Autonomous Vehicle Speaker Verification System

Aaron Pfalzgraf, Christopher Sullivan, Dr. Jose R. Sanchez

***Abstract*—With the increasing interest in vehicle automation, security methods for these systems have become a primary concern. One possible security measure is a speaker verification system (SVS), which can identify certain features of a pre-selected user's voice. The goal of this project was the implementation of a speaker verification-protected voice command system on a Texas Instruments (TI) c5535 eZdsp development board. The complete system is intended for integration with an autonomous vehicle control system, although this integration is outside the scope of the project. For safety reasons, the SVS was designed to minimize true speaker rejection errors at the cost of elevated imposter acceptance errors. Similarly, the speech recognition system was designed to minimize command rejection errors and command misinterpretation errors at the cost of elevated foreign word acceptance errors. The system accepts speech data through a handheld cardioid microphone. The speech data is stored in a buffer, processed with a Hamming window, and condensed into feature vectors of Mel-warped cepstral coefficients (MWCC). A set of four artificial neural networks (ANN) are used to accomplish both the speech recognition and speaker verification tasks. These ANN's are trained externally with feature vectors of pre-recorded training speech using the back-propagation algorithm. ANN training is performed in MATLAB and the resulting weight vectors are exported for real-time implementation of each ANN on the eZdsp. With a speaker population size of eleven and a word population size of six, simulations of the system have yielded a true speaker rejection rate of 0.5%, an imposter acceptance rate of 6.5%, a command rejection or misinterpretation rate of 0%, a true speaker foreign word acceptance rate of 13%, and an imposter foreign word acceptance rate of 0%. Early implementation results with a speaker population size of six and a word population size of twelve have yielded a true speaker rejection rate of 3.1%, an imposter acceptance rate of 5%, a command rejection or misinterpretation rate of 0%, a true speaker foreign word acceptance rate of 15.5%, and an imposter foreign word acceptance rate of 3.5%.**

I. Introduction

Speaker verification systems are systems that can identify someone by the sound of his or her voice. Speaker verification is not to be confused with speech recognition. Speech recognition systems determine which words an individual says, not which person said them. Speaker verification systems can be either text-dependent or text-independent. Text-dependent systems rely on the speaker saying a specific word or phrase to correctly identify him or her, while text-independent systems can identify a speaker regardless of the words he or she says. In theory, everyone's vocal tract is shaped differently enough to uniquely identify them. Through observation of the features of an individual's speech, an ideal speaker verification system should be able to uniquely determine the identity of any speaker. [1]

A. Background and Motivation

Speaker verification systems have applications primarily within the security industry. Many common existing security measures, such as passwords and keycards, are easily bypassed by imposters and lost or forgotten by the true operator. Using an individual's voice to confirm his or her identity is advantageous because it is a security measure that is as difficult for imposters to replicate as it is for the true operator to lose or forget. Speaker verification is particularly useful in securing voice command systems, because the operator's identity can be conveniently checked every time he or she says a command. No other standard or biometric security system can be integrated with a voice command system to achieve this level of security. This project investigates the value of integrating an SVS into an autonomous vehicle voice command control system. Voice command systems are inherently risky due to the potential for any speaker to say a command word and control the vehicle. With the integration of an SVS, the autonomous vehicle could be programmed to accept commands only from a designated operator's voice, reducing safety hazards, as shown in Figure 1.

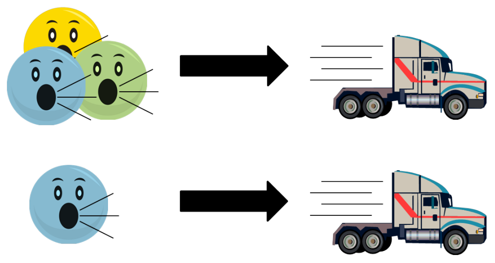

Fig. 1. An SVS can be used to reduce the number of designated operators of an autonomous vehicle.

