Real-Time Implementation of Cochlear Implant Speech Processing Pipeline on Smartphones

Shane Parris, Murat Torlak, *IEEE Senior Member*, and Nasser Kehtarnavaz, *IEEE Fellow*

***Abstract*— This paper presents the real-time implementation of an adaptive speech processing pipeline for cochlear implants on the smartphone platform. The pipeline is capable of real-time classification of background noise environment and automated tuning of a noise suppression component based upon the detected background noise environment. This pipeline was previously implemented on the FDA-approved PDA platform for cochlear implant studies. The paper discusses the steps taken to achieve the real-time implementation of the pipeline on the smartphone platform. In addition, it includes the real-time timing as well as the noise suppression results when the entire pipeline was run on the smartphone platform.**

***Index Terms*— Cochlear implants, real-time implementation of cochlear implant speech processing pipeline, smartphone implementation**

I. Introduction

The number of cochlear implant (CI) recipient patients has increased to more than 200,000 worldwide [1]. Advances in the signal processing technology have the potential to provide improved hearing sensation for these patients. CIs perform well in quiet environments, however, in noisy environments their performance has been shown to degrade noticeably [2]. In order to maintain cochlear implant performance across a wide range of noisy environments, a real-time adaptive speech processing pipeline was developed in [3]. This paper discusses an alternative and widely available platform of smartphones to run the previously developed speech processing pipeline. The main motivation in pursuing smartphones as an alternative processing platform for this and other medical applications is their ubiquitous aspect and widespread usage noting that more than a billion smartphones are in use today [4].

The previous PDA platform on which the noise adaptive speech processing pipeline was implemented has been approved for clinical trials by the US Food and Drug Administration (FDA) [5]. This paper builds upon the previous work done in [3, 5-12] to develop a real-time speech processing pipeline capable of classifying the background noise environment and automatically tuning a noise suppression component. The speech processing pipeline implemented on the PDA platform addressed the issue of balancing the computational complexity of its various components while maintaining acceptable classification and implant stimulation rates.

The real-time implementation covered in this paper is similarly capable of classifying the background noise environment and automatically adjusting the parameters of the noise suppression component with the difference that the entire pipeline runs on a smartphone platform at higher processing speeds. This implementation provides improvements over the previous implementation in terms of higher accuracy and higher computational efficiency.

The remainder of this paper is organized as follows. An overview of the cochlear implant speech processing pipeline is presented in Section II for readers to see all the components involved in the pipeline. Details and issues encountered in the real-time smartphone implementation are then mentioned in Section III. Finally, Section IV covers the computation and performance results when running the entire pipeline on a smartphone platform.

II. Previously Developed Cochlear Implant Speech Processing Pipeline

Fig. 1 shows the cochlear implant speech processing pipeline that was previously developed by our research team and reported in [3, 5-12]. The pipeline consists of two parallel simultaneous real-time processing paths. An input signal at 22 kHz is first accumulated into 256 sample frames representing 11.6 ms of audio. These frames are then decomposed into a frequency domain representation using either a recursive wavelet packet transform or a Fast Fourier Transform.

In the primary path, shown in the top path of Fig. 1, noise suppression parameters are computed using frequency decomposition coefficients and previously trained gain tables for each noise class. The noise suppression parameters are applied to the frequency bands of the wavelet packet transform. Channel envelopes are then extracted by combining the suppressed wavelet packet coefficients which fall within certain channel frequency bins. Lastly, the amplitudes of the electrode stimulating pulses are determined using the noise suppressed channel envelopes.

In the secondary processing path, shown in the bottom path of Fig. 1, a voice activity detector (VAD) labels a signal frame as either voiced/unvoiced speech or noise only based on a subband power difference measure between the low frequency and high frequency bands of the first stage wavelet packet coefficients [6]. A guard time of 25 frames is added to help ensure that unvoiced segments of speech are not classified as noise. If a frame is detected as having voice activity, no further processing is done in the secondary path for that particular frame. Otherwise, noise only frames are passed along to the feature extraction where a 26 element MFCC (mel frequency cepstrum coefficients) plus ΔMFCC noise feature vector is computed. A classification is then performed via a GMM (Gaussian mixture model) classifier

Authors are with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX 75080, USA.

