



# Efficient Three-stage Pitch Estimation for Packet Loss Concealment

Xuejing Sun and Sameer Gadre

Cambridge Silicon Radio  
 Auburn Hills, MI 48326 U.S.A  
 {xuejing.sun, sameer.gadre}@csr.com

## Abstract

This paper presents a low-complexity pitch estimation algorithm for packet loss concealment. The algorithm divides the pitch estimation into three stages with each additional stage providing further accuracy. Compared with a system based on G.711 Appendix I, the proposed algorithm requires approximately 32 percent fewer cycles on a DSP processor integrated in a Bluetooth chip. Furthermore, objective evaluation of the voice quality using PESQ showed the algorithm yields substantially higher scores.

**Index Terms:** packet loss concealment, pitch, low complexity, Bluetooth

## 1. Introduction

Packet loss occurs frequently in wireless or VoIP communications under adverse connection conditions. Lost packets result in clicks and pops or other audible artifacts that greatly degrade the perceived speech quality and intelligibility for the listener. It can render speech totally unrecognizable at high loss rates.

Packet Loss Concealment (PLC), also known as frame erasure concealment (FEC), generates a synthetic speech signal to cover missing data (erasures) in a received bit stream. Many PLC methods have been proposed in the past [1], among which pitch based waveform substitution is highly popular due to its relative simplicity and good performance [2]. Using the quasi-periodic property of speech signals, the basic idea of such systems is to fill the gap using a waveform segment that is one or integer multiple pitch periods apart from the gap. The accuracy of pitch detection naturally has a crucial impact on the quality of PLC systems. However, many pitch estimation algorithms are too computationally prohibitive for PLC applications even though they show good pitch detection performance. Instead, time domain correlation based metrics are often employed due to the good trade-off between complexity and performance. Commonly used methods include Normalized Cross-Correlation (NCC) [2], Sum of Squared Difference (SSD) [3], and Average Magnitude Difference Function (AMDF) [4]. For embedded devices a direct calculation of these functions is still too computationally expensive and various methods have been proposed to reduce their processing load. One frequently adopted approach is to perform pitch estimation in two phases [2][5][6]. G.711 Appendix I [2] is a widely used PLC algorithm. In this method, a coarse search is first performed on a 2:1 decimated signal, and then a finer search is performed in the vicinity of the peak of the coarse search. A decimated bisectonal pitch refinement procedure is proposed in [5] to further reduce the complexity of the second phase. In [6], a two-phase system is proposed in which a pattern matching method is described to refine pitch accuracy in the second phase. Although these approaches reduce the number of

calculations the algorithms require, the computational complexity associated with estimating the pitch period remains a problem. This is a particularly pertinent problem for battery powered devices using Bluetooth. To further reduce pitch detection complexity, this paper proposes a highly efficient three-stage pitch determination algorithm suitable for packet loss concealment. In the first stage, pitch is calculated within a very narrow range. Then the multiples of the estimated pitch are checked in the extended range in the second stage. In the third stage a pitch refinement is performed to minimize the mismatch during cross-fading at the boundaries. The remainder of this paper is organized as follows. Section 2 describes the three-stage algorithm in detail. Simulation results and complexity analysis are presented in section 3. Finally, conclusions are discussed in section 4.

## 2. Algorithm description

### 2.1. Stage one

Many pitch estimation metrics can be used for the first stage. In this instance, NCC is chosen to calculate pitch, similar to [2], even though empirically it has been found that SSD and AMDF provide comparable performance. The definition of NCC is shown below:

$$NCC_t(\tau) = \frac{\sum_{n=-N/2}^{(N/2)-1} x[t+n]x[t+n-\tau]}{\sqrt{\sum_{n=-N/2}^{(N/2)-1} x^2[t+n] \sum_{n=-N/2}^{(N/2)-1} x^2[t+n-\tau]}} \quad (1)$$

where  $\tau_{\min} \leq \tau \leq \tau_{\max}$  and  $N$  is the number of signal samples involved in the calculation.

The best pitch lag is then obtained as

$$\tau_0 = \arg \max_{\tau} NCC_t(\tau) \quad (2)$$

Conventionally the pitch range ( $\tau_{\min}$  to  $\tau_{\max}$ ) should be reasonably large to cover the typical human voice pitch, e.g. [50, 400] Hz in [6]. For PLC, however, the exact pitch is not always needed as pitch multiples are often sufficient. In the proposed algorithm, a much narrower pitch range of just one octave is used with the low bound corresponding to the lowest expected pitch of human speech. For instance, a range of [75, 150] Hz can be used which at a sampling rate of 8kHz corresponds to approximately 106 samples for  $\tau_{\max}$  and 53 samples for  $\tau_{\min}$ , respectively. Using such a narrow pitch range considerably reduces the number of calculation cycles when compared with a full pitch range. Similar to [2], the input signal is also decimated by a factor of 2, which

automatically decimates  $\tau$ . Thus, the overall reduction is approximately a factor of four.