B. Problem Formulation

Designing the proposed voice command system involved three main tasks:

- Design a speech recognition system
- Design a speaker verification system
- Integrate both systems in real time on a digital signal processor (DSP)

Because the emphasis of this project was on speaker verification, a simplistic speech recognition system was proposed. The speech recognition system was specified to recognize the command words "stop" and "go" and reject all others. Due to the monosyllabic nature of these command words, the system can function exclusively in the frequency domain without taking word length or sound order into account. Also, because the command words do not share any consonant or vowel sounds, the system does not need to be able to recognize the sounds of specific consonants or vowels. Each command word can be processed as a whole to minimize computation without tremendous loss of accuracy. With only two command words, the speaker verification system could be designed as a text-dependent system without too much added computational burden in the implementation stage. Text-dependent systems are generally more accurate than text-independent systems because they only need to be responsible for determining differences in the way different people say the same word. The final system was designed to accept audio from a microphone, buffer the audio for processing, apply the recognition and verification systems to the buffered audio in series, and output a final command score used to determine whether to set or clear a command flag internal to the DSP. Upon integration with an autonomous vehicle controller, this command flag would be the only necessary communication between the DSP and the vehicle.

C. System Specifications

The following system specifications were decided upon:

- True speaker rejection rate under 1%
- Imposter acceptance rate preferably beneath 2%
- Command rejection and misinterpretation rate under 1%
- Maximum of 50 ms delay between spoken command and command flag handling
- System functional in environments with mild background noise

II. Methods

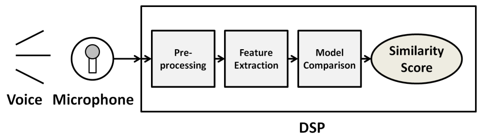

Fig. 2. Continuous audio data from a microphone is processed by a Digital Signal Processor (DSP) to perform speech recognition and speaker verification tasks.

Figure-2 shows an overall block diagram for the system. The user speaks a command word into a microphone. The microphone used for the final implementation is an AKG D5 dynamic microphone. This microphone was chosen for its cardioid pickup pattern and close distance of operation. Without any noise cancellation software or hardware, a microphone largely unaffected by ambient noise was a requirement of the system. The user's speech is passed through a PreSonus AudioBox USB pre-amplifier for volume control and read at an 8 kSamples/s sampling rate. The processing of the data was performed, in C, on an eZdsp5535 development board (Texas Instruments, Dallas, TX). Each block inside of the DSP, shown in Figure 2, was tested in MATLAB® (Mathworks,Natick,MA) before implementation in C.

A. Pre-processing

The first task to be performed in software is pre-processing. The pre-processing block buffers the continuous stream of incoming audio data into 25 ms frames with 50% overlap. With an 8 kSamples/s sampling rate, the 25 ms buffer only fills 200 memory locations in the DSP while remaining capable of accurately representing frequency content as low as 40 Hz. This buffer length is a good tradeoff between memory efficiency and frequency appropriateness for human speech processing. Utilizing 50% overlap between subsequent audio frames effectively doubles the amount of data that describes each spoken word without needing to increase the sampling rate or the syllable length of the command words. Figure 3 demonstrates the overlapping frames technique.

When an audio frame is detected to be full, the pre-processing block checks to see if the audio data present in the frame is loud enough to potentially contain speech before allowing the frame to be processed further. This increases the computational efficiency of the system. The system rejects any audio frame with a maximum amplitude less than one sixteenth of the maximum amplitude that can be represented by the fixed point DSP. This value is computationally efficient in a fixed point system because it is a power of two, and it was determined experimentally to be a good cut-off threshold for the D5 microphone in most ambient conditions.

To regulate the amplitude of the audio buffer, the pre-processing block normalizes any audio frame loud enough to potentially contain speech data so that the maximum amplitude contained in the frame is represented by the maximum possible amplitude. This mitigates the effect of the operator standing at different distances from the microphone and speaking at different volume levels. This mitigation comes at the cost of amplitude data loss across different audio frames from the same spoken command word. This has the potential to reduce the overall accuracy of the system, but this method was the most computationally efficient way for the system to handle volume discrepancies. It cannot perfectly eliminate volume error, however, because changes in distance from the microphone and speech volume affect the frequency content of the recorded audio.

