# CHALLENGES AND SOLUTIONS FOR DESIGNING SOFTWARE AEC ON PERSONAL COMPUTERS

*Qin Li, Chao He and Wei-Ge Chen*

{qinli, chaohe, wchen}@microsoft.com
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

## ABSTRACT

In this paper, we present some unique challenges of designing a robust and practical acoustic echo canceller (AEC) for personal computers (PC) and propose effective algorithms that meet these challenges. Specifically, the quality and robustness of our AEC is enhanced by selectively applying a glitch recovery process based on a novel feature that measures the quality of the alignment between the microphone and loudspeaker signals. In addition, a multi-step clock drifting compensation method is applied to improve the quality of AEC in case of clock drifting. The effectiveness of the algorithms is demonstrated by a real-time AEC component running in a variety of operating environments.

**Index Terms**— acoustic echo cancellation/canceller, software, clocking drifting, AEC, PC, VOIP

## 1. INTRODUCTION

Acoustic Echo Cancellation (AEC) is a digital signal processing technology which aims to remove the acoustic echo from a speaker phone in two-way or multi-way communication systems [1]. During the recent decade, the explosive growth of voice over IP (VOIP) applications calls for robust and reliable AEC running in software on a variety of PC operating systems. Although there has been extensive research on AEC and significant advances have been made, to date little effort has been paid to the unique software design challenges encountered on the PC platforms except for a few exceptions [2]-[4] where signal synchronization were briefly discussed.

Figure 1 illustrates an example of one end of a typical VOIP application on a PC, which includes an audio capture path and an audio render path in two directions. In the capture path, an analog to digital (A/D) converter converts the analog audio signal captured by microphone to digital samples continuously at a sampling rate $F_{sm}$. The digital audio samples are saved in a capture buffer sample by sample and are retrieved in frame increments (denoted as $m[i]$ ). Finally, samples in $m[i]$ are processed and transmitted. In the render path, a similar process occurs involving discrete time signal $s[i]$, the continuous time signal $s[t]$, at a sampling rate $F_{ss}$.

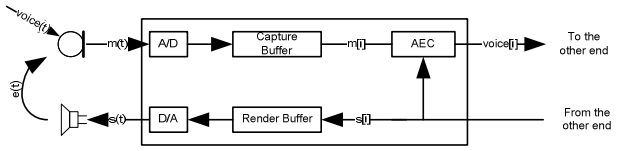


**Figure 1.** Hands-free VOIP configuration.

The capture and render buffers are necessary for practical purposes but they do introduces delay. For example, a sample generated by the A/D converter will stay in capture buffer for a short while before it is read out.

Typically, AEC assumes the room can be modeled as a finite duration linear plant. The echo $e(t)$ is represented per the following relationship

$$e(t) = s(t) * h(t) = \int_0^{T_e} h(\tau) \cdot s(t-\tau)\, d\tau \qquad (1)$$

where * denotes convolution, $h(t)$ the room response, $T_e$ the length of the room response filter, and $s(t)$ the loudspeaker signal. We assume our AEC uses a typical architecture, where adaptive filters, such as Normalized Least Mean Square (NLMS) adaptive filters [1], operate in the discrete Fourier transform (DFT) subbands to adaptively model the time-varying room response (see, e.g. [1]).

In the literature it is commonly assumed that the discrete-time version of Equation (1) holds. However, due to the imperfection of the PC platform, e.g. the imprecision of the A/D and D/A and the limitations of the system software components, $m[n]$ and $s[n]$ are far from ideally digitized versions of $m(t)$ and $s(t)$. If one does not take these factors into consideration, serious breakdown of AEC can happen, resulting in poor quality. In this paper, we consider these problems explicitly and demonstrate algorithms that can effectively alleviate these system "artifacts."