978-1-4244-7929-0/14/$26.00 ©2014 IEEE

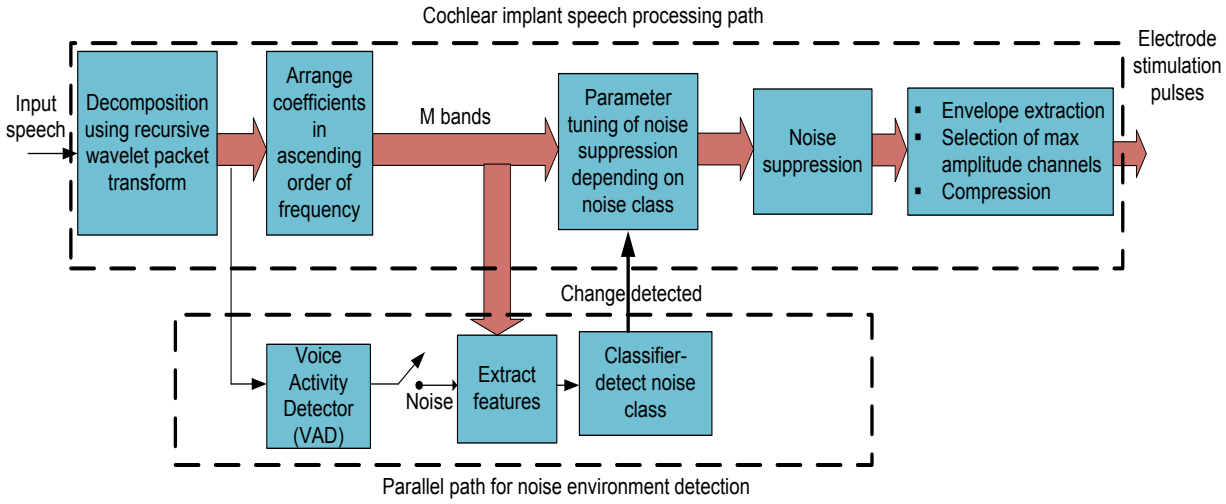

Figure 1. Cochlear implant speech processing pipeline [3]

previously trained for various noise classes. A buffer of previous classification decisions is kept and a majority voting decision is applied to this buffer. The majority voting outcome is passed along to the noise suppression component to trigger a switch of noise suppression parameters if necessary. The buffer is also used to help ensure that the classification does not rapidly fluctuate between classes as quick changes in the applied noise suppression create listener discomfort. The noise suppression component uses the log minimum mean square error criterion.

III. Smartphone Implementation

In this work, a smartphone running the Android operating system was used considering that eighty percent market share of smartphones are now Android smartphones [4]. The model of the smartphone used here was a Motorola Droid 3 smartphone. This phone utilizes an ARM Cortex-A9 processor (1 GHz) with 512 MB RAM on which our pipeline was implemented. The developed code was seamlessly run on other Android smartphones. The Android Developer Tools [14] were used for all the coding and debugging of the cochlear implant speech processing pipeline shown in Fig. 1. The main program code was written in Java to allow the use of Android APIs for the graphical user interface, data storage access, and audio recording capability. The use of Java also allowed easy threading of the program components. Aside from rewriting the code in floating-point, this constituted another major difference with the PDA implementation. Separate threads were created for the GUI, audio input, frequency domain transforms, noise classification, noise suppression, envelope computation, and file output. Real-time graphing was also performed using the AndroidPlot library.

The Android Native Development Kit (NDK) [14] was used to allow access to the NEON Media Processing Engine (MPE) [14, 15]. The NEON MPE is a SIMD coprocessor which supports both fixed and floating-point operations. Several code segments for performing floating-point vector computations such as scaling, dot product, and filtering were written in Assembly using the Java Native Interface (JNI).