After amplitude normalization, a Hamming window is applied to any non-silent 25 ms audio frame. A Hamming window is described by the following expression in the time domain:

$$w(n) = \alpha + \beta \cos\left(\frac{2\pi n}{N}\right), \quad \alpha = 0.54 \text{ and } \beta = 0.46. \quad (1)$$

The Hamming window was chosen for its ability to apply frequency coloration evenly throughout the spectrum. The Hamming window has the following characteristics: main lobe width of 1.3 frequency bins, first side-lobe attenuation of -42 dB, and side-lobe roll-off of -20 dB/decade. The window's relatively narrow main lobe prevents excessive frequency smearing, while its artificially lowered first side-lobe amplitude prevents the first side-lobe from contributing noticeably more to the total frequency coloration than the other side-lobes.

B. Feature Extraction

Feature extraction seeks to represent a frame of audio in a computationally efficient manner that clearly emphasizes distinctive characteristics of the speaker's voice. Most real time feature extraction techniques focus on representing the

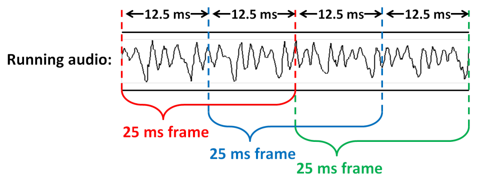

Fig. 3. The pre-processing block buffers continuous audio input into 25 ms frames with 50% frame overlap.

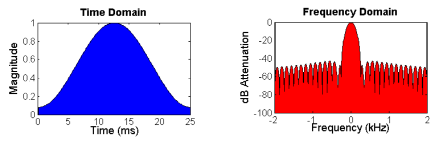

Fig. 4. The Hamming window in the time and frequency domains.

frequency content of the voice. Such techniques include linear predictive coding (LPC) coefficient extraction and Mel-warped cepstral coefficient (MWCC) extraction. This system uses MWCC extraction due to the popularity of MWCC's in modern speech recognition and speaker verification systems. [3] Each normalized and windowed audio frame is condensed into 15 MWCCs during feature extraction.

MWCCs are a measure of short-term power spectral density (PSD). Figure 5 details the MWCC extraction process. A 512-point Fast Fourier Transform (FFT) of the pre-processed audio frame is taken and multiplied by its complex conjugate to yield the PSD of the frame. The first 256 points of the PSD are summed into 32 Mel-warped triangular bins with 50% overlap. The 512-point FFT is necessary to ensure each triangular bin is described by at least three points. The number of bins is related to the number of desired MWCCs, so the selection of 15 MWCCs to describe each audio frame also helps ensure valid sized triangular bins. Due to the logarithmic warping of the Herz scale into the Mel scale, the triangular bins lower in the frequency spectrum are much narrower than those higher in the frequency spectrum. The relationship between the Mel scale and the Herz scale is described by the following equation:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{2}$$

where f is a frequency in Hertz and m is its Mel equivalent. Mel-warping is performed in order to mimic the frequency response of the human ear. Humans are very talented at identifying speakers by the sounds of their voices, so processing speech data as closely as possible to the way a human ear does is beneficial to system accuracy. After the PSD is collected into 32 Mel-warped triangular bins, the natural logarithm is performed on all 32 bin values. The Type II Discrete Cosine Transform (DCT-II or DCT) is performed on the resulting 32 values to remove the correlation between overlapping bin powers. The DCT-II is a computationally efficient inverse Fourier Transform using exclusively real numbers. The equation for the DCT-II is:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N} n(k+\frac{1}{2})\right] \quad k = 0, ..., N-1. \tag{3}$$

In this equation, X_k is the set of output MWCCs, x_n is the set of input bin powers, N is the total bin count of 32, n describes which bin power is being currently processed, and k describes which output value is being currently calculated. Out of the resulting 32 computed values, only 15 are kept to describe the audio frame. The DCT generates a symmetrical output, so the second half of the values can be discarded with no loss of information. Also, the first of the output values is closely related to the amplitude bias, or DC offset, of the given audio frame. The amplitude bias contains no useful information for either speech recognition or speaker verification systems, so the first value is also discarded. This results in a feature vector of 15 MWCC's. For a deeper explanation of MWCC calculations, consult [5].