## 2. SIGNAL ALIGNMENT ISSUES

Figure 2 shows the prediction timeline for both continuous and discrete time signals with the underlying assumption that $s[t]$ and $m[t]$ are sampled at a consistent sampling rate and synchronized. To fully illustrate the point of consistency, we introduce the concept of the "relative

sample offset" (RSO, $d[i]$). Conceptually, the RSO can be understood as follows:

1. Given a discrete-time microphone sample $m[i]$, suppose we can find the physical time $\tau$ when $m[i]$ was generated by the A/D converter (Figure 1).
2. According to Equation (1) and Figure 2, the echo at $m(\tau)$ is a function of the loudspeaker signal $s(t)$ during the time interval $t = [\tau - T_e, \tau]$ preceding time $\tau$.
3. Next, let's assume that we find the index $j$ of loudspeaker signal such that $s[j]$ is played back at time $\tau$ at the loudspeaker.

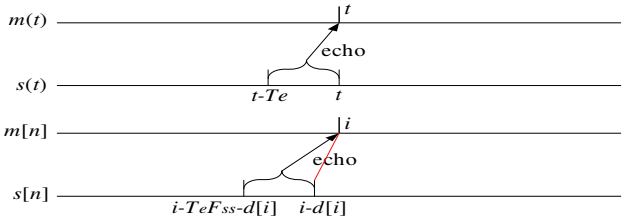


**Figure 2.** Time-line of prediction.

In other words, we identify a loudspeaker sample $s[j]$ that is rendered at the same time as the microphone sample $m[i]$ is captured. We define the RSO as the difference of the indices, or $d[i]=i-j$. When the precise sampling rates $F_{sm}$ and $F_{ss}$ are known, $d[i]$ should be an exact linear function of $i$, or $j$:

$$d[i] = i(F_{sm} - F_{ss}) / F_{sm} + C \qquad (2)$$

where C is an arbitrary constant depending on the starting point of the two sampling clocks. Under the ideal condition, we can always make $F_{sm}$ and $F_{ss}$ identical through re-sampling, in which case $d[i]$ reduces to a constant and $m[i]$ and $s[i]$ are said to be in "alignment". In reality, however, such is rarely the case due to at least one of the following manifestations that we have encountered during our investigations:

(1) Clock drifting: The microphone signal $m[i]$ and loudspeaker signal $s[i]$ generally are sampled by two different sampling clocks. The actual clock frequencies are usually slightly different and *unknown* even though their nominal frequencies are known to be the same. If the nominal frequency is used as is, $m[i]$ and $s[i]$ will eventually lose "alignment" over time. This phenomenon is commonly referred to as clock drifting.

(2) Time-varying delay: As mentioned earlier, buffering introduces a delay in both capturing and playback paths. The delay is unfortunately not constant over time.

(3) Noisy timing measurements: Modern audio hardware provides timing data in order to synchronize $m[i]$ and $s[i]$. The information is always noisy, due to limited numerical precision, data transfer delay, multi-threading, etc.

(4) Missing samples/Glitch: The system may unpredictably lose some samples, typically referred to as a "glitch". After a glitch, $m[i]$ and $s[i]$ will no longer be in alignment, even if they have been prior to that event.

Considering these factors, we propose a modification of Equation (2) as a model for the RSO:

$$d[i] = Ri + e[i] + w[i] + C \qquad (3)$$

where $e[i]$ is zero-mean WGN with variance $\sigma^2$ to model the noisy timestamps, $w[i]$ is a sparse (i.e. $w[i]$ = constant in most places) and random step function representing the discontinuities caused by the glitches, and $R$, typically a very small value, is the clock drifting rate. Note that the RSO can be any rational number, instead of integers only.

It is clear to us now that unless compensated, the variability of RSO will break the basic linear prediction model that AEC relies on and often cause AEC to fail completely. Next, we will be introducing techniques that compensate the non-ideal RSO, seek to bring $m[i]$ and $s[i]$ into alignment and consequently improve the quality and the robustness of AEC substantially.

## 3. GLITCH DETECTION AND RECOVERTY

To handle glitches, our objectives are to detect where the glitches happen, bring signals into alignment and take necessary steps to recover quickly. In order to achieve these goals, we first need to estimate the parameters that characterize the RSO, i.e., clock drifting rate $R$ and the variance of the observation noise $\sigma^2$.

