Saremi *et al.*
EURASIP Journal on Audio, Speech, and Music Processing (2023) 2023:39
https://doi.org/10.1186/s13636-023-00305-7

EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH

Open Access

An acoustic echo canceller optimized for hands-free speech telecommunication in large vehicle cabins

Amin Saremi[1,2*], Balaji Ramkumar[3], Ghazaleh Ghaffari[1] and Zonghua Gu[1]

Abstract

Acoustic echo cancelation (AEC) is a system identification problem that has been addressed by various techniques and most commonly by normalized least mean square (NLMS) adaptive algorithms. However, performing a successful AEC in large commercial vehicles has proved complicated due to the size and challenging variations in the acoustic characteristics of their cabins. Here, we present a wideband fully linear time domain NLMS algorithm for AEC that is enhanced by a statistical double-talk detector (DTD) and a voice activity detector (VAD). The proposed solution was tested in four main Volvo truck models, with various cabin geometries, using standard Swedish hearing-in-noise (HINT) sentences in the presence and absence of engine noise. The results show that the proposed solution achieves a high echo return loss enhancement (ERLE) of at least 25 dB with a fast convergence time, fulfilling ITU G.168 requirements. The presented solution was particularly developed to provide a practical compromise between accuracy and computational cost to allow its real-time implementation on commercial digital signal processors (DSPs). A real-time implementation of the solution was coded in C on an ARM Cortex M-7 DSP. The algorithmic latency was measured at less than 26 ms for processing each 50-ms buffer indicating the computational feasibility of the proposed solution for real-time implementation on common DSPs and embedded systems with limited computational and memory resources. MATLAB source codes and related audio files are made available online for reference and further development.

Keywords Speech signal enhancement, Automotive speech processing, Acoustic echo cancelation, Adaptive filters, NLMS, Keyword spotting, Hands-free telephony, Automotive voice assistant

1 Introduction

1.1 Background

Acoustic echoes are generated in speech telecommunication networks due to acoustic feedback from loudspeakers to microphones. This phenomenon deteriorates the perception of sound by causing the users to hear a delayed replica of their own voice being reflected from the other side of the network. To remove acoustic echoes in earlier speech telecommunication systems, microphones were completely muted while the far end was talking over the loudspeaker. This simple method was efficient; however, it reduced a natural full-duplex dialog (i.e., simultaneous transmission in both directions) into a half-duplex (one direction) [1, 2]. Today, a less aggressive version of this technique, known as acoustic echo suppression (AES), is still used. This technique works by automatically reducing the gain on the microphone signal whenever the far end is talking to suppress the echo. The side effect of AES is, however, the nonlinearity in the

*Correspondence:
Amin Saremi
amin.saremi@umu.se
[1] Department of Applied Physics and Electronics, Umeå University, 901 87 Umeå, Sweden
[2] Department of Electronics, and Connected Services (EACS), Volvo Technology, 417 15 Göteborg, Sweden
[3] Department of Statistics and Machine Learning, Linköping University, Linköping 581 83, Sweden

system due to the automatic gain control. Another type of solution, categorized as acoustic echo cancelation (AEC), has been developed which dynamically estimates the acoustic echo path between the loudspeaker and the microphone in real time and subtract its effect from the captured signal so that only the desired near-end speech components remain [1–4]. AEC, unlike AES, is a fully linear solution, and it also conserves the full-duplex nature of the communication.

Although AEC might appear like a straightforward "system identification" problem, it bears several intrinsic challenges. Most of these challenges are because the acoustic echo path from the loudspeaker to the microphone is very dependent on the acoustical characteristics of the specific room where the near end is located. In many applications, the room's acoustic characteristics are either totally unknown or solely partially known. Moreover, these characteristics alter significantly due to object movements, temperature differences, and any relocation of the loudspeaker or the microphone. This means that there is generally not a generic time-invariant estimation of the acoustic echo path but rather a room-specific estimation that should be adaptively adjusted continuously [1–4]. Because of this, AEC algorithms need to impose a constant computational load on the embedded system that they run on. In many cases, the computational cost might become infeasible demanding a compromise between accuracy and computational efficiency [2–4].

Various model-based adaptive techniques have emerged to address the AEC problem: Least mean square (LMS) adaptive filter was used even in the early generation of the AECs [1], and several of its variants such as normalized LMS (NLMS) have been developed since then [2–4]. Other adaptive filtering approaches have been developed based on the recursive least square (RLS) method [2] and Kalman filters [5]. The RLS method constitutes a similar adaptive approach to LMS. However, the major distinction is that the RLS includes a "forgetting factor" (usually denoted by λ) which gives exponentially less weight to older error samples. The Kalman method is based on defining time-varying states for the filter properties which enable the system to adapt quickly to fast-varying changes in the echo path.

Besides the fourth mentioned model-based adaptive filters, probabilistic clustering techniques have been successfully used for blind source separation (BSS) of the echo components from the near-end speech signal [6]. However, BSS methods rely on the existence of multiple microphones (usually 4 microphones) for successfully estimating the location cues for source separation. Therefore, BSS methods are not applicable to most vehicles which are typically equipped with either single or dual microphones only.

Moreover, deep neural networks (DNNs) have also been deployed successfully to directly address acoustic echoes [7]. Some DNNs are also used for improving the parameterization of the existing model-based adaptive methods such as the step-size parameter in RLS [8] or the covariance of the state noise and observation noise in the Kalman filtering technique [9]. DNNs can only be used effectively in less time-critical applications that run on much larger computers. Consecutively, for real-time implementation on local embedded systems (DSPs) with limited computational and memory resources, adaptive filtering approaches (specifically NLMS) are still the most commonly used solutions due to their good balance between computational efficiency and accuracy [3, 4].

The performance of an AEC system is often assessed in accordance with the technical requirements announced by the International Telecommunication Union (ITU) G.168 standards for AECs [10]. These standards mainly focus on two metrics: (1) the echo loss achieved by the AEC which determines how much the acoustic echo is attenuated and (2) convergence time which determines how quickly the AEC addresses the acoustic echo. Besides ITU standards, word error rate (WER) during double-talk periods is also used as a benchmarking metric to evaluate the success of the AEC system. This metric is specifically more interesting if the AEC system is supposed to be used as a front end for automatic speech recognition (ASR) applications [8, 11].

1.2 AEC in the automotive context

The main application of AEC in the automotive industry is related to Bluetooth-based hands-free telephony. Bluetooth-based hands-free telephony systems were introduced into the vehicle's infotainment systems in the 1990s enabling drivers to make phone calls wirelessly using the vehicle's embedded microphone and loudspeaker while driving [12]. The common practice for echo cancelation in vehicles is to use an NLMS-based adaptive AEC combined with a nonlinear AES as a post-processor to further attenuate the residual echoes [2, 12, 13].

Although echo cancelation has become relatively successful in personal cars despite some common failures [14], it has proved very challenging in large commercial vehicles. One main reason is the physical attributes of these vehicles, e.g., a truck cabin could be three to five times larger than a personal vehicle's cabin leading to much longer reverberation times. Furthermore, relative to personal cars, the geometry and the furnishing in truck cabins can hugely vary across manufacturers and from one model to another. As a result, creating a generic model of the acoustic echo path in truck cabins is challenging and typically requires longer adaptive filters which usually impose an unaffordable computational

load on the vehicle's embedded system. Consequently, truck producers often receive a large number of customer complaints about low audio quality due to echo issues. These complaints are important given that a typical commercial truck driver might make several telephone conversations a day to arrange the logistics of their deliveries. Moreover, truck drivers could cause serious safety risks for themselves and other road users if they resort to using their phones while driving due to frustration with the audio quality of the hands-free telephony system.

Besides hands-free telephony, the speech dialog system is the other major application that requires echo cancelation. Mercedes-Benz launched the first speech dialog system in 1996 which allowed the driver to interact with the vehicle functions using their voice [13, 15]. Since then, many automotive brands have adopted similar systems in their products [13]. All of these conventional speech dialog systems are half-duplex because the user needs to push a specific physical button on the steering wheel (known as the *push-to-talk* (PTT) button) each time they want to talk to the system [13, 15]. When the user pushes the PTT button, the loudspeakers are immediately muted and thus no echo is generated.

In contrast, voice assistant systems (such as *Google Assistant*, *Siri*, and *Amazon Alexa*) have been developed in recent years that are independent of the PTT button and rather rely on specific words (known as *keywords*) to trigger the application [16, 17]. These are for instance "OK Google!" for Google Assistant, "Alexa!" for Amazon Alexa, and "Hey Siri" for Siri. These voice assistants aim to provide full-duplex natural communication in the sense that they allow users to interrupt ("barge in") using the keyword while the system is reading information over the loudspeakers. This is a vital ingredient in the desired user experience and, to achieve that, acoustic echoes need to be constantly canceled during the entire operation of these voice assistants. Google and Amazon publicly announced that they will introduce their voice assistants in vehicles with the first products launched in 2020 [16, 17].

The key difference between a hands-free telephony system and a speech dialog system is that the far-end party in the former is a human, whereas in the latter, the far end is a neural network performing the speech recognition task. A major requirement for speech recognition is that the input audio stream provided is free of nonlinearity [8, 18]. As a result, unlike for hands-free telephony, the echo cancelation for such systems can solely rely on AEC, and no AES post-processing is allowed. This limitation places a higher burden on echo canceling systems that are designed to work for both hands-free telephony and speech dialog systems.