Fig. 2 shows a snapshot of the configuration screen of the implemented pipeline on the smartphone. This settings configuration interface was devised to allow the parameters of the pipeline to be adjusted without needing to recompile the code. The configuration settings allow disabling NEON floating-point processing to simulate the performance of a smartphone not having the NEON coprocessor. The settings also allow disabling noise suppression and classification. An audio playback option is included to allow the user to hear the original sound when testing a recorded audio file. The tunable parameters are the microphone sampling rate, frame size, buffer length for the noise classification majority vote, and

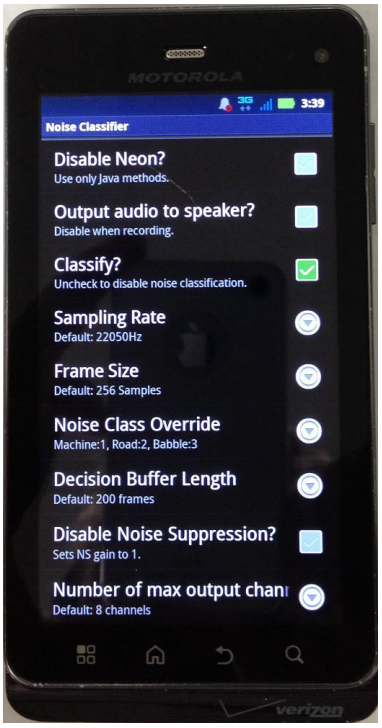

Figure 2. Settings screen of the cochlear implant speech processing pipeline on smartphone platform

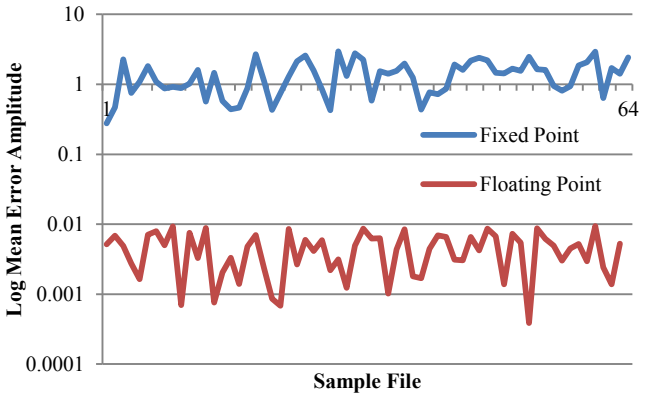

Fig.3. Log mean absolute error of fixed-point electrode pulse amplitudes over 64 unique 30 second audio clips

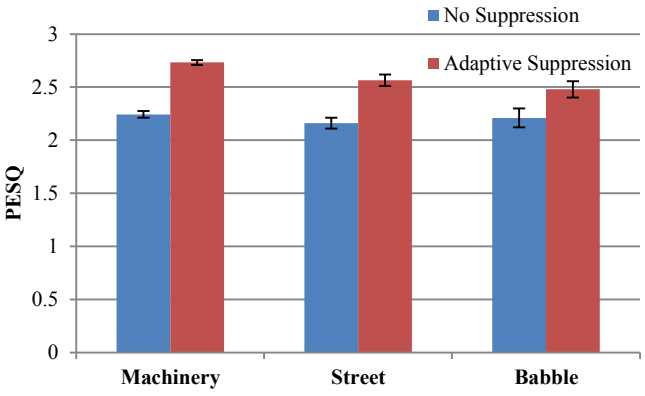

Fig.4. PESQ comparrison between adaptive noise supression and no noise suppression running on smartphone platform - error bars represent standard deviations

number of maximum amplitude channels to select for electrode pulse generation. Lastly, the user can select which channel from the electrode pulse amplitudes to graph on the smartphone screen.

The processing pipeline starts at the audio input stage where sampled audio data is either read from a file or accumulated from the smartphone microphone. In other words, the pipeline can be tested based on either pre-recorded signals or the microphone signals in real-time. An object is created for each frame of audio data which stores the audio samples as well as intermediate computation results. A blocking queue is used to connect each stage of processing to the next. Processing is linear, following each step in order before a program object is passed along to the next stage. At the end of the pipeline, debugging data are stored, along with sampled audio if the microphone input is used.