### 3.1 Parameter estimation

Generally speaking, estimating the parameters of Equation (3) jointly is a very hard problem. However, due to the sparseness of $w[i]$, we propose a simplified, suboptimal solution that takes place in two steps. In the first step, we assume the local observation window doesn't contain any glitches. With $w[i]$ out of the picture and given $N$ number of frames, we can estimate $R$ and $\sigma^2$ by means of the formal equations:

$$\hat{R} = \frac{(S_{id} - S_i S_d / N)}{(S_{dd} - S_d S_d / N)},$$

$$\hat{C} = S_d / N - \hat{R} S_i / N,$$

$$\hat{\sigma}^2 = (S_{dd} - 2\hat{R} S_i S_d + \hat{R}^2 S_{ii} - 2\hat{C} S_d + 2\hat{C}\hat{R} S_i) / N + \hat{C}^2$$

where $S_i = \sum_{i=1}^{N} i$, $S_d = \sum_{i=1}^{N} d[i]$, $S_{ii} = \sum_{i=1}^{N} i^2$, $S_{dd} = \sum_{i=1}^{N} d[i]^2$, and $S_{id} = \sum_{i=1}^{N} i \cdot d[i]$. As these estimates are optimal in terms of the mean squared error, they should approach their true values, as $N$ increases. However, in practice, it is not a good idea to let $N$ increase unbounded. The clock drifting rate $R$ may be slowly varying over time and observation noise may not be strictly stationary. Thus, we define an estimation window of several

hundred seconds long and for simplicity reset the estimates at the end of the window. Practical experience shows that $\hat{R}$ and $\hat{\sigma}^2$ usually converges in several seconds. Furthermore, standard recursive least square formulations can be chosen for online estimation of these parameters, in which case the explicit windowing is no longer needed.

### 3.2 Glitch Detection

Based on our observations, glitches can be classified into two categories by the amount of time discontinuity the glitch introduces. To this end, we choose a threshold several times of $\hat{\sigma}$, the standard deviation of the RSO noise. Any discontinuity larger than the threshold is detected as a "large" glitch. Note that once a large glitch is detected, the adjacent RSO data will be discarded from parameter estimation. Although this doesn't change the fact that the steps described in Section 3.1 is suboptimal, our experiments show the remaining small glitches don't cause any serious problem.

Small glitches present a challenge as the glitch size is indistinguishable from the range of RSO noise. We propose to apply a Move-Average (MA) filter to the RSO data $d[i]$:

$$d_{MA}[i] = \frac{1}{L}\sum_{l=0}^{L-1} d[i-l]$$

where $L$ denotes the window size and is made proportional to $\hat{\sigma}^2$ in order to adapt to different noise conditions. If $d_{MA}[i]$ has a change larger than a predefined threshold within a certain time period, a small glitch is identified.

To further enhance the glitch detection process, we make the detection adaptive to $\hat{\sigma}^2$ in three zones. First, when $\hat{\sigma}^2$ is low, our AEC applies both large and small glitch detection. Secondly, when $\hat{\sigma}^2$ is within a medium range, we cease the process of small glitch detection, while continuing the process for large glitch detection. Finally, when $\hat{\sigma}^2$ is too high, our AEC ceases to perform both large and small glitch detection processes.

### 3.3 Fast Glitch Recovery

After a glitch is detected, the AEC needs to re-align the two streams $m[i]$ and $s[i]$. The exact mechanism of re-alignment will be covered in Section 4. In addition, our AEC stops updating the adaptive filter coefficients for all samples involved in the re-alignment until all samples involved are passed. We found this approach vastly improve AEC quality compared to the other two obvious choices: If the adaptive filter coefficients are reset, the AEC would take time to converge during which echo would likely be heard. On the other hand, if the coefficients are updated continuously, the adaptive filters could lose convergence due to the mis-aligned signals.