1.3 Our contribution

1) AEC in large vehicle cabins is challenging which leads to substantial customer complaints reported to truck manufacturers. We provide an adaptive time-domain AEC solution tailored for large vehicle cabins and tested it on four main Volvo truck models. The solution achieves and surpasses the quality standards required by ITU G.168 standards.
2) Our AEC solution provides a good balance between accuracy and computational feasibility. The AEC algorithm runs only when it is deemed needed. A real-time version of the solution was coded in C and implemented on a very low-cost commercial DSP platform with limited computational and memory resources.
3) The proposed AEC solution is fully linear which makes it a good lightweight frontend for ASR systems and more specifically keyword spotting algorithms, which typically need to run locally on the host device, i.e., an embedded system with limited memory and computational resources.
4) The successful features mentioned above are only achievable by means of carefully optimizing four key parameters, namely (1) the length of the Wiener filter, (2) the regularization parameter of the step size (β), (3) the correlation threshold of the statistical double-talk detector (η_0), and (4) the energy threshold for voice activity detection (E_0). In this work, we describe how these four parameters could be successfully optimized. To our knowledge, our work is unique in explicitly describing and presenting these tailored optimizations for AEC inside large vehicle cabins.

The paper provides information on the used methods, the performance, the computational efficiency, and the DSP implementation of the proposed solution, in the following sections.

2 Methods

The proposed AEC solution was implemented and evaluated in MATLAB v.2019b. The real-time implementation of the AEC solution was performed in *C* language using uVision Keil environment on a common ARM Cortex M7 DSP. All participants in the tests consented to the anonymous use of their voices, in compliance with applicable ethical regulations.

2.1 NLMS adaptive filtering

Figure 1A shows an overview of the AEC setup. The signal received from the far-end party over the communication "downlink" port is denoted by *x[n]*. This signal is also

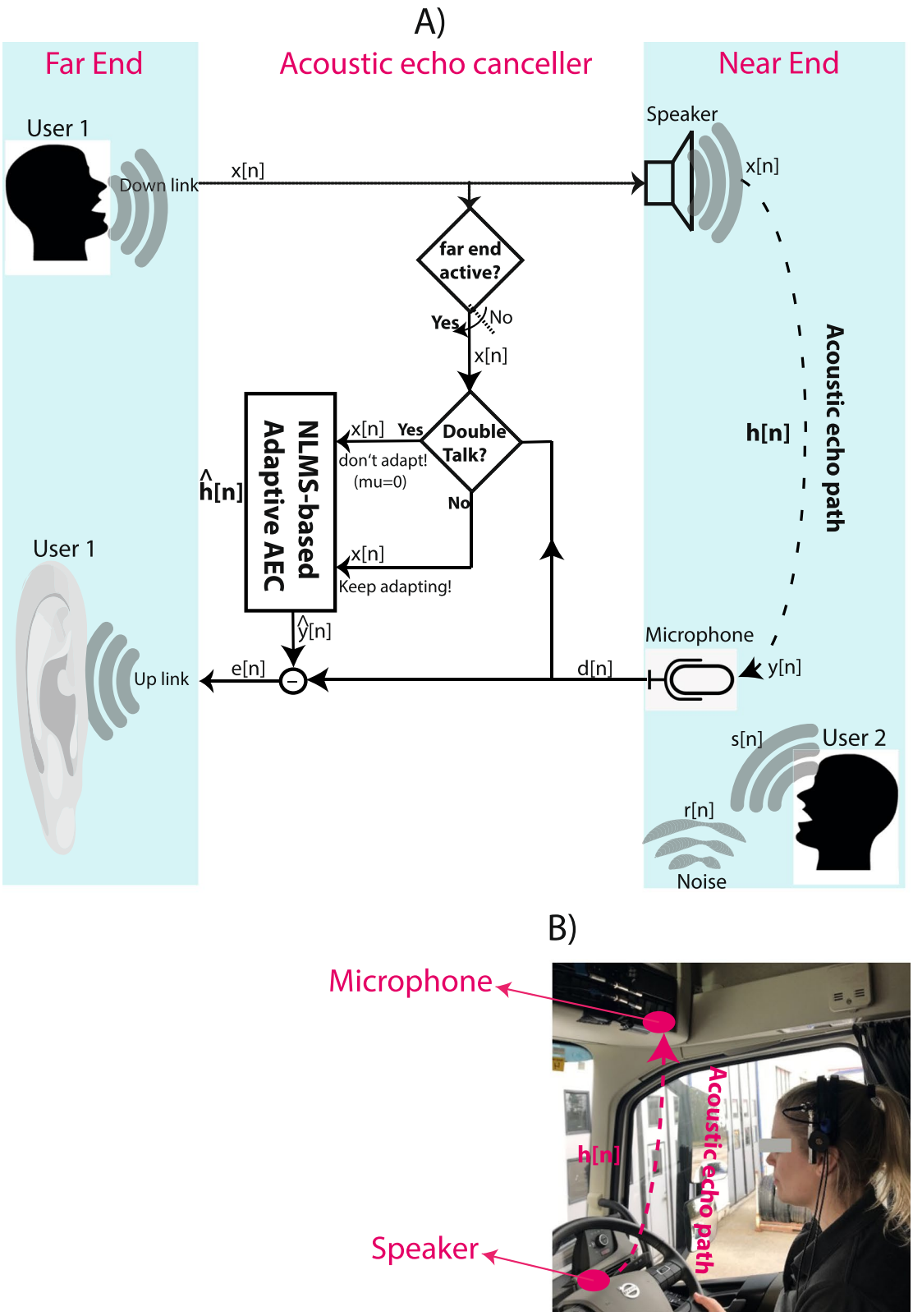

Fig. 1 **A** An overview of the proposed AEC algorithm which consists of the main "NLMS-based adaptive AEC" block accompanied by two decision circuits, i.e., "far-end voice activity detector" (VAD) and "double-talk detector" (DTD). The AEC algorithm runs only whenever the far end is active (i.e., "YES" switch is activated). The double-talk detector stops the AEC adaptation (sets μ =0 in Eq. 2.b) if double talk is detected. **B** A driver in the FM928 Volvo truck serving as the near-end party during a hands-free phone conversation

called "the reference signal" since it is used as the reference for the adaptive algorithm. After this signal propagates through the near-end acoustic environment (i.e., the truck cabin), the microphone receives *y[n]* which is an echo of *x[n]* that has been filtered by the corresponding echo path impulse response. The microphone also receives *s[n]*, i.e., components of the speech signal produced by the near-end talker as well as some ambient noise components represented by *r[n]* in Fig. 1A. The signal captured by the microphone (*d[n]*) is then an addition of these three signals (i.e., $d[n] = y[n] + s[n] + r[n]$). All these signals are considered to be zero-mean, real, statistically independent of each other, and wideband (i.e., convey frequencies from 0.1 to 8 kHz). In this section, we begin by presenting the AEC concept based on the assumption that the near-end talker is silent (*s[n]*=0) and the ambient noise is absent (*r[n]*=0). Therefore, *y[n]* is the sole signal received by the microphone, i.e., *d[n]*=*y[n]*. The effects of the near-end speech (*s[n]*), and ambient noise (*r[n]*) are added to the model and analyzed in subsequent sections.

Figure 1B shows a driver in an FM928 truck during a hands-free phone conversation serving the role of the near-end talker in Fig. 1A. The microphone and the front loudspeaker are marked by red circles while the dashed line represents the corresponding acoustic echo path. The AEC algorithm assumes that the acoustic echo path between the loudspeaker and the microphone is a linear system that could be simulated by a finite impulse response (FIR) filter, with its impulse response denoted by *h[n]*, at any given time. This impulse response consists of *L* coefficients which corresponds to *Fs/L* seconds in time, where *Fs* is the sampling rate.

The true impulse response of the acoustic echo path (i.e., *h[n]* in Fig. 1A) is unknown, and the task of an AEC solution is to estimate it. To do so, the NLMS algorithm is an iterative process that tries to find an impulse response (represented by $\widehat{h}[n]$ in Fig. 1A) that ideally matches the true impulse response of the acoustic path (*i.e.*$\widehat{h}[n] \cong h[n]$). As a result, once the algorithm finds this impulse response, the output of the adaptive filter ($\widehat{y}[n]$) approaches the received signal by the microphone. In other words, the error signal (denoted by *e[n]* in Fig. 1A), which is the difference between $\widehat{y}[n]$ and *y[n]*, approaches zero (i.e.,$e[n] = y[n] - \widehat{y}[n] \cong 0$). The NLMS algorithm slightly adjusts the coefficients of the $\widehat{h}[n]$ in each iteration in order to minimize the error signal (*e[n]*). In this sense, $\widehat{h}[n]$ acts as an adaptive Wiener filter in this system identification problem [2–4].