Previously, on the FDA-approved PDA platform, the processing was done using the fixed-point Q format to accommodate for the lack of a floating-point arithmetic unit on the platform. The floating-point number representation on the smartphone platform allows one to maintain a consistent arithmetic manipulation throughout the entire pipeline, rather than converting between different Q formats when higher precision is needed.

IV. Results and Discussion

A thorough analysis was performed to compare the operation of the smartphone implementation with the previous PDA implementation. The VAD and GMM decisions matched in the two implementations, correctly identifying speech segments and noise classes. The accuracy of the recursive wavelet packet transform represented by the mean squared error was obtained to be $1x10^{-5}$ for the fixed-point implementation and $1x10^{-8}$ for the floating-point implementation. The Q24 fixed-point implementation began to differ from the double-precision reference at the noise suppression and electrode pulse generation stages.

Mean absolute errors for sample audio files were calculated by averaging the absolute error of each pulse as per the following equation:

$$MAE = \frac{1}{NPC}\sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{c=1}^{C}\left|Y_{n,p,c} - Q_{n,p,c}\right| \qquad (1)$$

where Y denotes the signal under analysis and Q the double-precision reference signal, N the total number of sample frames in a file, P the number of pulses per frame, and C the number of channels. This error is shown in Fig. 3 where the mean absolute error for 64 different audio files is displayed in log scale. During the final low-pass filtering stage and channel selection, the error in the fixed-point implementation caused different maximum amplitude channels to be selected due to differing pulse amplitudes. Overall the errors in the fixed-point implementation were brief, lasting 10 to 20 frames in duration. In the smartphone implementation, the channel selection error was not present, and the negligible mean absolute errors were caused by rounding errors in the final data type conversion.

The widely used PESQ (Perceptual Evaluation of Speech Quality) speech quality measure [16, 17] was used to analyze the performance of the adaptive noise suppression in the pipeline. Fig.4 provides the PESQ comparison results between the adaptive noise suppression and no noise suppression cases on the smartphone platform. A dataset for this test was created based on the IEEE speech corpus consisting of 720 sentences [18]. The speech files were artificially corrupted with machinery, street, and babble noise with a SNR of 0 dB. The three sets of noise corrupted files were then subjected to the noise suppression running on the smartphone to get the enhanced speech files. As can be seen from Fig. 4, for all the noise environments, the adaptive noise suppression generated a higher PESQ score than the no suppression case.

Timing results for the three platforms of smartphone, PDA, and PC are shown in Table I. The PC target had a 3.33 GHz processor with 4 GB RAM and the PDA target had a 520 MHz processor with 64 MB RAM. On the smartphone platform, the processing time took 3.98 ms with NEON and 5.45 ms without NEON for one 256 sample frame of audio data at 22 kHz sampling frequency (11.6 ms of audio). The use of the NEON coprocessor led to a 27 percent decrease in

TABLE I. SPEECH PROCESSING PIPELINETIMING PROFILE (256-SAMPLE FRAMES AT 22 KHZ OR 11.6 MSEC FRAMES)

Processing Time in ms on	Entire Pipeline	Recursive WPT	Voice Activity Detector	Feature Extraction, Noise Classifier	Noise Suppression	Envelope Computation
Smartphone w/ NEON	3.98	1.10	0.22	0.21	1.77	0.68
Smartphone w/o NEON	5.45	1.13	0.46	0.73	2.07	1.06
PDA	8.41	1.24	0.91	2.03	2.40	1.83
PC	0.70	0.12	0.03	0.14	0.36	0.05

the overall computation time. Timing differences between the three target platforms are attributed mainly to clock speed differences, and to a lesser extent, to their memories. The recursive wavelet packet transform and the noise suppression did not gain much increase in computational efficiency from the floating-point implementation. The largest processing gains were seen in the VAD, MFCC feature extraction, noise classifier, and envelope computation. The increase in the computational efficiency of these modules was found to be due to the loops within these functions being replaced with the NEON SIMD code.