## 4. CLOCK DRIFTING COMPENSATION

As mentioned earlier, when there is either clock drifting (CD) or a glitch, we need to compensate the time discontinuity and bring the microphone and loudspeaker signals into alignment again. Here, we propose a novel method for effective CD compensation.

### 4.1 Frequency Domain Multi-Step CD Compensation

In common practice, the AEC adjusts one of the stream buffers by one sample when the accumulative clock drifting is greater than one sample. The nature of adjustment dictates that the upper limit of the alignment accuracy is one sample. Suppose *x[n]* is one of the streams and *X[k]* is its DFT. In case the stream is adjusted by one sample delay, i.e., *x'*[*n*]=*x*[*n-1*], the spectrum of the adjusted signal is given by $X'[k] = X[k]e^{-j2\pi k/K}$ where $K$ is the DFT size, and $e^{-j2\pi k/K}$ is the phase change due to the one-sample delay. The phase variability is sudden and significant at high frequencies, causing filter divergence and consequently noticeable quality degradation.

We propose that the CD compensation should be applied similarly to how CD occurs, i.e., the compensation should be applied gradually and continuously with time. The phase changes should be small (fractional sample) and graduate enough so that the subband adaptive filters are able to catch up easily without quality degradation. We called this method *multi-step CD compensation,* in contrast to *uni-step CD compensation* where the minimal adjustment is always one sample.

However, instead of performing fractional sample delay in time which can be computationally expensive, we take advantage of the frequency domain that the AEC operates in. Specifically, for the case of $P$ sample delay, we spread $P$ evenly across $M$ consecutive frames by performing: $X''[k] = X[k] \cdot e^{-j2\pi kS/K}$ which approximates the fractional sample delay in time domain $x''[n] = x[n-S]$ where $S = P/M$ is referred to as the *step size*. Due to the circular shift property of the DFT, the approximation is only valid when $P << K$. When $P$ accumulates to a certain level, we reduce the burden on the frequency domain method by offloading a convenient portion of $P$ to an equivalent time domain operation so that $P$ remains small. Although it is rare, when the CD rate is larger or close to 1 sample per frame, this offloading process will happen frequently enough to cause a stability problem. In this case, we instead use a time domain re-sampler to handle the higher CD more effectively.

### 4.2 Adaptive Step Size Determination

Ideally, to precisely compensate for the CD, the step size should match the CD rate, i.e. $S = RF_sI$, where $R$ is the CD

rate, $F_s$ the sampling rate and $I$ the frame size. However, in practice since $R$ and $F_s$ are unknown and may be time varying, we use the following mechanism to determine the step size adaptively.

We let $S_j = \rho P_j$ at the $j^{\text{th}}$ frame where $\rho$ is a constant set empirically to ensure stability and quick convergence, and $P_j$, a rational number, is accumulated based on $\hat{R}$. At each data frame where $|P_j| > \rho$, a phase compensation of $e^{-j2\pi k\phi_j / K}$ will be applied, where $\phi_j = \sum_{l=0}^{j} S_l$. As time goes on, when $\phi_j$ crosses a predetermined threshold $Q$, we perform an equivalent $Q$ sample shift in time. $P_j$ and $\phi_j$, then, will be updated as follows: $P_j = P_j - Q$ and $\phi_j = \phi_j - Q$.

## 5. RESULTS

We have integrated the algorithms presented above to our software AEC system and obtained the following positive results. Our software AEC runs robustly on various operating systems under diverse operating conditions. The average computation load is about 50 MIPS.

Figure 3 shows an example of small and large glitches in the RSO. The raw RSO is shown as the blue trace, while the output of the MA filter is shown in red. For convenience, the sample numbers and frame number are converted into physical time. In this example, the estimated clocking drifting rate $\hat{R}$ is 0.0002, and the estimated variance of RSO $\hat{\sigma}^2$ is 0.083 (ms$^2$). At 8 second, there is a big glitch with size of 5.3 ms, which is detected immediately. At 12 second, there is a small glitch of 0.5 ms, which is about the same amount as the maximum of RSO error. Despite the challenge, the small glitch is successfully identified after about 1 second.