The output of the adaptive filter ($\widehat{y}[n]$) is given by Eq. (1.a) below where*L* tap of *x[n]* is transposed (represented by ${x_L}^T[n]$) and is then multiplied by $\widehat{h}[n]$. Equation (1.b) provides a short form of Eq. (1.a) where * denotes convolution.

$$\widehat{y}[n] = [x(n)x(n-1)...x(n-L-1)]^T[\widehat{h}[1]\widehat{h}[2]\ldots\widehat{h}[L]] \tag{1.a}$$

$$\widehat{y}[n] = x[n] * \widehat{h}[n] \tag{1.b}$$

The adaptation process occurs by estimating a new $\widehat{h}[n]$ per each sample of *y[n]* through a small adjustment $\Delta\widehat{h}[n]$ in each iteration, as expressed in Eq. (2.a, 2.b and 2.c). This adjustment value is determined based on the error signal (*e[n]*) and the reference signal (*x[n]*) according to Eq. (2.b) where $\mu[n]$ is known as "step size." Choosing an optimized step size is important for the convergence rate and accuracy of the system and has been studied extensively in the literature [2–4, 19]. According to the NLMS method, the step size is adaptively changed in each iteration proportional to the variance of $x_L[n]$, denoted by $\sigma^2_{x_L[n]}$, as presented by Eq. (2.c) [2–4].

$$\widehat{h}[n+1] = \widehat{h}[n] + \Delta\widehat{h}[n] \tag{2.a}$$

$$\Delta\widehat{h}[n] = \mu[n](x_L[n]e[n]) \tag{2.b}$$

$$\mu[n] = \frac{\alpha}{\beta + \sigma^2_{x_L[n]}}, \sigma^2_{x_L[n]} = \mathrm{var}(x_L[n]) = {x_L}^T[n]x_L[n] \tag{2.c}$$

The terms α and β need to be adjusted according to the specifics of every given NLMS problem. Choosing appropriate values for α and β has been comprehensively studied to a great complication [2–4, 19]. The term α, which could take a value between 0 and 2, determines the speed of convergence. Higher α values result in quicker adaptation of the NLMS algorithm; however, there is a tradeoff between convergence and overall success of the echo canceller in terms of echo return loss enhancement (ERLE) [3]. Here, we chose $\alpha = 1.98$ to assure the fast convergence of the algorithm. The term β, known as the regularization parameter, is meant to improve the performance of the NLMS in the presence of noise, and it has to be adjusted according to the characteristics of the ambient noise (*r[n]* in Fig. 1) and the intrinsic signal-to-noise ratio (SNR) of the microphone hardware [3]. Here, we chose $\beta = 0.1$ which corresponds to the intrinsic SNR of the electret condenser microphones that are commonly used in the automotive industry. The Wiener adaptive filtering method has been successfully implemented in vehicles for non-stationary noise cancelation, e.g., by [20].

2.1.1 Choosing the length of the Wiener filter

The length of $\widehat{h}[n]$, represented by L, has a great impact on the outcome of an AEC, and it is related to the reverberation in the near-end acoustic space [2]. If the goal of the AEC is to achieve an ERLE of 30 dB, then L should be chosen long enough to cover the T30 reverberation time (i.e., the time it takes for sound to decay 60 dB after the source is muted). Previous studies have shown that setting the length much longer than the corresponding reverberation time not only imposes unnecessary excess computational cost but also increases numerical errors [3, 4, 19]. T30 has been measured around 40 ms in medium-sized trucks (Volvo FM series) and estimated up to 60 ms in larger heavy-duty trucks (Volvo FH series). Here, we use the average of these two numbers (i.e., 50 ms) to have a generic solution for both series. Therefore, at Fs = 16 kHz, the Wiener filter, $\widehat{h}[n]$, would have 800 coefficients ($L = Fs \times T30 = 16000 \times 0.05 = 800$). To achieve a tailored solution, it is reasonable to allow automakers to set L based on the T30 reverberation time associated with the cabin geometry of every particular vehicle model.

2.2 Double-talk detection based on normalized cross-correlation

The method presented in the previous section was built on the assumption that the near-end talker is silent ($s[n] = 0$) while the echo from the far end is present. However, in natural full-duplex communication, the near-end talker might as well talk at the same time as the far end (i.e., double talk, DT). Moreover, a non-negligible amount of ambient noise ($r[n]$) might also be present. In these situations, the captured microphone signal ($d[n]$) conveys components that are unrelated to the acoustic path, $h[n]$, and therefore, the underlying modeling principal of the AEC becomes violated, and consecutively, the adaptive process would diverge and fail [2, 3, 21–23]. To avoid this scenario, as illustrated in Fig. 1A, our solution constantly monitors the signals $x[n]$ and $d[n]$ for detecting the occurrence of DT. In case DT is detected, as shown in Fig. 1A, the adaptation process is halted by setting $\mu = 0$ in Eq. (2.b). By doing so, $\Delta\widehat{h}[n] = 0$ which means that no new impulse response will be estimated as long as a DT is ongoing but, instead, the last valid $\widehat{h}[n]$ will be applied to the signal. This mechanism prevents DT from contaminating the process of new $\widehat{h}[n]$ estimation.

Different methods have been developed to detect DT and stop the adaptive process until the end of the DT period. These methods can be categorized into two groups: (1) methods that detect DT by comparing the amplitude of the captured signal ($d[n]$) with the far-end reference signal ($x[n]$) [21, 22] and (2) methods that detect the DT by analyzing the statistical differences between $d[n]$ and $x[n]$ [5, 23].

While amplitude-based DTDs are easier to implement and are more computationally efficient than statistical DTDs, their main shortcoming is that the sound level produced by the loudspeaker is in many cases unknown, and thus, comparing the amplitudes of $x[n]$ and $d[n]$ might lead to a wrong decision by the DTD. This is especially an important shortcoming for vehicle applications wherein the user is capable of increasing the loudspeaker's gain at any time and thereby manipulating the amplitude of $d[n]$.

Therefore, we developed a statistical DTD decision circuit based on the normalized cross-correlation (NCC) between $x[n]$ and $d[n]$ [23]. NCC is also called the "Pearson correlation coefficient" in statistics [24]. In case the far-end is the only talker, there will be a non-zero cross-correlation between $x[n]$ and $d[n]$. However, when the near end talks too (i.e., DT occurs), the cross-correlation between $x[n]$ and $d[n]$ diminishes and approaches zero since $d[n]$ would convey $s[n]$ components as well. Accordingly, DT is detected if NCC drops below a certain threshold (η_0).

Equation (3) presents the NCC between $x[n]$ and $d[n]$ where $\sigma_{x_L[n]}$ and $\sigma_{d_L[n]}$ are the standard deviation (square root of variance) of L samples of $x[n]$ and $d[n]$, respectively, and $\mathrm{cov}(x_L[n], d_L[n])$ is the covariance between them. NCC can yield a number in the range $[-1, +1]$, where $+1$ indicates a perfect correlation and -1 is a perfect anti-correlation between the two inputs while 0 shows a non-existing correlation.

$$\mathrm{NCC}(x_L[n], d_L[n]) = \frac{\mathrm{cov}(x_L[n], d_L[n])}{\sigma_{x_L[n]}\sigma_{d_L[n]}} \tag{3}$$

2.2.1 Optimizing the DTD threshold (η_0)

DT is detected when NCC falls below a certain threshold (η_0) and approached zeros. Choosing an appropriate η_0 is vital to assure accurate DTD and avoid false alarms [23]. There are four possible scenarios: (1) both far-end and near-end parties are silent, (2) the far end is talking meanwhile the near end is silent, (3) the far end is silent meanwhile the near end is talking, and (4) both far-end and near-end parties are talking (DT). Scenarios 1 and 3, wherein the far end is silent, are taken care of by the VAD which is explained in the next section. Therefore, η_0 should be chosen to enable the system to distinguish the last scenario (i.e., scenario 4) from the scenario 2 above. NCC is always higher for scenario (2) than for scenario 4 in every situation. Accordingly, η_0 should be chosen around the minimum of the NCC for scenario

2. Every NCC that falls below η_0 would then indicate that scenario 4 has occurred (i.e., the DT is detected).

The truck's loudspeaker is not ideal and thus constantly produces a low-intensity wideband noise that travels through the acoustic path and leaks to the captured signal by the microphone. We symbolize this particular noise signal by$v[n]$. In case of scenario 2, the signal captured by the microphone ($d[n]$) consists of two components (i.e.,$d[n] = y[n] + v[n]$). The NCC between $d[n]$ and $x[n]$ in this case is $\mathrm{NCC}(x_L[n], d_L[n]) = \frac{\mathrm{cov}(x_L[n], d_L[n])}{\sigma_{x_L[n]}\sigma_{d_L[n]}}$ $= \frac{\mathrm{cov}(x_L[n], y_L[n]+v_L[n])}{\sigma_{x_L[n]}\sigma_{d_L[n]}}$. According to the principle of bi-linearity of covariance, this term can be written as $\frac{\mathrm{cov}(x,y)+cov(x,v)}{\sigma_x\sigma_d}$. Given that the minimum possible value for $\mathrm{cov}(x, y)$ is 0 (which occurs when the far end is talking at very low intensities), η_0 can be then calculated according to Eq. (4) below.

$$\eta_0 = \min\left(\frac{\mathrm{cov}(x, v)}{\sigma_x\sigma_d}\right) \tag{4}$$

We calculated η_0 defined by Eq. (4) above in all of our audio files recorded in various truck models. The results yielded 1.1×10^{-4} in average with a standard deviation of $\mathrm{SD} = 0.5 \times 10^{-5}$. This choice of η_0, as seen in Eq. (4), is dependent on the specific statistical relations between the loudspeaker noise and the captured microphone signal. This can vary based on the loudspeaker hardware and cabin geometry. Our verification tests have shown that choosing this threshold results in a low false alarm (pf) of lower than 0.1 across all truck models, as recommended by [23], while a substantial majority of DTs are appropriately detected. Our DTD optimization presented in this section is specifically targeting large vehicle cabins and truck models (with their typical loudspeaker hardware and cabin geometry), but our method can be generalized and used to optimize η_0 for any type of cabin and loudspeaker hardware.