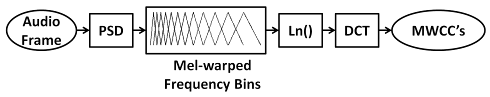

Fig. 5. Flow chart for the Mel-warped cepstral coefficient calculation.

MWCCs are valuable features for performing speech recognition and verification, because they measure the total amplitude contribution of the frequencies present in the audio signal in a memory-efficient and easily separable way. For speech recognition, the MWCCs can be analyzed in terms of the frequencies associated with specific utterances. An utterance is a consonant or vowel sound. The command words ”stop” and ”go” are composed of several distinctive utterances that can be identified by their frequency content. For speaker verification, the MWCCs can be analyzed in terms of the frequencies associated with a specific speaker's vocal tract. The unique shape of every person's vocal tract contributes a certain frequency coloration to their speech which can be used to identify them. Changing which frequencies are used to differentiate between the calculated MWCCs can change which of the two tasks is being performed.

C. Model Comparison

The model comparison software block accepts data that describes the input audio and compares this data against pre-generated models to perform speech recognition and speaker verification. The input data to this block is a set of MWCC vectors that ideally describe an entire spoken word. To pass a representative number of MWCC vectors into model comparison, the system must be able to store computed MWCCs in a buffer and determine when the operator's speech begins and ends. The MWCC storage buffer can store a maximum of sixty MWCC feature vectors. This number of feature vectors can represent a maximum of 0.75 s of audio. The minimum number of stored MWCC's necessary for the system to detect a spoken word is fifteen. This equates to 0.1875 s of audio. The MWCC buffer content is sent to the model comparison block when either all sixty buffer values fill up or the operator's speech falls silent for at least 75 ms between fifteen and sixty stored MWCC vectors. Model comparison itself is essentially a hyper-dimensional cluster analysis problem. The MWCC feature vectors plotted in 15-dimensional space would form clusters that represent different utterances and speakers. Due to the highly non-linear separation of these clusters, artificial neural networks (ANNs) were selected to perform model comparison. ANNs use a complex series of weights, summations, and activation functions to perform universal function approximation by drawing partitions between clusters of data. Figure 6 shows a simple ANN layout. The weights, or gain blocks, associated with an ANN are calculated during its training stage. By feeding a set of training data into an ANN and iteratively modifying its weights so that the network output approaches desired values, the ANN can learn how to perform a certain task. Figure 7 shows the results of applying an ANN to a 2-dimensional cluster analysis problem. Four ANNs are used in this system to partition the MWCC data in a way that performs speech recognition and speaker verification. All four ANNs share the following structural characteristics:

- Fifteen input nodes to accept fifteen MWCCs
- Two hidden layers for computational efficiency and training feasibility
- Fifteen nodes per layer to avoid overflow or accuracy reduction during summation
- Hyperbolic tangent activation function to draw smooth partition lines between clusters
- Single output node to generate a similarity score between negative one and one

While the ANNs share the same structure for code memory efficiency, they are trained with different training data to perform different tasks. Two ANNs work in tandem to accomplish speech recognition. One of these ANNs outputs a score between negative one and one measuring the similarity between the input MWCC vectors and the word ”go”, while the other measures the similarity between the input vectors and the word ”stop”. The positive training data for these ANNs was the true speaker saying the corresponding correct command word, and the negative training data was this same true speaker saying the wrong command word and a collection of other invalid words. Using only takes of true speaker data to train the recognition system helps each ANN draw a specific and strong partition between the two true speaker command words to minimize true speaker rejection and command misinterpretation errors.

The second two ANNs each perform text-dependent speaker verification. These ANNs used the true speaker saying the corresponding correct command word as positive training and a set of imposters saying this command word as negative training data. The speaker population was different for simulation and implementation conditions due to availability of speakers. These populations are elaborated upon during the simulation and implementation results analyses. Splitting the system into four subsystems led to greater overall accuracy, because each ANN was trained to draw a very specific partition line between groupings of MWCC data. Trying to accomplish all four tasks with one ANN would have required a more complicated network structure and a more elaborate training process.