The numerator of (1) can be efficiently computed using a fast multiply-accumulate (MAC) operation, which is available in a typical DSP processor such as CSR's Kalimba [7]. To avoid calculating the relatively expensive square root function in the denominator, the following approximation is used:

$$NCC_i(\tau) = \frac{\sum_{n=-N/2}^{(N/2)-1} x[t+n]x[t+n-\tau]}{\sum_{n=-N/2}^{(N/2)-1} x^2[t+n-\tau]} \quad (3)$$

The term  $\sum_{n=-N/2}^{(N/2)-1} x^2[t+n-\tau]$  can be efficiently computed in

a recursive manner [2]. It is worth noting that the division operation required in the NCC calculations requires only a single cycle instruction on the CSR Kalimba DSP due to the parallelized divide operation.

With a subset of pitch range that covers only low pitch values, the detected pitch period could be a multiple of the "true" pitch period. In many cases this does not pose serious issues for packet loss concealment as speech is considered almost periodic within a short term. However, when there are fast pitch variations, it is preferable to use the "true" pitch period instead of its multiples to minimize the mismatch during waveform substitution. The pitch refinement in the second stage is designed to tackle this issue.

## 2.2. Stage two

In the second stage, the candidate pitch lag value  $\tau_0$  is refined by checking pitch multiples, a process similar to that in [8]. If we let  $\tau'_{\min}$  denote the new minimum pitch lag, the new pitch lag candidate  $\tau_i$  is obtained by dividing  $\tau_0$  with an integer, as shown below:

$$\tau_i = \max\left(\left\lfloor \frac{\tau_0}{i} + 0.5 \right\rfloor, \tau'_{\min}\right) \quad (4)$$

Where  $i = 1, 2, 3, \dots, \left\lfloor \frac{\tau_{\max}}{\tau'_{\min}} \right\rfloor$

$\tau'_{\min}$  generally corresponds to the maximum pitch value in a typical pitch detector. For example, if the target pitch is as high as 400 Hz,  $\tau'_{\min}$  would be 20 samples at a sampling rate of 8kHz. Since  $\tau_{\max}$  defined in Stage one is 106, the maximum number of additional pitch candidates to check is only  $106/20 \approx 5$ . This represents a substantial saving compared to a single stage pitch calculation using a larger pitch range.

The new best pitch lag value is thus determined through evaluating  $\tau_i$  against Eq. (1) using the following pseudo code as shown in Figure 1, where  $\alpha$  is a constant with a typical value between 0.9 and 1. Note that since we use NCC (in contrast to autocorrelation) where the number of samples involved in the calculation is constant, the estimate is unbiased. Hence an  $\alpha$  value less than one makes the algorithm slightly favor pitch multiples, i.e. shorter pitch periods.

---

```

 $\tau'_0 = \tau_0$ 
for  $i = \left\lfloor \frac{\tau_{\max}}{\tau'_{\min}} \right\rfloor \dots 2$ 
    if  $NCC_i(\tau_i) > \alpha \cdot NCC_i(\tau_0)$ 
         $\tau'_0 = \tau_i$ 
    break
end
end

```

---

Figure 1: Pseudo code for pitch refinement.

The above pseudo code exits as soon as a pitch multiple is found that satisfies the condition  $NCC_i(\tau_i) > \alpha \cdot NCC_i(\tau_0)$ . Alternatively, all the pitch multiples can be evaluated and the best candidate  $\tau'_0$  is selected as

$$\tau'_0 = \arg \max_{\tau_i} NCC_i(\tau_i) \quad (5)$$

This method potentially yields more accurate pitch values but at higher computational cost.

In practice, particularly for a highly resource limited platform such as a Bluetooth headset, only one pitch multiple is checked at  $\max\left(\left\lfloor \frac{\tau_0}{2} + 0.5 \right\rfloor, \tau'_{\min}\right)$  for maximum efficiency,

which is also the setting used in the simulation described later in section 3.

## 2.3. Stage three

The estimate of the pitch period calculated in the second stage,  $\tau'_0$ , is optimal in the sense of maximizing the NCC metric. However, on insertion into a voice signal, a replacement packet, that has been generated based on the estimated pitch period, may still contain discontinuities at the boundaries with the audio samples on either side of it. These discontinuities occur because voice signals are not truly periodic.

Typically, cross-fading of the signals on either side of a lost packet is used to reduce the discontinuity at the boundary [2]. This is sometimes referred to as an overlap-add (OLA) operation. In the OLA operation, the ending portion of the signal prior to the degraded packet is multiplied by a down-sloping ramp. This provides a fade out of the signal. The beginning portion of the audio following the degraded packet is multiplied by an up-sloping ramp, to provide a fade in. The current algorithm achieves this using a triangular window. If the overlap length is  $L$ , the window length  $M$  is typically  $2L$ . The overlap length determines how much cross-fading is performed at the boundary. It is normally shorter than the packet length. For example, a common packet length in Bluetooth is 30 samples (HV3/EV3 packet types) at 8 kHz sampling rate [9]. In this case, an overlap length of 10 samples is used to perform cross-fading at the boundary. In contrast to [2], the OLA length is fixed for efficiency and ease of system implementation.