2.2.2 Ambient noise

The main contributor to the ambient noise is the noise generated by the engine. The engine noise recorded in Volvo truck models is mostly a low-frequency noise with a peak at its fundamental frequency of about 30 Hz and decaying harmonics that can be detected up to about 1 kHz. The fan noise, if the air conditioner is on, and road noise, if the vehicle is in motion, are other contributors to *r[n]*. All these sources are uncorrelated with the speech signals ($\mathrm{cov}(r, s) \cong 0$), and they are low intensity ($\sigma_r \cong 0$). In other words, the NCC between these noise components and the speech signals must be close to zero [i.e., $\mathrm{NCC}(x_L[n], r_L[n]) \cong 0$, and $\mathrm{NCC}(s_L[n], r_L[n]) \cong 0$].

2.3 Far-end voice activity detection

The AEC is a necessity when there is a meaningful amount of echo in the system, but when the far-end reference signal is empty (i.e., *x[n]* does not convey significant audio components to be canceled), the AEC becomes a computationally expensive unnecessary burden. To address this issue, as illustrated in Fig. 1A, we constantly monitor the reference signal (*x[n]*) for relevant content with regard to echo cancelation and run the AEC and DT algorithms only when necessary.

2.3.1 Optimizing the VAD threshold (E_0)

In hands-free telephony applications, the far-end signal comes from a microphone mounted on a phone, often from a mobile phone. Google has announced in its requirements for Android mobile phone manufacturers that the analog signal captured by the microphone should be sampled at 16 bits and should yield specific sensitivity to sounds [18]. These requirements are the industry standards and are generally followed by non-Android phone producers as well.

The quantization error associated with a 16-bit sampling analog-to-digital converter is −96.3 dB FS which implies that sound components that are lower than 96.3 dB in intensity from the full scale (FS) fall under the quantization noise floor [25]. It is reasonable to assume that there is some imperfection in the microphone circuitry used on the far-end side and some irrelevant background noise that can mount to −80 dB FS. We, therefore, set E_0 to 10^{-4}, which corresponds to −80 dB FS. Accordingly, any L tap of the reference signal ($x_L[n]$) that conveys root mean square (RMS) energy below this level is considered irrelevant (i.e., containing virtually zeros). As Fig. 1A shows, the AEC and DTD algorithms are not applied on such taps and the microphone signal is passed to the output without any processing ($e_L[n] = d_L[n]$). Our evaluations show that by stopping AEC and DTD in these circumstances, a considerable amount of unnecessary computation is avoided. The proposed VAD proves vital for any realistic real-time implementation of our AEC solution on a commercial DSP.

2.4 Evaluation

Data were recorded in four common large commercial vehicles produced by the Volvo group: two medium-duty trucks from the FM series (FM928 and FM930), and two heavy-duty trucks from the FH series (FH2250 and FH2099). These truck models have various cabin geometries that represent the majority of trucks sold worldwide. Each recording consisted of 2 min of communication between far-end and near-end parties over the phone.

The far-end party talked on an HTC-11 phone while the near-end party used an Apple iPhone 11 connected to the head unit over Bluetooth. The phone volume was set to maximum on both phones and maintained throughout the tests. The truck's loudspeaker's volume was set to produce a comfortable and clear sound level (volume = 25) although the tests.

Both far-end and microphone signals, *x[n]* and *d[n]* respectively, were recorded by the vehicle's head unit and saved in the memory. Each recording started with the far end reading a list of 10 special sentences from the standard Hearing-in-noise test (HINT). These sentences are short everyday sentences that have been phonemically balanced and are currently used widely in audiological research setups for measuring speech recognition thresholds in humans [26]. During the time the far end was reading the HINT sentences, the near end was silent. Afterward, there was a natural multi-turn conversation between the two parties followed by a DT wherein both near-end and far-end parties articulated numbers simultaneously.

The recording was done while the vehicle was stationary under two scenarios: (1) engine off or (2) engine on. The recorded files were in *.Wav format and originally sampled at 48 kHz. These data were imported to MATLAB v.2022(a) and down-sampled to 16 kHz for analysis.

The algorithm shown in Fig. 1A and explained so far was coded in MATLAB and applied to the two files *x[n]* and *d[n]* using an iCore7 Intel computer and the output, *e[n]*, was saved. For each case, the time it takes for the code to run and finish the task was also assessed using the "*tic, toc*" instruction in MATLAB.

2.4.1 Echo reduction loss enhancement as a measure of the AEC performance

ERLE is a commonly used indicator for quantifying the achievement of an AEC solution to attenuate echoes [2–4, 6, 7, 10]. To estimate the ERLE achieved by the proposed solution, a segment of the signals corresponding to the part wherein the HINT sentences are read by the far-end participant is extracted and the ERLE is estimated using Eq. (5) below [10] when the far end is the sole talker ($s[n]=0$).

$$\mathrm{ERLE} = 10 \times \log_{10}^{\frac{\sigma^2_{d[n]}}{\sigma^2_{e[n]}}} \tag{5}$$

2.4.2 Convergence time as a measure of the AEC performance

The convergence time is another important quantitative measure of an AEC success [2–4, 6, 7, 10]. It refers to the time it takes for the AEC to reach a specific amount of ERLE, usually 25 dB, and stay over that afterward [2–4, 6, 7, 10]. Here, the convergence time is calculated using the Swedish HINT sentence. According to the requirements stated by ITU, an AEC solution should yield at least 6 dB of ERLE at the second frame (L). The ERLE should then increase to a minimum of 20 dB at 1 s. Thereafter, the ERLE should reach its steady state at 10 s and should stay over that steady state value, afterward [10].

2.4.3 Word error rate as a measure of the AEC performance in DT periods

WER is a standard metric to measure the speech intelligibility of a speech signal [8, 11]. During the DT, the system is supposed to stop adapting but still be able to cancel far-end echo components while preserving the near-end speech signal. WER during DT periods is a good measure for assessing the system's ability in preserving the near-end speech quality in the presence of simultaneously competing far-end speech components. The WER calculates the Levenshtein distance of the recognized word sequence from the ground truth transcription divided by the number of words in the ground truth transcription. It is defined as $WER = \frac{S+D+I}{N}$ where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the number of words in the ground truth transcription.

To assess the performance of our solution in improving speech intelligibility during DTs, we fed the original microphone recording ($d[n]$) during the periods where both near-end and far-end parties simultaneously read out numbers into Google's speech-to-text API, which is freely available online. The WER was calculated for the recognized text. We also fed in the error signal ($e[n]$), which is the output of our AEC processing, to the same speech-to-text API and likewise calculated the WER. We compared the two WER scores (original versus processed) and reported as a measure of our solution's success in increasing the speech intelligibility during challenging DT periods.

2.5 Real-time implementation

To assess the computational feasibility of the proposed solution for real-time embedded implementation, it was implemented in *C* language on an evaluation board provided by ST [27] using the standard CMSIS libraries and uVision Keil, which is a commercial integration development environment (IDE) provided by ARM for compiling *C* code into the machine code for ARM microprocessors. This evaluation board contains a standard ARM Cortex-M7 microprocessor, called STM32F767, manufactured by ST microelectronics. This microprocessor is equipped with DSP capabilities and libraries. This microprocessor's central processing unit (CPU), when clocked at 216 MHz

(rated maximum), provides a computation power of 462 Dhrystone million instructions per second (DMIPS). It comes with 2 Mbytes of internal flash memory and 512 kB of volatile RAM [27].

A new project was created in the uVision Keil IDE. The reference far-end signal (*x[n]*) and the microphone signal (*d[n]*) were down-sampled to 16 kHz and saved as *.Wav files in the project user data space. An open-source code was used to read the files in STM32F767 with buffers of size 800 which matches the filter length (L) and the tap size used throughout this work.

A key concept in real-time signal processing is that the algorithm processes the data without being interrupted for reading the input data from memory. To secure this, the direct memory access (DMA) unit of the microprocessor should be configured and used [28]. The DMA unit runs on an independent clock from that of the CPU, which allows the microprocessor to apply the algorithm and process a buffer while the next buffer is being read from the memory in parallel. Two buffers (B1 and B2 in Fig. 2) are used for reading *x[n]* data. When the first buffer (B1) is filled, an interrupt is generated. The buffer is then processed by the CPU. While the buffer is being processed, the new samples are read and transferred by the DMA into the second buffer (B2) to be fetched and processed by the CPU in the next cycle. This procedure continues and the data are filled in B1 and B2 and fetched by the CPU consecutively. This double buffering mechanism, empowered by the DMA, guarantees a continuous real-time flow and processing of the data. The same procedure is also performed to read the microphone data (*d[n]*) using buffers C1 and C2, as seen in Fig. 2.

The described echo canceling algorithm (Fig. 1A) is applied to each buffer, and the output (*e[n]*) is sent to the digital-to-analog converter (DAC) of the microprocessor which then sends it via an analog operational amplifier (LM4871) to a 4-W loudspeaker to play the result. This analog amplifier is supplied by 3.3 V DC and is configured according to the circuit in Fig. 2. The analog amplifier amplifies and broadcasts the result on a loudspeaker to be heard.

