spk}}/\pi_{\mathrm{spk,\,training}}$ times the model after training using (11).

To incorporate priors into the training process, first assume $\pi_{\mathrm{spk}}$ is the desired prior. Then, we can write the expectation in (4) as

$$\begin{aligned}\mathbf{E}\{(\mathbf{w}^t\mathbf{p}(\mathbf{x}) - y(\omega))^2\} &= \pi_{\mathrm{spk}}\mathbf{E}\{(\mathbf{w}^t\mathbf{p}(\mathbf{x}) - 1)^2 \,|\, \omega = \mathrm{spk}\} \\ &\quad + \pi_{\mathrm{imp}}\mathbf{E}\{(\mathbf{w}^t\mathbf{p}(\mathbf{x}))^2 \,|\, \omega = \mathrm{imp}\}.\end{aligned} \tag{29}$$

The criterion for training then becomes

$$\mathbf{w}_{\text{spk}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left[ \frac{\pi_{\text{spk}}}{N_{\text{spk}}} \sum_{j=1}^{N_{\text{spk}}} |\mathbf{w}^t \mathbf{p}(\mathbf{x}_j) - 1|^2 + \frac{\pi_{\text{imp}}}{N_{\text{imp}}} \sum_{j=1}^{N_{\text{imp}}} |\mathbf{w}^t \mathbf{p}(\mathbf{y}_j)|^2 \right]. \quad (30)$$

The prior compensation is easily incorporated into the training algorithm. The new problem becomes

$$\mathbf{w}_{\text{spk}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{DMw} - \mathbf{Do}\|_2 \quad (31)$$

where $\mathbf{D}$ is a diagonal matrix. The method in Table II is easily adapted to the new training (31).

The choice of the desired prior in (30) depends upon the application. Theoretically, one should choose priors corresponding to the actual observed number of impostors and valid access attempts. For our experiments, we chose "uniformative" priors-$\pi_{\text{spk}} = \pi_{\text{imp}} = 0.5$; that is, we do not bias toward a high number of impostor attempts or a high number of valid accesses. This choice also weights approximation error equally on the speaker and anti-speaker training set. Another method, which we do not explore in this paper, is to treat $\pi_{\text{spk}}$ as a hyperparameter. Because we are dealing with approximations of probabilities and finite training data, the best choice of prior is database dependent. We could hold back a cross-validation set and test $\pi_{\text{spk}}$ over a range of values to determine the best performing $\pi_{\text{spk}}$ for our evaluation criterion.

## V. Extensions

New class addition is easily performed with the new method. Suppose we have $N_{\text{spk}}$ classes. Assume we have stored the vectors $\mathbf{r}_i$. Then as new class data is acquired, we form a new vector $\mathbf{r}_{N_{\text{new}}}$ using the update

$$\mathbf{r}_{N_{\text{new}}} = \mathbf{r}_{N_{\text{new}}} + \mathbf{p}_2(\mathbf{x}_i) \quad (32)$$

where $\mathbf{x}_i$ is a feature vector from the new class. Then we construct a new $\mathbf{r}$ and retrain all models. This reduces computation considerably since we do not need to recompute $\mathbf{r}$ from scratch. This low computational complexity can be derived by noting that the process of forming $\mathbf{r}$ using matrix multiplication is $O(N_{\text{spk}}M^2)$ where $M$ is the length of the vector $\mathbf{p}(\mathbf{x})$. The cost of solving the matrix equation is $O(M^3)$. For typical problems, $N_{\text{spk}} \gg M$, so retraining is fast. For instance, for the YOHO database [18], if we assume the training consists of 96 2.5-s phrases for 138 speakers, then approximately 3 million feature vectors are generated. For our typical system, we have an $M$ of 455, so that $NM^2 = 5 \times 10^9$ and $M^3 = 94 \times 10^6$.

Reinforcement can also be performed very quickly. We can store the vectors $\mathbf{r}_i$ as in the class addition case. Updating the class $i$ is performed by updating the $\mathbf{r}_i$ vector

$$\mathbf{r}_{i,\text{new}} = \mathbf{r}_i + \mathbf{p}_2(\mathbf{x}_{i,\text{new}}). \quad (33)$$

Retraining can be performed after (33) has been calculated for all new feature vectors.