The ANNs are trained in MATLAB® using speech data recorded at 8 kSamples/s with the microphone. The training process uses the back-propagation algorithm over one million iterations. The back-propagation equation for iterative weight adaptation is as follows [4]:

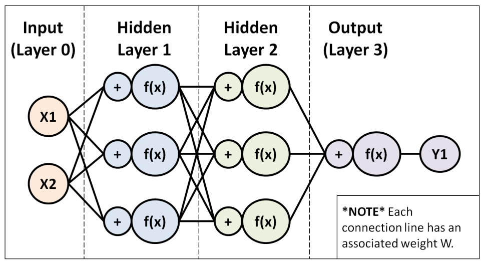

Fig. 6. ANN's can be trained to approximate any function no matter its linearity or complexity with a ”spider web” network of node connections.

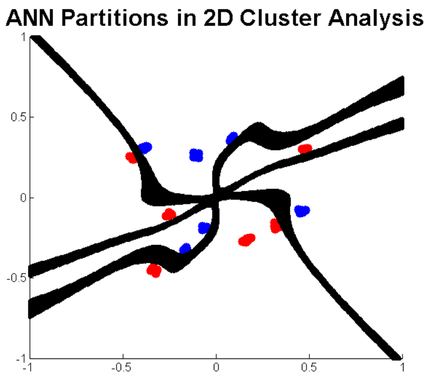

Fig. 7. ANN's can draw partition lines between clusters of data. The method is as valid for the 15-dimensional feature space considered by the voice command system as it is for the 2-dimensional cluster analysis shown above.

$$\Delta w_i = \alpha(t-y)\varphi' x_i, \tag{4}$$

where w_i is the weight being adjusted, α represents the learning rate of the system, y is the node output, t is the target node output, φ' is the derivative of the activation function, and x_i is the current node output. The learning rate, α is a variable that ensures the stability of the system and determines how quickly the weights can change. The training of the system can be thought of as an optimization problem. The system attempts to follow the gradient of steepest descent of the error to minimize the error as quickly as possible. However, if the system follows this gradient of steepest descent into a local error minima, any additional small adaptation of the weight values will likely increase the total error. Such a circumstance may cause the training algorithm to get stuck in the local minima and stop calculating more optimal weight values. To correct this, an adaptive learning rate was implemented. The adaptive learning rate steadily increases α during periods of weight adjustment inactivity so the system can stop following the same gradient of steepest descent. This helps the training algorithm swing out of local minima and follow new gradients of steepest descent to further optimize the weight values. Figure 8 shows graphically how adaptive learning assists the weight training.

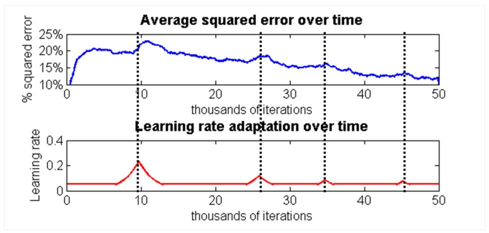

Fig. 8. Adaptive learning helps the training algorithm break out of periods of inactivity and continue to improve the ANN weights.

D. Scoring

MWCC vector data is passed through the ANNs to generate similarity scores between negative one and one. Choosing the optimal score threshold for each ANN requires observation of typical and worst-case scores associated with several circumstances. The two recognition ANN score thresholds must be set low enough that typical true speaker valid command scores are well within the success range. Worst-case true speaker valid command scores should be very near the threshold so that the 1% command rejection and misinterpretation error specification can be met. Likewise, the threshold of the two speaker verification ANNs should be set to meet the 1% true speaker rejection specification while minimizing the imposter acceptance error. The optimal score thresholds for the DSP-implemented ANNs were found by passing each true speaker command word into the system twenty times and observing the worst case and average scores. The decision thresholds in use are:

- ”Go” speaker recognition: 0.3052
- ”Go” speaker verification: 0.6104
- ”Stop” speech recognition: -0.2747
- ”Stop” speaker verification: 0.3662

E. DSP Implementation Considerations

The voice command system was implemented on a TI c5535 eZdsp development board for real time operation. The chosen DSP is a 16-bit system optimized for fixed-point math. It includes $\frac{1}{8}$” stereo input and output jacks for audio communication and 320 kBytes of on-chip memory. The board was chosen for its simplicity, ample onboard memory, audio processing capabilities, and compatibility with Code Composer Studio (Texas Instruments, Dallas, TX). Successfully implementing the system in real time on a fixed-point DSP required consideration of several error sources.

It is hugely inefficient to perform floating-point math on the fixed-point DSP. In order to represent fractional values, Q number format was necessary. This format allocates a certain number of bits of a variable to represent the variable’s fractional component at the cost of reducing the maximum magnitude that the variable can store. A Q10 number, for example, has 10 bits of fractional precision and can only represent values with magnitudes between 32 and -32 (the six remaining magnitude bits can represent values between plus or minus 32 in two’s complement format). This magnitude vs. precision tradeoff can cause substantial accuracy errors if not accounted for properly in software. Even when well accounted for, some level of fixed-point quantization error is unavoidable.

Some of the mathematical functions necessary for ANN application and MWCC extraction must be approximated with their corresponding Taylor series for computational efficiency. The three functions requiring Taylor series approximation were the natural logarithm, cosine, and hyperbolic tangent. Five terms of each of these series are computed to generate reasonable approximations of the functions. Each Taylor series introduces a new source of error into the system. These errors are especially noticeable when the approximated functions have to operate towards the outer limits of their convergence region. Approximation errors are nearly unavoidable without sacrificing system computation speed.

Real time operation of the voice command system is not always possible considering the huge computational burden of applying four ANNs. By using an audio read interrupt, the system is able to successfully operate without dropping samples during the MWCC extraction stage, but the ANN application stage requires too many clock cycles for real time operation to continue. During this stage, audio interrupt capability is toggled off so the stage can complete as quickly as possible and audio data collection can begin reliably again. Audio interrupt capability must also be toggled off any time the audio buffer itself is being accessed. This occurs only during the frame normalization and windowing functions. The potential for dropping single isolated samples during normalization and windowing and the potential for dropping large chunks of command words said in quick succession during the ANN application stage bring about additional sources of error. Presently, the implemented system does not take acoustics or room noise into consideration. An ideal system would include a noise removal stage in the pre-processing block. It is hard to precisely quantify the effect room noise has on the system, but earlier system tests with an omnidirectional microphone with a 20 ft operating distance yielded large error margins due to room noise. Replacing this microphone with a cardioid, close-distance microphone has reduced these errors significantly, but not completely. Distance discrepancies between the user and the microphone are partially accounted for by audio frame amplitude normalization, but room acoustics have coloration effects on the frequency content of the speech that are not corrected by the software. Room acoustics and ambient noise both affect the accuracy of the system.

III. Results

A. Simulation

MATLAB® simulations of the full voice command system were performed to analyze the expected error rates of the system before implementation on the eZdsp. The simulation results are shown in Figure 9. Simulation results were gathered over 200 iterations of full system ANN generation with six takes of true speaker speech and ten instances of different imposter speech for both command words and four additional foreign words. Ideal simulated ANN decision thresholds were not the same as implementation thresholds and were chosen retroactively to yield the lowest possible error margins. The

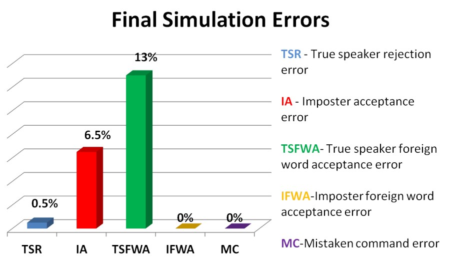

Fig. 9. MATLAB® simulation results come close to meeting system specifications.