We assessed the computational efficiency of the implementation by measuring the algorithmic latency (i.e., the time it takes for the CPU to process a buffer) using the debugging tools available in the Keil uVision environment. A key requirement for a real-time system to function properly is that the algorithmic latency is less than the length of the buffer itself (i.e., 50 ms here). If this requirement fails, a phenomenon known as "CPU overrun" occurs whereby the continuous flow of the data from input to output is disrupted [28]. Through our code development, we used 'optimization level-2 for speed' option in our tool chain to guide the compiler to generate underlying assembly codes that are optimized for enhancing the execution speed on our particular microprocessor CPU.

3 Results

3.1 Echo attenuation and convergence time in absence of engine noise

Figure 3A shows the microphone signal (*d[n]*) versus the output of the AEC (*e[n]*) for an FM928 truck while the engine is off. The AEC solution manages to attenuate the echo received by the microphone significantly by 25.54 dB according to Eq. (5). Figure 3B shows the ERLE per sentence and how the ERLE becomes stronger as the algorithm continues adapting.

Figure 4 shows the ERLE per HINT sentences and, additionally, at 0.1 and 1 s (marked by crosses). The dashed line shows the convergence requirement announced in

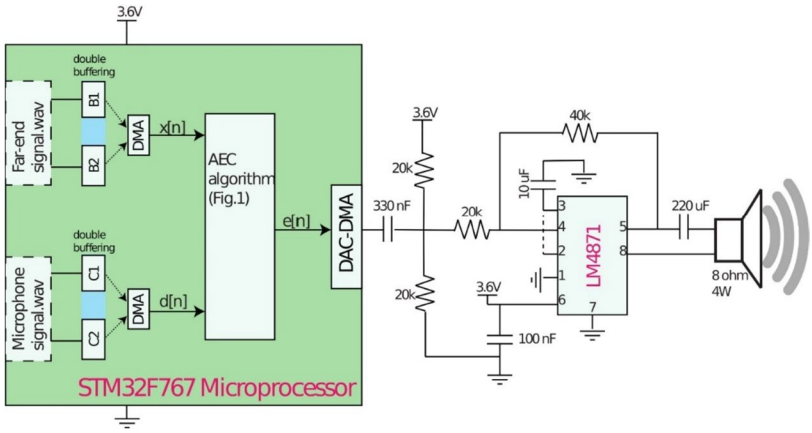

Fig. 2 An overview of the real-time implementation on a Cortex M-7 microprocessor (STM32F767) accompanied by an analog amplifier circuitry and a loudspeaker

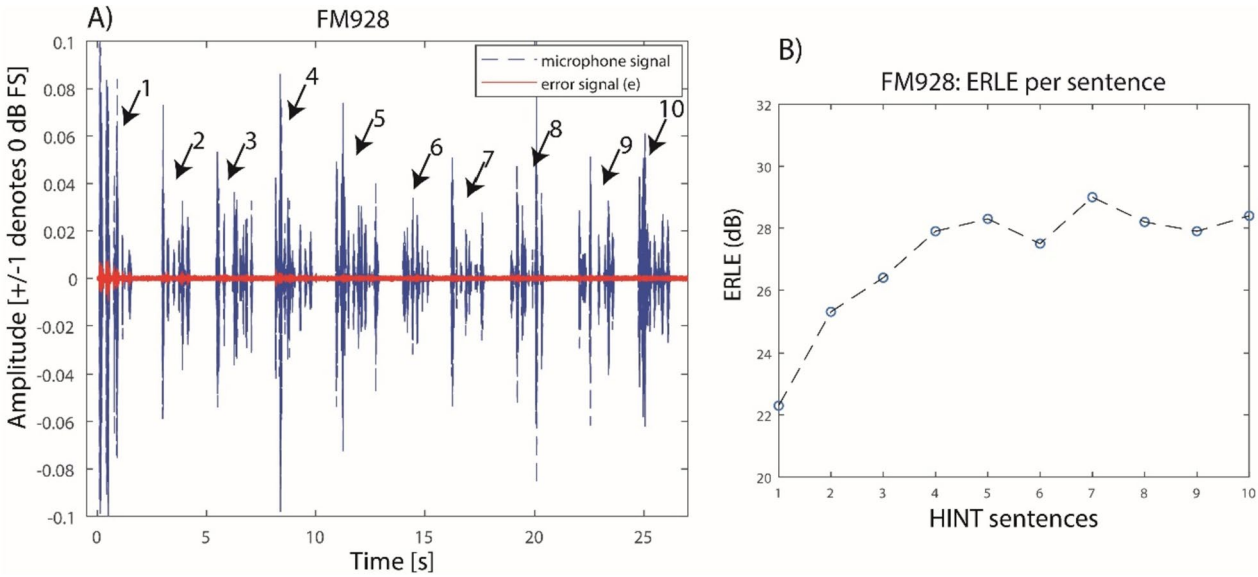

Fig. 3 **A** Captured microphone data ($d[n]$) versus the output of the AEC, error signal ($e[n]$) while the far end is reading the HINT sentences (altogether 10 sentences). The sentences are marked by numerical indicators. **B** The echo attenuation achieved by the presented AEC solution in terms of ERLE per sentence

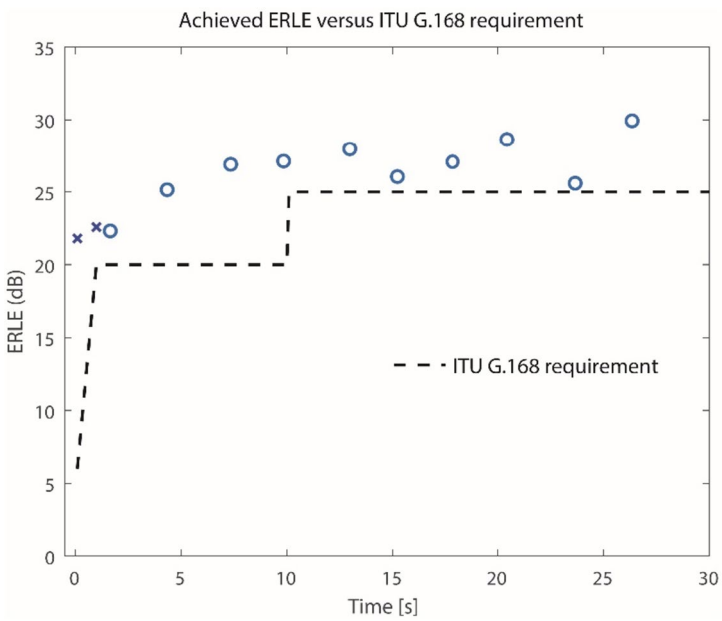

Fig. 4 The ERLE convergence achieved by the presented AEC method as a function of time (circles and crosses) versus the requirement stated by ITU G.168 standards (dashed)

Fig. 12(A) of ITU G.168 standard for AECs [10]. Based on this requirement, the AEC should yield ERLE values that lie above the dashed line. Accordingly, the AEC should yield at least 6 dB of ERLE at the second frame (since each buffer is 50 ms in our solution, this means 0.1 s). The ERLE should then increase to a minimum of 20 dB at 1 s. Thereafter, the ERLE should reach its steady state at 10 s and should stay over that steady state value, afterward. Figure 4 shows that the presented AEC solution achieves over 22 dB of ERLE at 0.1 s which is much higher than the requirement (6 dB). This demonstrates the very fast convergence of the AEC because it already reaches 22 dB in a