Additional extensions to the methods shown here are iterative training [19], feature transformation [13], and applications to other pattern recognition problems [20], [21]. The goal of iterative training is to reduce the memory required by the algorithm in Table II. In particular, step 9 in Table II requires a large jump in memory space since the "compressed" representation $\mathbf{r}$ must be expanded to a large matrix $\mathbf{R}$. In [19], we show how the algorithm in Table III can be used to construct an iterative training algorithm. Another extension is feature transformation of the vector $\mathbf{p}(\mathbf{x})$. Transformation of this vector allows us to control the number of model terms, i.e., the dimension of the vector $\mathbf{w}$. The method in [13] shows how the model size can be varied in increments of one using random dimension reduction. An interesting aspect of this method is that the average EER is predictable as a function of the dimension of $\mathbf{w}$. We show that $\log(\text{EER}) \approx 4.70 - 0.72\log(\text{dim})$.

## VI. Results

### A. Setup

We applied our method to the YOHO speaker recognition database [18], [22]. YOHO uses prompted combination lock phrases, e.g., "26-81-57." YOHO has four enrollment sessions with 24 phrases per session. Verification consists of 10 sessions with 4 phrases per session.

Feature analysis was performed by using 12th order LP analysis on a 30-ms frame every 10 ms. Each frame was pre-emphasized with the filter $1 - 0.97z^{-1}$, and a Hamming window applied to the result. From the LP coefficients, 12 LP cepstral coefficients (LPCC) and delta-cepstral coefficients were produced. The choice of LPCCs was arbitrary; the polynomial method works with any feature set.

For enrollment, all four sessions of YOHO were used. For one-phrase verification, all 40 utterances per speaker were used. For four-phrase verification, we grouped all phrases within a session. The classifier in Fig. 1 was used in text independent mode. That is, one model was generated per speaker without using knowledge of the particular text spoken. Fast scoring, as in (22), was used.

### B. Results for Verification

Table IV shows the speaker model size and background model size (i.e., the dimensions of the vectors $\mathbf{p}(\mathbf{x})$ and $\mathbf{p}_2(\mathbf{x})$, respectively). Table V shows a comparison of the average equal error rate (EER) for the classifier at different degrees. Prior compensation (with $\pi_{\text{spk}} = 0.5$), see Section IV-B, and the algorithm in Table II were used. The average EER was computed by finding the EER for each speaker and then averaging across all speakers. Note that the performance of the new technique compares favorably to other systems where EERs of 2.84% (VQ, 1 phrase) [23], 1.07% (HMM, one phrase) [23], 1.7% (DTW, four phrase) [2], and 0.12% (GMM, four phrase) [22] are reported. Note that the comparison is not entirely accurate because of the different experimental setups and the use of text-prompted systems. Text-prompted systems assume knowledge of text; whereas, our system is text-independent. A good compromise between accuracy, computation, and storage appears to be the degree three system which has 455 speaker model parameters per

TABLE IV
POLYNOMIAL CLASSIFIER CHARACTERISTICS WITH DIFFERENT PARAMETERS

| cep | dcep | degree | Speaker Model Size | Background Model Size |
|---|---|---|---|---|
| 12 | - | 2 | 91 | 1,820 |
| 12 | - | 3 | 455 | 18,564 |
| 12 | - | 4 | 1820 | 125,970 |
| 12 | 12 | 2 | 325 | 20,475 |
| 12 | 12 | 3 | 2925 | 593,775 |

TABLE V
AVERAGE EER PERFORMANCE ON THE YOHO DATABASE

| cep | dcep | degree | Avg. EER % 1 phrase | Avg. EER % 4 phrase |
|---|---|---|---|---|
| 12 | - | 2 | 3.62 | 0.90 |
| 12 | - | 3 | 1.52 | 0.28 |
| 12 | - | 4 | 1.00 | 0.14 |
| 12 | 12 | 2 | 1.81 | 0.32 |
| 12 | 12 | 3 | 0.35 | 0.07 |

speaker and 18 564 entries in $\mathbf{p}_2(\mathbf{x})$ (and thus in the background). Table VI shows the performance of the system in terms of *pooled* EER. The pooled EER is the equal error of the system assuming a constant threshold across all speakers. This performance measure is a more reasonable gauge of system performance since it is difficult in practice to set the speaker adapted thresholds needed to obtain the average EER.

To illustrate the effect of prior compensation on accuracy, we trained without the weighting in (31). For the four phrase test, 12 cepstral coefficients, and degree three, the resulting average EER was 0.37%, i.e., a decrease in accuracy over the weighted case was observed. The prior compensation normalized the score range, so that typical EER thresholds were around 0.5.