An example of the multi-step CR compensation is shown in Figure 4. The original data shown in the figure is sampled at 16 kHz with a CR rate of 1.7x10$^{-4}$, which needs one-sample adjustment for every 0.37 seconds. The blue line shows the AEC output using the uni-step compensation method where there are three adjustments at 3.08, 3.45, and 3.82 second, respectively. The red line shows the AEC output using the multi-step CR compensation. The multi-step method has clearly better quality and yields much smoother output with very low echo level, while with the uni-step method the residue echo level rises after the adjustment and the degradation (echo leak) lasts about 50-100 ms. In terms of Echo Return Loss Enhancement, we observe about 6-7 dB local improvement, and about 2 dB average improvement. In this example, the step size was adapted automatically between 0.02 and 0.03. The long term average of the step size matches the actual CD rate when the sampling rate $F_s$ and frame size $I$ are factored in.

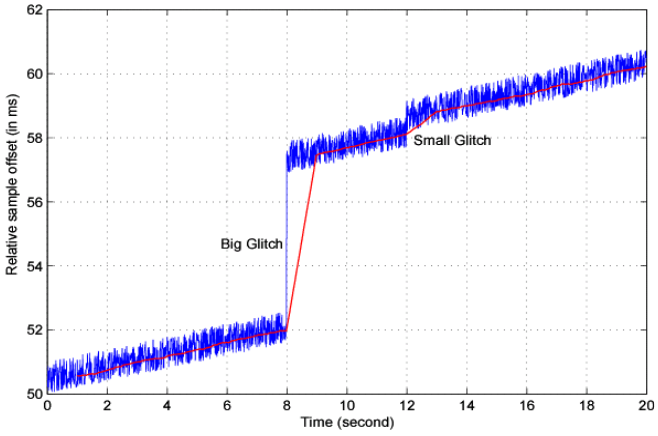


**Figure 3.** Glitch Detection

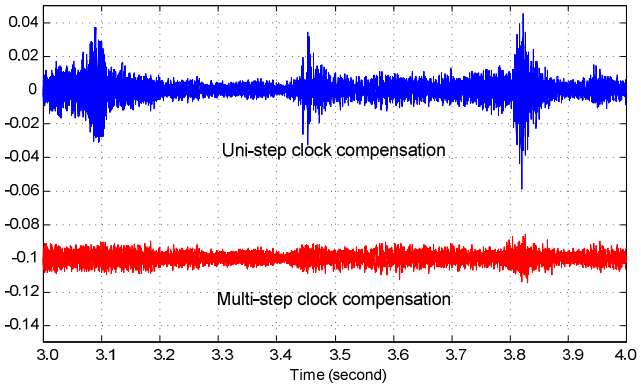


**Figure 4.** Uni-step vs. multi-step compensation.

## 6. CONCLUSIONS

In this paper, we have considered several unique challenges that adversely affect the quality of software AEC running on a PC. Effective solutions are proposed and demonstrated. Our AEC implementation, while significantly improved, is still far from perfect. Future work and further improvements are definitely recommended.

## 7. REFERENCES


[1] S. L. Gay and J. Benesty, *Acoustic Signal Processing for Telecommunication*, Kluwer Academic Publishers, 2001.

[2] V. Fischer, T. Gänsler, E.J. Diethorn, J. Benesty, "A software stereo acoustic echo canceller for Microsoft Windows", in *Proc. IWAENC, 2001*, pp. 87-90.

[3] T. Gänsler, V. Fischer, E. J. Diethorn, and Jacob Benesty, "The WinEC: A Real-Time Hands-Free Stereo Communication System," in *Audio Signal Processing for Next-Generation Multimedia Communication Systems*, pp. 171-193, Springer, 2004.

[4] J. W. Stokes and H. S. Malvar, "Acoustic Echo Cancellation with Arbitrary Playback Sampling Rate", in *Proc ICASSP, 2004*, pp. 153 -156.