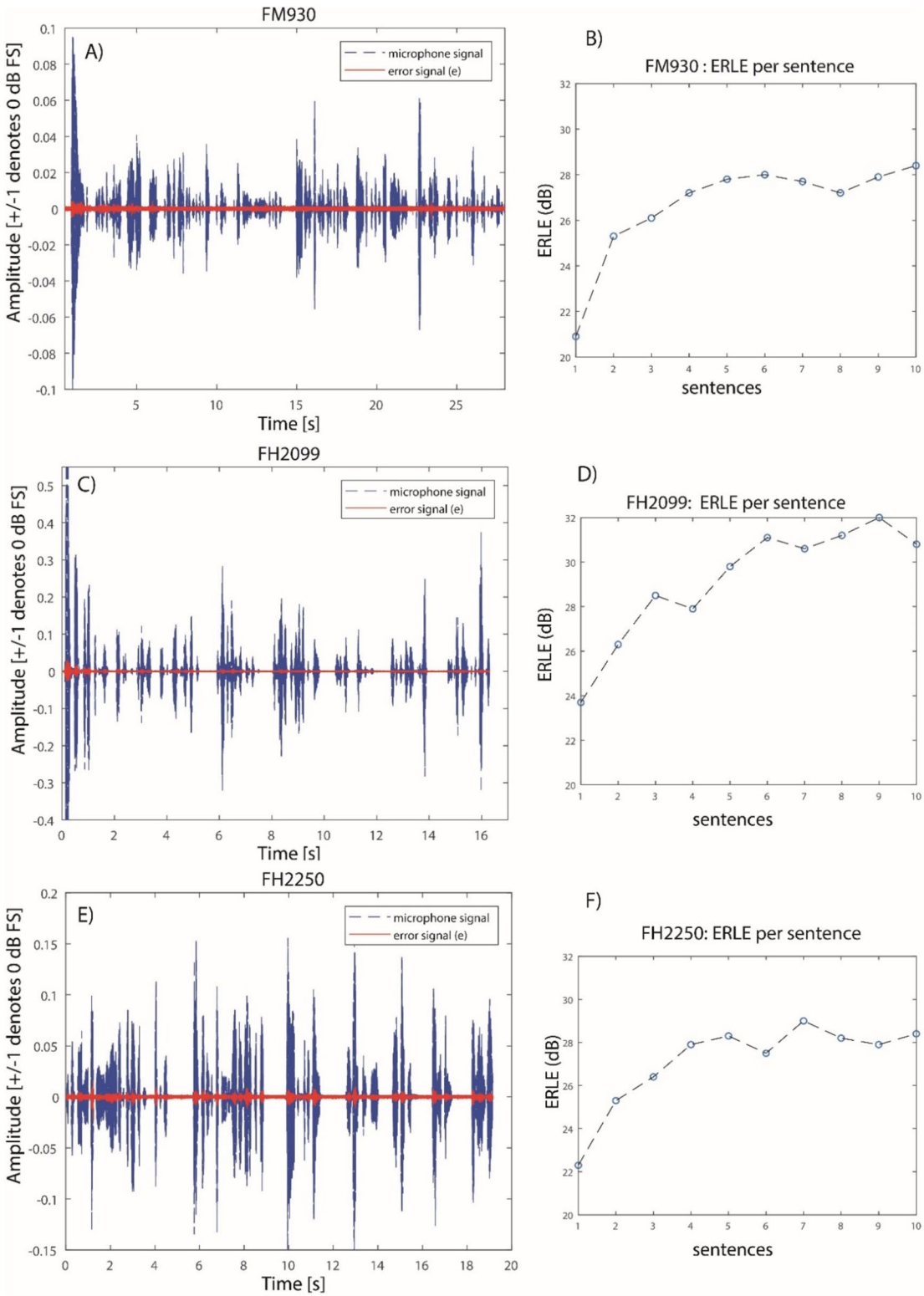

Fig. 5 **A**, **C**, **E** Captured microphone data ($d[n]$) versus the output of the AEC, error signal ($e[n]$) while the far end is reading ten HINT sentences for FM920, FH2099, and FH2250, respectively. **B**, **D**, **F** The ERLE achieved by the presented AEC solution in terms of ERLE per sentence for each truck. The engine was off in all these measurements

very short time. Thereafter, the ERLE stays over the dashed line and its steady state at 25 dB, as depicted in Fig. 4.

Figure 5A shows the microphone signal (*d[n]*) versus the output of the AEC (*e[n]*) for the FM930 truck while the engine was off illustrating a 25.43-dB ERLE. Figure 5B shows the ERLE values per sentence. Figure 5C–F illustrates similar data for FH2099 and FH2250 where the overall ERLE values are 27.26 and 30.9, respectively. Figure 5 indicates that in all cases the ERLE surpassed 25 dB somewhere before the onset of the second sentence.

Table 1 reports the ERLE assessed in the four truck models under test while the engines were off. The execution times for the MATLAB code to process 2 min (120 s) of data are also given. The WER during the DT period before and after the AEC are provided too. Table 1 also reports the execution time of the C code on the real-time ARM STM32F767 DSP. The execution time on the DSP has been reported as an average of the algorithmic latency per buffer. Each buffer is 50 ms long.

The IDE debugging tools provide a tool that can provide information on the time the CPU takes for processing each buffer. Table 1 reports the average of this value over all buffers in each audio file. It is worth noticing that the proposed algorithm consists of about 6.4 million basic operations per buffer. Since the STM32F-767CPU can process 460 million instructions per second (i.e., DMIPS = 460), it should take around 13.9 ms for the CPU to execute 6.4 million basic operations. However, as Table 1 shows, the average processing time per buffer is about 20 ms and 25 ms for recordings in FM and FH trucks, respectively. The difference between the theoretical estimation above (13.9 ms) and the measurement (around 20 and 25 ms) must be due to overheads, peripheral computations, and other background operations.

3.2 Echo attenuation in the presence of engine noise

Table 2 reports ERLE for the four truck models while the engine is on. The difference with Table 1 is that, due to the running engine, there is a remarkable amount of ambient noise (*r[n]*) that contaminates the microphone signal (*d[n]*). The existence of noise affects the calculation of ERLE as defined by Eq. (5) since that equation is based on the assumption that ambient noise is negligible. However, in the presence of noise, the microphone signal should be considered an addition of the speech component received at the microphone, and the ambient noise (i.e., $d[n] = s[n] + r[n]$) and the output error signal will likewise be an addition of the actual error signal from the speech (denoted by $e_s[n]$) which has been processed by AEC and the ambient noise that has been captured by the microphone (i.e., $e[n] = e_s[n] + r[n]$).

Since *r[n]* is uncorrelated with speech (*s[n]* and $e_s[n]$), Eq. (6.a) and (6.b) can be written as below. Consequently, the effect of the ambient noise could be excluded from the ERLE estimation using Eq. (6.c) below. Equation (6.c) is only valid if *r[n]* is uncorrelated with speech signals (*s[n]* and *x[n]*) which is a realistic assumption for vehicle engine noise. Equation (6.c) replaces Eq. (5) in order to compensate for ambient noise in our ERLE estimation, and it yields the corresponding values reported in Table 2. From another perspective, Eq. (6.c) presents the spectral subtraction of the ambient noise from the microphone signal and the error signal. Spectral subtraction is one of the most common de-noising methods in speech

Table 1 The ERLE achieved by the proposed echo cancelation solution in four truck models while the engine is off

Truck model	ERLE	WER during DT (before and after AEC)	Total code execution time (MATLAB)	Average processing time per buffer (DSP)
FM 928	25.54 dB	0.28 → 0.08	9.6 s	20.3 ms
FM 930	25.43 dB	0.46 → 0.1	9.7 s	20.4 ms
FH 2099	27.26 dB	0.4 → 0.08	10.7 s	25.5 ms
FH 2250	30.9 dB	0.5 → 0.1	10.1 s	25.2 ms

Table 2 The ERLE achieved by the proposed echo cancelation solution in four truck models while the engine is on

Truck model	ERLE	ERLE (after correction for noise)	WER during DT (before and after AEC)	Total Code execution time (MATLAB)	Average processing time per buffer (DSP)
FM 928	17.13 dB	27.84 dB	0.28 → 0.08	10.2 s	24.2 ms
FM 930	15.6 dB	25.5 dB	0.45 → 0.14	11.7 s	25.1 ms
FH 2099	23.07 dB	49.37 dB	0.5 → 0.12	12.2 s	25.7 ms
FH 2250	12.7 dB	45.3 dB	0.54 → 0.14	11.7 s	24.1 ms

signal enhancement [29]. However, it is used here in Eq. (6.c) for the purpose of ERLE estimation.

$$\sigma^2_{d[n]} = \sigma^2_{s[n]+r[n]} = \sigma^2_{s[n]} + \sigma^2_{r[n]} \Rightarrow \sigma^2_{s[n]} = \sigma^2_{d[n]} - \sigma^2_{r[n]} \tag{6.a}$$

$$\sigma^2_{e[n]} = \sigma^2_{e_s[n]+r[n]} = \sigma^2_{e_s[n]} + \sigma^2_{r[n]} \Rightarrow \sigma^2_{e_s[n]} = \sigma^2_{e[n]} - \sigma^2_{r[n]} \tag{6.b}$$

$$\mathrm{ERLE} = 10 \times \log_{10}^{\frac{\sigma^2_{s[n]}}{\sigma^2_{e_s[n]}}} = 10 \times \log_{10}^{\frac{\sigma^2_{d[n]} - \sigma^2_{r[n]}}{\sigma^2_{e[n]} - \sigma^2_{r[n]}}} \tag{6.c}$$

As Table 2 shows, the ERLE values rise substantially after correcting for the ambient noise. In the two first cases (FM 928 and FM 930), the corrected ERLE values lie in the same range as the ERLE values in Table 1. However, the corrected ERLE values for the two last cases (FH 2099 and FH 2250) in Table 2 are respectively 49.37 and 45.3 dB, which are much higher than previous cases. The reason for this anomaly is that the ambient noise levels are higher in these two truck models because they are heavy-duty models with much larger (noisier) engines. As a result, the term $\sigma^2_{r[n]}$ in Eq. (6.a, 6.b and 6.c) is large meanwhile the algorithm is successful in reducing the echo and thus yields a low value for $\sigma^2_{e[n]}$. Consequently, the term $\sigma^2_{e[n]} - \sigma^2_{r[n]}$ in the denominator of Eq. (6.c) becomes very small which results in high ERLE output.

Tables 1 and 2 show that the WER improves (reduces) substantially during the challenging DT periods as a result of the AEC processing. In Table 1, the WER has fallen from an average of 0.41 (SD = 0.09) down to 0.09 (SD = 0.01). In Table 2, the WER falls from an average of 0.44 (SD = 0.11) down to 0.12 (SD = 0.02). In both cases, the AEC has significantly ($p < 0.05$) reduced (improved) the WER which indicates the success of the AEC in making the near-end speech signal much more intelligible. The WER values in Table 2 are higher compared to Table 1 (both before and after applying the AEC) which is likely due to the presence of relatively higher engine noise in the latter case.

The presented results show that the AEC algorithm attenuates the echo by more than 25 dB in all cases and that the convergence time of the algorithm complies with ITU G.168 requirements [7], as seen in Fig. 4. From an implementation perspective, our results in Tables 1 and 2 show that each buffer of the data could be processed in a time significantly shorter than the buffer size itself (i.e., 50 ms) which indicates the feasibility of the proposed algorithm for being implemented on common DSPs.