One concern about the YOHO database is the lack of a background separate from the verification population. Our original enrollment and verification strategy used all 138 speakers for training and testing. We also tried enrolling the first 69 speakers and second 69 speakers separately. Speakers one through 69 were trained only against one to 69; speakers 70 through 138 were trained only against speakers 70 through 138. Verification was performed by using the second 69 speakers as impostors for the first 69 and vice versa. The resulting one phrase EER for 12 cepstral coefficients and degree three was 1.63%. The result is close to the one shown in Table V; this test shows that the polynomial classifier generalizes well to an unknown impostor set. The comparison between the two training approaches is not entirely valid since a larger training background and a larger number of impostors is available for the case in Table V.

## C. Results for Identification

Table VII shows the identification error rates of our system. Error rate is defined as the number of misclassified utterances over the total number of utterances tested. These results compare well to other systems. Results available in the literature for identification using a 10-s test are 0.14% [14] and 0.22% [23]. Our best result is 0.07% for text *independent* speaker identification. Note that [23] and [24] are text *dependent*.

We also compared different strategies for prior compensation for a one phrase (2.5 s) test using 12 cepstral coefficients and a degree three classifier, see Table VIII. Table VIII shows that the weighting method increases accuracy noticeably over the base system. The error criterion weighting also shows improvements over the training prior division method.

TABLE VI
POOLED EER PERFORMANCE ON THE YOHO DATABASE

| cep | dcep | degree | Pooled EER % 1 phrase | Pooled EER % 4 phrase |
|---|---|---|---|---|
| 12 | - | 2 | 4.68 | 2.61 |
| 12 | - | 3 | 2.70 | 1.55 |
| 12 | - | 4 | 2.02 | 1.23 |
| 12 | 12 | 2 | 2.86 | 1.52 |
| 12 | 12 | 3 | 1.07 | 0.65 |

TABLE VII
IDENTIFICATION PERFORMANCE ON THE YOHO DATABASE

| cep | dcep | degree | Error Rate % 1 phrase | Error Rate % 4 phrase |
|---|---|---|---|---|
| 12 | - | 3 | 4.71 | 0.72 |
| 12 | - | 4 | 2.74 | 0.36 |
| 12 | 12 | 2 | 6.16 | 1.16 |
| 12 | 12 | 3 | 1.01 | 0.07 |

TABLE VIII
IDENTIFICATION PERFORMANCE FOR VARIOUS PRIOR COMPENSATION STRATEGIES

| Method | Error Rate % 1 phrase |
|---|---|
| None | 17.34 |
| Score Divided by Prior | 11.07 |
| Error Criterion Weighting | 4.71 |

## VII. CONCLUSION

We have demonstrated a novel set of methods for using polynomial classifiers for speaker recognition. We have derived a novel training algorithm with low storage, simple implementation, and reduced computational complexity over traditional methods. We have also given a scoring method that is computationally scalable with the number of speakers allowing the method to be used on large databases. Accuracy was shown to be excellent and to be competitive with more traditional methods such as HMMs and GMMs.

## REFERENCES

[1] D. A. Reynolds, "Automatic speaker recognition using Gaussian mixture speaker models," *Lincoln Lab. J.*, vol. 8, no. 2, pp. 173–192, 1995.

[2] A. Higgins, L. Bahler, and J. Porter, "Speaker verification using randomized phrase prompting," *Digital Signal Process.*, vol. 1, pp. 89–106, 1991.

[3] A. E. Rosenberg, J. DeLong, C.-H. Lee, B.-H. Juang, and F. K. Soong, "The use of cohort normalized scores for speaker verification," in *Proc. Int. Conf. Spoken Language Processing*, 1992, pp. 599–602.

[4] M. J. Carey, E. S. Parris, and J. S. Bridle, "A speaker verification system using alpha-nets," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1991, pp. 397–400.

[5] A. E. Rosenberg and S. Parthasarathy, "Speaker background models for connected digit password speaker verification," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1996, pp. 81–84.

[6] K. R. Farrell, R. J. Mammone, and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 194–205, Jan. 1994.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.

[8] D. F. Specht, "Generation of polynomial discriminant functions for pattern recognition," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 308–319, June 1967.

[9] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag, 1996.

[10] J. Schürmann, *Pattern Classification*. New York: Wiley, 1996.

[11] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Appl. Opt.*, vol. 26, no. 23, pp. 4972–4978, Dec. 1987.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: John Hopkins Univ. Press, 1989.

[13] W. M. Campbell, K. Torkkola, and S. V. Balakrishnan, "Dimension reduction techniques for training polynomial networks," in *Proc. 17th Int. Conf. Machine Learning*, vol. 200, pp. 1015–1022.

[14] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[15] N. Morgan and H. A. Bourlard, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA: Kluwer, 1994.