V. CONCLUSION

In this paper it was demonstrated that the previously developed cochlear implant speech processing pipeline could be implemented on a smartphone platform and be operated in real-time. Modification to the pipeline was done to allow higher precision floating-point data representation and architecture specific optimizations. Benefits gained from this new implementation are faster processing time, and greater computation precision. Further work is needed to properly feed the generated pulses by the smartphone platform into implanted electrodes. In essence, the real-time smartphone-based implementation presented in this paper makes smartphones a viable alternative to the FDA-approved PDA platform for cochlear implant studies.

ACKNOWLEDGMENT

This work was in part supported by a grant from the National Science Foundation (RAPD Award: CBET-0932542).

REFERENCES

[1] National Institute on Deafness and Other Communication Disorders, "Cochlear Implants," National Institutes of Health publication number 11-4798, http://www.nidcd.nih.gov/health/hearing/pages/coch.aspx, 2011.

[2] B. Fetterman and E. Domico, "Speech recognition in background noise of cochlear implant patients," *Otolaryngol. Head Neck Surg.*, vol. 126, no. 3, pp. 257-263, 2002.

[3] V. Gopalakrishna, N. Kehtarnavaz, T. Mirzahasanloo, and P. Loizou, "Real-time automatic tuning of noise suppression algorithms for cochlear implant applications," *IEEE Transactions on Biomedical Engineering*, vol. 59, pp. 1691-1700, 2012.

[4] Gartner. Inc., "Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time," [online] Aug 2013, http://www.gartner.com/newsroom/id/2573415 (Accessed : 3 May 2014).

[5] T. Mirzahasanloo, V. Gopalakrishna, N. Kehtarnavaz, and P. Loizou, "Adding real-time noise suppression capability to the cochlear implant PDA research platform," *Proceedings of 34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, San Diego, CA, Aug 2012.

[6] T. Mirzahasanloo, N. Kehtarnavaz, and I. Panahi, "Adding quiet and music detection capabilities to FDA-approved cochlear implant research platform," *Proceedings of 8th International Symposium on Image and Signal Processing and Analysis*, Trieste, Italy, Sept. 2013.

[7] V. Gopalakrishna, S.Yousefi,N.Kehtarnavaz, and P. Loizou, "Markov random field-based features for background noise characterization in hearing devices," presented at the 14th Appl. Stochastic Models Data Anal. Conf., Rome, Italy, 2011.

[8] Y. Hu and P. Loizou, "Environment specific noise suppression for improved speech intelligibility by cochlear implant users," *J. Acoust. Soc. Amer.*, vol. 127, no. 6, pp. 3689–3695, 2010.

[9] G. Kim and P. Loizou, "Improving speech intelligibility in noise using environment-optimized algorithms," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 8, pp. 2080–2090, Nov. 2010.

[10] V. Gopalakrishna, N. Kehtarnavaz, P. Loizou, and I. Panahi, "Real-time automatic switching between noise suppression algorithms for deployment in cochlear implants," *Proceedings of 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Buenos Aires, Argentina, Sept. 2010.

[11] V. Gopalakrishna, N. Kehtarnava, and P. Loizou, "A recursive wavelet based strategy for real-time cochlear implant speech processing on PDA platforms," *IEEE Trans. on Biomedical Engineering*, vol. 57, pp. 2053-2063, August 2010.

[12] V. Gopalakrishna, N. Kehtarnavaz, and P. Loizou, "Real-time implementation of wavelet-based advanced combination encoder on PDA platforms for cochlear implant studies," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process.*, 2010, pp. 1670–1673.

[13] Google. Inc., "Developer Tools," [online], https://developer.android.com/tools/index.html (Accessed: 3 May 2014).

[14] Google. Inc., "Android NDK," [online], https://developer.android.com/tools/sdk/ndk/index.html (Accessed: 3 May 2014).

[15] ARM, Inc., *Cortex™-A9 NEON™ Media Processing Engine Technical Reference Manual*, 2012.

[16] International Telecommunication Union, "Perceptual evaluation of speech quality (PESQ), and objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs," Technical Report, 2000.

[17] Y. Hu and P. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Trans. Speech Audio Process.*, vol. 16, no. 1, pp. 229–238, Jan. 2008.

[18] P. Loizou, *Speech Enhancement: Theory and Practice*. Boca Raton, FL: CRC Press, 2007.