3.3 Evaluating the choice of the Wiener filter length

Choosing an appropriate length for the Wiener filter ($\widehat{h}[n]$), L, has a great impact on the outcome of an AEC [2]. It also has a very substantial effect on the computational complexity of the AEC i.e., long Wiener filters might not be computationally feasible to be implemented on real-time DSPs with limited computational and memory resources. Here, based on the acoustical attributes of the truck cabins, we chose 800 coefficients for $\widehat{h}[n]$ corresponding to 50 ms at Fs = 16 kHz. To verify how a longer choice of L could have affected the result, we used 1360 coefficients for the Wiener filter, corresponding to 85 ms in time, and then ran the AEC algorithm over the 10 HINT sentences recorded in the FM928 truck.

Figure 6 shows the Wiener filter, $\widehat{h}[n]$, adapted to the echo path after 10 sentences have been read by the far end on the loudspeaker while the near-end party is silent. Figure 6 shows that the choice of the first 50 ms (corresponding to $L = 800$ coefficients for $\widehat{h}[n]$) is a good choice because the filter coefficients become sparse and negligible after 50 ms. Furthermore, Fig. 6 shows that the first 110 coefficients bear sparse values near zero, as well. These correspond to 6.9 ms (i.e., 110/Fs = 110/16,000 = 0.0069). This time corresponds to the "end-to-end delay" (or "flat delay") which is determined by the time it takes for the sound wave to travel from the loudspeaker to the microphone. Some real-world AEC algorithms benefit from prior knowledge about the fixed physical distance between the loudspeaker and the microphone by zero-padding the corresponding coefficients of the Wiener filter beforehand and thereby save computation (i.e., "delay-coefficient method" or "segmented method") [30]. However, we chose not to apply these methods to keep our solution as generic and robust as possible for different cabin sizes and models.

4 Discussion

We presented a linear NLMS-based echo canceller that together with a statistical DTD and audio activity detector can achieve high ERLE and fast convergence time consistent with ITU G.168 guidelines. The presented solution is a wideband time domain algorithm in the sense that it applies the algorithm on the input time series ($x[n]$ and $d[n]$) directly and does not distinguish the frequency content of the input signals before applying the algorithm. An alternative method is to decompose $x[n]$ and $d[n]$ into N frequency sub-bands, forming a filterbank, and run the adaptive NLMS algorithm in each sub-band independently. The outputs of all sub-band are then synthesized together. A substantial benefit of this method, known as frequency sub-band adaptive filtering (FSAF), is that it offers computational simplicity because the input signals can be down-sampled by a factor of Fs/N. Furthermore, it assures a faster and more successful convergence [2, 31]. A drawback of this method is that the "flat delay" must be already known when filtering $x[n]$. Moreover, the analysis/synthesis process in FSAF means that the spectral content of the final output is

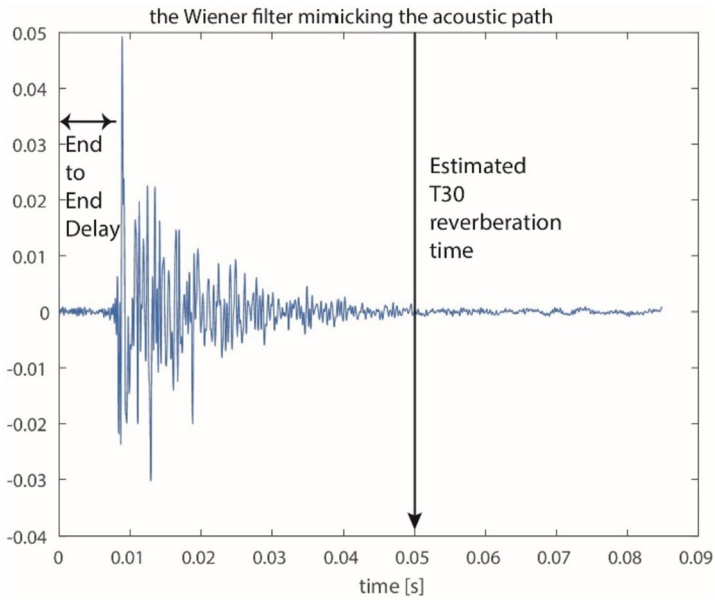

Fig. 6 The Winer filter ($\widehat{h}[n]$) coefficients after the AEC adapted to 10 HINT sentences read by the far end in an FM928 truck, engine off. The end-to-end delay corresponds to the physical distance between the microphone and the loudspeaker. The coefficients become sparse after 50 ms indicating that 50 ms is an optimal length for $\widehat{h}[n]$ in this AEC setup

manipulated. Here, we presented a time domain solution instead that is computationally efficient for being implemented on modern commercial DSPs. More importantly, the presented echo canceler is largely independent of the "flat delay" and thus is a more generic solution that could also be suitable for setups where the distance between loudspeaker and microphone is different from these specific truck models.

The length of the impulse response (L) has a great impact on the accuracy and also the computation feasibility of an AEC. Our results show that T30 reverberation time is a good indicator for choosing L. The T30 metric is very dependent on the size of the cabin and its furniture. The size and the furnishing in truck cabins can hugely vary across manufacturers and from one model to another which is most likely the main source of challenge in designing generic AEC systems for large vehicle cabins. We tested our solution in FM (mid) and FH (heavy) truck cabins and our choice of L proved appropriate for both series. However, it is more optimal to allow the manufacturers to fine-tune this important parameter based on the T30 associated with each cabin. Our codes (both MATLAB and C) are written in a manner to allow the parametric adjustment of L, thereby allowing the software architect to optimize and adjust this parameter.

Furthermore, besides L, we identified three other key parameters (namely: β, η_0, and E_0) that have a decisive impact on the general performance of our AEC system. We described in detail how these parameters are explicitly optimized according to the cabin geometry, typical noise levels, and typical microphone and loudspeaker hardware characteristics. The suggested values turned out to be functional for all four truck models used in this study, which suggests that these values must be valid for a large variety of other truck models in the market. However, we recommend that these parameters be exposed to the software architect to be able to potentially tune them for each cabin model, if necessary. This recommendation is more advisable if the proposed AEC solution is supposed to be implemented in small personal vehicles with quite different cabin acoustics.

The results showed that the presented echo canceller achieves a high level of ERLE values. However, these ERLE values might still not be sufficient if the gain of the loudspeaker is increased by the driver to very high volumes. In such cases, although the AEC attenuates the echo to a large degree since the initial echo intensity is very high, the far end might still be able to hear a noticeable amount of echo.

To address such issues and to attenuate the residual echoes beyond what a linear AEC can deliver, different post-processing methods based on nonlinear AES have been developed [2]. According to these methods, a gain is applied to the microphone signal when the near-end party is talking and the gain is reduced whenever the far-end party is talking and thereby reducing the effect of echo. The transition between the increase and the decrease of the gain function in AES is usually

facilitated by a ramping attack/release function to ease the abrupt changes and suppress the undesired spectral splatter [2].

Adding an AES module as a post-processor to the AEC can produce further attenuation of the residual echo and thereby improve the audio quality during a phone call. However, since AES is a nonlinear process [2], it cannot be applied to audio streams that are transmitted to ASR neural networks for voice assistance [8, 18]. In the Android operative system, the audio designer is capable of defining two separate streams for telephony applications versus speech recognition applications [32]. Android hardware abstraction layer (HAL) exposes three different audio streams: (1) *Voice_communication*, which is used by telephony applications; (2) *Voice_recognition*, which is used by voice assistance and speech dialog applications for speech recognition; and (3) *UNPROCESSED*, which enables applications to access the raw data captured by the microphone [32]. An optimal solution is that both AEC and AES are applied to the *Voice_communication* stream whereas only the AEC is applied to the *voice_reconition* stream. Accordingly, the telephony application can benefit from both AEC and AES while the speech recognition application fulfills its linearity requirements by only applying the fully-linear AEC algorithm.

5 Conclusion and future works

We presented a wideband time-domain NLMS-based adaptive AEC for large commercial vehicles and evaluated its performance in four different truck models produced by the Volvo group using standard HINT in the presence and absence of engine noise. The results showed that the ERLE and the convergence time achieved by this adaptive algorithm fulfill and even surpass the ITU G.168 requirements. The paper presents a fully linear AEC that could be used as a front end in telephony applications but also for keyword spotting modules in speech recognition systems and voice assistants. Furthermore, a real-time implementation of the algorithm on a Cortex-M7 DSP was presented which showed the computational feasibility of the proposed solution for implementation on automotive embedded systems with limited computational and memory resources.

The AEC solution presented here is a one-microphone, one-speaker solution that suits the current typical truck production setup where speech-related signals (telephone calls) are played on a specific loudspeaker. However, if the AEC is supposed to also cancel echo from music being played on multiple loudspeakers, the presented algorithm must be expanded to multiple loudspeakers, e.g., [33]. Furthermore, dual microphones are becoming popular in the vehicle industry which implies that future AEC solutions should be expanded to include multi-microphone multi-speaker situations, which is a more general form of AEC [2, 33].