results presented in Figure 9 show five error categories: true speaker rejection, imposter acceptance, true speaker foreign word acceptance, imposter foreign word acceptance, and mistaken command. The true speaker rejection rate is a rate at which the authorized user's commands are denied. The specified maximum error rate for this section is 1%. Imposter acceptance is the rate at which unauthorized users were permitted to operate the system. The specified preferable maximum error rate for this section is about 2%. True speaker foreign word acceptance is the rate at which the system performed commands when the true speaker said a foreign word instead of a command word. While minimizing this error source would be beneficial to overall system performance, no specification for this error was decided upon. True speaker foreign word acceptance errors can be easily mitigated by incorporating a manual "on/off" switch in the final design so that the system does not accept any commands when the true speaker is not choosing to command the vehicle. Imposter foreign word acceptance is the rate at which the system performed commands when someone other than the true speaker said a foreign word in the vicinity of the microphone. This category reflects the ability of the system to function safely in environments with background conversational noise. While not explicitly specified, the voice command system should be robust enough so that this error source is essentially negligible (below 1%). Mistaken command is the rate at which the system misinterpreted command words spoken by the true speaker as the wrong command word. The specified maximum error rate for this section is 1%.

The simulation results prove that the theory behind the voice command system's operation is valid. However, the observed error rates from this simulation are an estimate of the best case implementation error rates. In practice, the sources of error associated with real time implementation of the system on the DSP should increase the error rates in almost every category. Also, the limited population size of the simulation does not accurately reflect the ultimate conditions of the implemented system and has likely introduced a bias on the observed simulation results.

B. Implementation

A thorough analysis of the functionality of the implemented voice control system was performed to calculate the error margins associated with each of the five possible error categories. The results are shown in Fig. 10. These results were

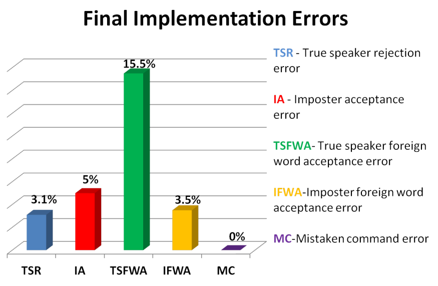

Fig. 10. Implementation results are generally less accurate than simulation results but still demonstrate the functionality of the system.

gathered by observing the output flag of the voice command system after having the true speaker or an imposter say either command words or foreign words into the microphone. The four ANNs in the system were generated in MATLAB and hard coded into the DSP. The ANNs were trained over one million back-propagation iterations each with training data composed of thirty takes of the true speaker saying each command word, three takes of the true speaker saying each of ten foreign words not sharing any vowel or consonant sounds with either command word, ten takes of six imposters (one female, five male) saying each command word, and one take of each of the six imposters saying each of the ten foreign words. The recognition ANNs were trained only with true speaker data to minimize command misinterpretation and true speaker rejection errors. The verification ANNs were trained only with corresponding command word data to generate a text-dependent system.

True speaker rejection error data was gathered over 160 requests of the true speaker saying both "stop" and "go". Eighty of these requests were performed on the same day that the ANN training data was recorded and the other eighty were performed a day afterwards to observe how day-to-day voice changes affect the system. A total of five out of 160 requests were denied. All five of these denied requests occurred during a "go" command on the day after the ANN was trained.

Imposter acceptance error data was gathered over 120 requests of two imposters saying "stop" and "go". One imposter was male and one was female. A total of six out of 120 requests were accepted. Five of these six occurred during a "stop" command by the female imposter, while one came from a "stop" command by the male imposter. This is moderately surprising, because the true speaker is a male.