[16] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[17] D. Lowe and A. R. Webb, "Exploiting prior knowledge in network optimization: An illustration from medical prognosis," *Network: Comput. Neural Syst.*, vol. 1, pp. 299–323, 1990.

[18] J. P. Campbell Jr., "Testing with the YOHO CD-ROM voice verification corpus," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1995, pp. 341–344.

[19] W. M. Campbell, "An iterative technique for training speaker verification systems," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2000, pp. 1185–1188.

[20] W. M. Campbell, K. T. Assaleh, and C. C. Broun, "Low-complexity small-vocabulary speech recognition for portable devices," in *Proc. Int. Symp. Signal Processing Applications*, 1999, pp. 619–622.

[21] K. Torkkola and W. M. Campbell, "Mutual information in learning feature transformations," in *Proc. 17th Int. Conf. Machine Learning*, 2000, pp. 1015–1022.

[22] J. P. Campbell Jr., "Speaker recognition: A tutorial," *Proc. IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.

[23] J. Colombi, D. Ruck, S. Rogers, M. Oxley, and T. Anderson, "Cohort selection and word grammar effects for speaker recognition," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1996, pp. 85–88.

[24] C. Che and Q. Lin, "Speaker recognition using HMM with experiments on the YOHO database," in *Proc. Eurospeech*, 1995, pp. 625–628.

**William M. Campbell** received B.S. degrees in electrical engineering, computer science, and mathematics from the South Dakota School of Mines and Technology, Rapid City, in 1990, the M.S. degree in applied mathematics in 1993, and the Ph.D. degree in applied mathematics in 1995, both from Cornell University, Ithaca, NY.

From 1995 to 1999, he was a Senior Research Scientist with the Speech and Signal Processing Lab, Motorola Space and Systems Technology Group (SSTG), Tempe, AZ. While at Motorola SSTG, he did research in biometrics, speech interfaces for wearable computing, and communications for the battlefield. He is currently a Principal Research Scientist with Motorola Labs. He has received the Motorola Distinguished Innovator award and holds 11 patents. His research interests include machine learning, speech processing, and mathematical techniques in signal processing.

**Khaled T. Assaleh** (S'90–M'90–SM'00) received the B.Sc. degree in electrical engineering from Jordan University, Amman, Jordan, in 1988, the M.S. degree in electronic engineering from Monmouth University, West Long Branch, NJ, in 1990, and the Ph.D. degree in electrical engineering from Rutgers, The State University of New Jersey, Piscataway, NJ, in 1993.

From 1993 to 1994, he was with Rutgers Unviersity/CAIP Center as a Research Assistant Professor where he researched and developed algorithms for robust speech and speaker recognition. During that period, he was a key contributor to the development of Speak-EZ speaker authentication technology. In November 1994, he joined Motorola, Inc., Scottsdale, AZ, as a Staff Engineer where he was responsible for the speech and speaker recognition activities in the Speech and Signal Processing Lab (SSPL) in the Government and Space Technology Group (GSTG). Since March 1997, he has been with Conexant Systems, Inc. (formerly Rockwell Semiconductor Systems), Newport Beach, CA. He joined as a Senior Staff Engineer with the Speech Processing Group at the Multimedia Communications Division where he was a key contributor to several projects that involved speech and signal processing. Since December 2000, he has been an Engineering Group Leader with the Wireless Communication Division where he is responsible for speech and audio processing including speech and speaker recognition, acoustic echo cancellation, noise cancellation, and text-to-speech synthesis. He holds seven patents and has authored and coauthored over 25 published articles. His research interests have included wavelets, time-frequency analysis, speech signal processing, robust speech and speaker recognition algorithms, and pattern classification.

**Charles C. Broun** received the B.S. degree in 1990 and the M.S. degree in 1995, both in electrical engineering from the Georgia Institute of Technology, Atlanta. He specialized in digital signal processing, communications, and optics.

From 1996 to 1998, he conducted research in speech and speaker recognition at the Speech and Signal Processing Lab, Motorola SSG, Scottsdale, AZ, where he worked on the speaker verification product CipherVOX and the System Voice Control component of the U.S. Army's Force XXI/Land Warrior program. In 1999, he joined the newly formed Motorola Labs—Human Interface Lab, Tempe, AZ, where he expanded his work to multi-modal speech and speaker recognition. Currently, he is the Project Manager for several realization efforts within the Human Interface Lab—Intelligent Systems Lab. The most significant is the Driver Advocate, a platform supporting research in driver distraction mitigation and workload management.