Abbreviations

ADC	Analog to digital conversion
AEC	Acoustic echo cancelation
AES	Acoustic echo suppression
CPU	Central processing unit
DAC	Digital to analog conversion
DMA	Direct memory access
DNN	Deep neural network
DT	Double talk
DTD	Double-talk detection
DSP	Digital signal processor
ERLE	Echo reduction loss enhancement
FIR	Finite impulse response
HINT	Hearing in noise test
ITU	International telecommunication union
NCC	Normalized cross-correlation
NLMS	Normalized least mean square
PTT	Push to talk
SNR	Signal-to-noise ratio
VAD	Voice activity detection
VGTT	Volvo Group Truck Technology

Acknowledgements

Preliminary parts of this work were done under the supervision of AS and were successfully presented by BR as his M.Sc. thesis [34] at the Department of Statistics and Machine Learning, Linköping University, Sweden. The thesis presented a basic proof-of-concept implementation of the proposed AEC solution without including a detailed optimization of the parameters and the real-time DSP implementation and ITU, and WER analyses. These items were performed later and are comprehensively explained in this manuscript. The authors would like to thank Ibáñez Gabriel at VGTT for helping setting up the project and Josef Wilzén from Linköping University for his helpful comments. The first author presented Figs. 1B and 3 in his book chapter published by IntechOpen under open source license [35]. IntechOpen has agreed to the re-publications of these figures in this paper.

Authors' contributions

AS designed the study, planned and carried out the data collection, supervised the code development, performed the various evaluation analyses, and wrote the manuscript. BR implemented the first prototype of the algorithm in MATLAB under AS's supervision and reported the preliminary results in his thesis. GG helped with the real-time DSP implementation of the algorithm in *C* language and with organizing the references. ZG oversaw the text and provided scientific comments.

Funding

Open access funding provided by Umea University. This research was initially defined as an M.Sc. thesis work and was partly funded by the innovation office at the vehicle connectivity department at Volvo Group Truck Technology (VGTT), Göteborg, Sweden.

Availability of data and materials

The corresponding MATLAB source code is publicly available online on the project's GitHub page [36] together with exemplary audio files that convey near-end signals recorded inside the trucks before and after applying the presented AEC solution.

Declarations

Competing interests

The authors declare no competing interests.

Received: 27 February 2023 Accepted: 9 September 2023
Published online: 07 October 2023

References

1. M.M. Sondhi, A.J. Presti, A self-adapting echo canceller. Bell syst. Tech. J. **45**(10), 1851–1854 (1966)
2. W. Kellermann, Echo cancellation, in *Handbook of signal processing in acoustics*, vol. 1, ed. by D. Havelock, S. Kuwano, M. Vorländer (Springer, New York, 2008), pp.883–895
3. C. Paleologu, S. Ciochin, J. Benesty, S.L. Grant, An overview on optimized NLMS algorithms for acoustic echo cancellation. EURASIP J. Adv. Signal Process. (2015). https://doi.org/10.1186/s13634-015-0283-1
4. G. Enzner, H. Buchner, A. Favrot, F. Keuch, *Acoustic echo control, in Academic press library in signal processing*, vol. 4, ed. by J Trussell, A Srivastava, A.K Roy-Chowdhury, A Srivastava, et al. (Elsevier, Amsterdam, 2014), pp. 807–877
5. G. Enzner, P. Vary, Frequency-domain adaptive Kalman filter for acoustic echo control in hands-free telephones. Signal Process. **86**(6), 1140–1156 (2006)
6. M. Souden, J. Wung, B.H.F. Juang, 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, in *A probabistic approach to acoustic echo clustering and suppression*. (IEEE, New Paltz, 2013), pp.1–4
7. H. Zhang, D. Wang, *2018 Proc. Interspeech. Deep learning for acoustic echo cancellation in noisy and double-talk scenarios* (Interspeech, Hyderabad, 2018), pp.3239--3243
8. O. Schwartz, A. Schwartz, *IEEE ICASSP conference. RNN-based step-size estimation for the RLS algorithm with application to acoustic echo cancellation* (IEEE, Rhodes, 2023), pp.1–5
9. D. Yang, F. Jiang, W. Wu, X. Fang, M. Cao, *2023. IEEE ICASSP conference. Low-complexity acoustic echo cancelation with neural Kalman filtering* (IEEE, Rhodes, 2023), pp.1–5
10. International telecommunication union, G.168: 04/2015 Digital network echo canceller, https://www.itu.int/rec/T-REC-G.168-201504-I/en. Accessed 13 Feb 2023.
11. T. von Neumann, C. Boeddeker, K. Kinoshita, M. Delcroix, R. Haeb-Umbach, *IEEE ICASSP conference. On word error definitions and their efficient computation for multi-speaker speech recognition systems* ((IEEE, Rhodes, 2023), pp.1–5
12. S. Oh, V. Viswanathau, P. Papamichalis, *IEEE ICASSP conference. Hands-free voice codcation in an automobile with a microphone array* (IEEE, San Francisco, 1992), pp.281–284
13. F. Chen, I.M. Jonsson, J. Villing, S. Larsson, Application of speech technology in vehicles, in *speech technology: theory and applications*. ed. by F. Chen, K. Jokinen (Springer, New York, 2010), pp.195–219
14. Discussion forums for all front-wheel drive and all-wheel drive Volvo models, Bluetooth calls echoeing, https://www.swedespeed.com/threads/bluetooth-calls-echoing.639118/. Accessed 15 Feb 2023.
15. P. Heisterkamp, *Proc. HLT. Linguatronic- product-level speech system for Mercedes-Benz cars* (HLT, San Diego, 2001)
16. Volvo Cars connectivity, Volvo Cars brings infotainment system with Google built in to more models, https://www.media.volvocars.com/global/en-gb/media/pressreleases/279230/volvo-cars-brings-infotainment-system-with-google-built-in-to-more-models. Accessed 13 Feb 2023.
17. Volvo trucks Global, Volvo trucks to deliver Amazon Alexa in new heavy-duty trucks, https://www.volvotrucks.com/en-en/news-stories/press-releases/2020/dec/volvo-trucks-first-to-deliver-amazon-alexa-in-new-heavy-duty-trucks.html. Accessed 13 Feb 2023.
18. Google Android team, 5.4.2 Capture for voice recognition, in Android compatibility definition document, https://source.android.com/compatibility/10/android-10-cdd. Accessed 13 Feb 2023.
19. E. Hänsler, G. Schmidt, *Acoustic echo and noise control: a practical approach* (Wiley, Hoboken, NJ, USA, 2004)
20. Y.H. Chen, S. Ruan, T. Qi. *IEEE conference on signal processning, communication, and computing (ICSPCC)*. An automotive application of real-time adaptive Wiener filter for non-stationary noise cancellation in a car environment (IEEE, Hong Kong, 2012), pp. 597–601.
21. M.S. Hussain, M.A. Hasan, M.F. Bari, and A.B.M. Harun-Ur-Rashid *4th International Conference on Advances in Electrical Engineering (ICAEE)*. A fast double-talk detection algorithm based on signal envelopes for implementation of acoustic echo cancellation in embedded systems (ICAEE, Dhaka, 2017), pp. 199–204
22. D. Duttweiler, A twelve-channel digital echo canceler. IEEE Trans Commun **26**(5), 647–653 (1978)
23. J. Benesty, D.R. Morgan, J.H. Cho, A new class of doubletalk detectors based on cross-correlation. IEEE Trans Speech Audio Proc **8**(2), 168–172 (2000)
24. A. Field, *Correlation, in Discovering statistics using IBM SPSS statistics*, 4th edn. (SAGE publications Ltd, London, 2013), pp.262–293
25. D.C. Swanson, *Acoustic data acquisition, in Handbook of signal processing in acoustics, vol. 1* (Springer, New York City, USA, 2008), pp. 17–32
26. M. Hällgren, B. Larsby, S. Arlinger, A Swedish version of the Hearing In Noise Test (HINT) for measurement of speech recognition. Int. J. Audiolog. **45**, 227–237 (2006)
27. STMicroelectronics, STM32F7 Series, https://www.st.com/en/microcontrollers-microprocessors/stm32f7-series.html. Accessed 13 Feb 2023
28. A. Osborne, *Basic concepts, in An introduction to microcomputers, vol. 1*, 2nd edn. (McGraw Hill, London, 1983), pp. 5–93
29. N. Upadhyay, A. Karmakar, Speech enhancement using spectral subtraction-type algorithms: a comparison and simulation study. Procedia Comput Sci **54**, 574–584 (2015)
30. Y. Lu, R. Fowler, W. Tian, L. Thompson, Enhancing acoustic echo cancellation via estimation of delay. IEEE trans Signal Proc **53**(11), 4159–4168 (2005)
31. Z. Shen, Y. Yu, T. Huang, Normalized subband adaptive filter algorithm with combined step size for acoustic echo cancellation. Circuits Syst Signal Process **36**, 2991–3003 (2016)
32. Google Android team, Online documentation: audio architecture in Android, https://source.android.com/devices/audio. Accessed 13 Feb 2023.
33. L. Romoli, S. Cecchi, F. Piazza, Multichannel acoustic echo cancellation exploiting effective fundamental frequency estimation. Speech Com **86**, 97–106 (2017)
34. B. Ramkumar, Master's thesis, Linköping University, 2020, Acoustic echo cancellation inside a truck cabin, http://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A1437199&dswid=-6580
35. A. Saremi, *Spatial audio signal processing for speech telecommunication inside vehicles, in Advances in fundamental and applied research on spatial audio*, ed. by BFG Katz, and P Majdak (Intechopen, UK., 2022), pp. 175–192
36. A. Saremi, Github page (London, 2023), https://github.com/AminSaremi/AcousticEchoCancelation. Accessed 13 Feb 2023

Publisher's Note