True speaker foreign word acceptance error data was gathered over eighty requests of the true speaker saying words that were neither "stop" nor "go". Forty of these requests were made on the ANN training day, and the other forty were made the day after. Twenty of each of the sets of forty requests used monosyllabic request words with different vowel sounds than the ones found in "go" and "stop", while the other twenty used monosyllabic request words that shared their vowel sound with either "go" or "stop". A total of seventeen out of eighty requests were accepted. None of the day one, different vowel words were accepted. Eight of the day one, same vowel words were accepted. Two of the day two, different vowel words were accepted. Five of the day two, same vowel words were accepted. Assuming an even distribution of vowel sounds in words (roughly 20% to each vowel), this yielded a total error of 15.5%. It is important to note that this error can be considered as only 7.8% due to the nature of the command system. If the robot is currently in the stopped state, a foreign word misinterpreted as a ”stop” command would not have any effect on the system. Likewise, a robot in the moving state would not be affected by a foreign word misinterpreted as ”go”. This essentially halves the effect foreign word acceptance has on the system. Unfortunately, this effect cannot be applied to imposter acceptance, because imposters should be considered to be purposely trying to affect the state of the robot in most cases.

Imposter foreign word acceptance error data was gathered over eighty requests of the male and female imposters saying words that were neither ”stop” nor ”go”. The composition of the imposters' word selection was the same as the true speaker's for true speaker foreign word acceptance error analysis. A total of three out of eighty requests were accepted. One of the female imposter, same vowel words was accepted. One of the female imposter, different vowel words was accepted. One of the male imposter, same vowel words was accepted. None of the male imposter, different vowel words were accepted.

A mistaken command error would have been observed if any of the true speaker rejection error data points had shown the system to have misinterpreted a true speaker ”stop” command for a true speaker ”go” command. This never happened.

IV. Conclusion

As expected, the implementation errors were slightly higher than those predicted in simulation. This should be due mostly to errors associated with the fixed-point DSP implementation as well as the more strenuous implementation testing conditions. Ultimately, the specifications were not fully met, but the final system does function with a usable degree of accuracy for many applications. Also, the true speaker rejection and imposter acceptance errors should be able to be decreased in a number of ways. The decision thresholds of each ANN can be more finely tuned to accept the true speaker and reject imposters. A noise cancellation system can be added to the pre-processing stage to reduce the effects of room noise. A pre-emphasis filter can be added to the pre-processing stage to emphasize the frequencies commonly associated with speech recognition and speaker verification. Such a filter was not included in this design due to time constraints and for ease of application of both the recognition and verification system to the same MWCC vectors.

Unfortunately, the system's foreign word acceptance error is more difficult to remedy. The bulk of this error margin was due to words that share a vowel sound with the command words. Due to the ANN method of analyzing entire words at once, it is a very difficult task to train the recognition ANNs to tell the difference between two words composed of similar sounds. Creating a more robust speech recognition system would likely involve time domain analysis or consonant identification.

The final implemented voice command system is able to function in real time with generally acceptable margins of error that do not quite reach specified accuracy levels. The system is presently unfit for use in high-accuracy voice control applications, but it could be easily and beneficially integrated into safe, consumer electronics devices. The completed system demonstrates that a real-time speaker verification-protected autonomous vehicle voice control system is feasible.

Acknowledgment

The authors would like to thank the following persons for providing valuable speech data: Mr. Anson Goode, Ms. Brooke Voeller, Mr. Derek Yeghiazarian, Ms. Emily Goldman, Mr. Ethan Hoerr, Mr. Jacob Knobbe, Mr. Jake Hayes, Mr. Jake Siegers, Ms. Kaitlin Pell, Ms. Kathryn Kennedy, Ms. Lorelei Volpe, Mr. Matt Farbota, Mr. Noah Dupes, Mr. Ryan Burke, and Mr. Sam Rosen.

References

[1] J.P. Cambell Jr., *Speaker Recognition: A Tutorial* NSA,Ft.Mead,MD, Sep. 1997.
[2] F.K. Soong et al., *A Vector Quantization Approach to Speaker Recognition.* AT&T, Murray Hill, NJ, 1985.
[3] T. Kinnunen et al., *Comparison of Clustering Algorithms in Speaker Identification.* Univ. of Joensuu, Joensuu, Finland.
[4] A. K. Jaine et al., *Artificial Neural Networks, A Tutorial.* Michigan State University, East Lansing MI, Mar. 1996.
[5] *Mel Frequency Cepstral Coefficient MFCC Tutorial* Oct. 2013. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/