Despite use of an OLA operation, discontinuities often remain a problem and are noticeable as artifacts in the regenerated output voice signal. The third stage of this method is therefore designed to reduce the mismatch between the two segments used for the OLA operation by refining pitch

lag values around  $\tau'_0$ . The operation resembles the pattern matching process used by [6] but is more explicitly applied to the OLA segments. First, several quantities are defined below to facilitate illustration:

- $t_1$  : start of the lost packet
- $t_2$  : end of the lost packet
- $L$  : overlap-add (OLA) length
- $\tau_j$  : a candidate pitch lag value
- $k$  : fine pitch search range

For each candidate pitch lag value  $\tau_j$ , the Sum of Squared Difference (SSD) is computed, which includes the signal segment just before the gap and the corresponding signal that is  $\tau_j$  samples away:

$$D_1(\tau_j) = \sum_{n=1}^L (x[t_1 - n] - x[t_1 - n - \tau_j])^2 \quad (6)$$

where  $\tau'_0 - k \leq \tau_j \leq \tau'_0 + k$

For 8 kHz sampling rate,  $k=4$  is found to be sufficient. To further reduce complexity, only every other pitch candidate  $\tau_j$  in the range of  $[\tau'_0 - k, \tau'_0 + k]$  is evaluated. This results in a set of pitch candidates given by  $[\tau'_0 - 4, \tau'_0 - 2, \tau'_0 + 2, \tau'_0 + 4]$ . Such decimation does not show any noticeable negative impacts. The optimal pitch is then selected that minimizes the SSD function:

$$\tau_0'' = \arg \min_{\tau_j} D_1(\tau_j) \quad (7)$$

If  $L$  future samples after the gap are available, a second value can be computed in a similar manner:

$$D_2(\tau_j) = \sum_{n=1}^L (x[t_2 + n] - x[t_2 + n - \tau_j])^2 \quad (8)$$

The final best pitch lag is then selected which minimizes the overall difference.

$$\tau_0'' = \arg \min_{\tau_j} \frac{D_1(\tau_j) + D_2(\tau_j)}{2} \quad (9)$$

In [2] and [5], the pitch refinement is performed using the same NCC function where  $N$  samples are needed. In the case presented here only  $L$  samples are involved in the distance calculation. For an 8kHz sampling rate,  $N$  is typically in the order of several hundreds while  $L$  is normally below 30 samples. Furthermore, by evaluating pitch using cross-fading segments, the discontinuities at the boundaries can be minimized which results in better voice quality.

### 3. Experiments and results

To evaluate the effectiveness of the proposed algorithm, the ITU standard PESQ (Perceptual Evaluation of Speech Quality) [10] is used as an objective measure. It should be noted that in this paper the PESQ MOS-LQO score [11] is reported, which is converted from the raw PESQ score to allow a linear comparison with MOS. Clean speech files were selected from NOIZEUS [11], which contains 30 IEEE sentences (spoken by three male and three female speakers). The files were corrupted by dropping frames of audio samples at random

locations within the files and the gaps were filled by repeating the last good sample. Loss rates of 5%, 10%, 15%, and 20% were used. Packet sizes of 30 samples and 60 samples at a sample rate of 8kHz were chosen, which correspond to the maximum packet length of HV3/EV3 and 2-EV3 packet types in the Bluetooth protocol [9]. To evaluate the overall system performance as well as the impact of each refinement stage, the corrupted files were processed using several different versions of the algorithm:

Baseline: the pitch range was set to [66, 200] Hz, which is the range used in G.711 Appendix I [2]. No decimation was used. This was treated as a reference system as it has a larger pitch range and higher resolution for pitch estimation. It also serves as an upper bound for the actual G.711 algorithm.

Stage 1: Use stage 1 of the proposed algorithm with a pitch range of [75, 150] Hz.

Stage 1+2: Use stage 1 and stage 2 of the proposed algorithm with a pitch range of [75, 150] Hz.

Stage 1+3: Use stage 1 and stage 3 of the proposed algorithm with a pitch range of [75, 150] Hz.

Stage 1+2+3: Use the complete algorithm including stage 1, 2, and 3. The pitch range is [75, 150] Hz.

Besides the differences listed above, these systems were strictly the same.

Figure 2 shows the PESQ MOS-LQO improvement over the corrupted signal (unprocessed) for a 30-sample packet size. This was derived by subtracting the PESQ scores of unprocessed signals from the scores of the various processed signals. Relative score, rather than the absolute, is presented in order to highlight the comparative performance of proposed algorithm over the baseline G.711 algorithm. Similarly Figure 3 shows the results for a 60-sample packet size.

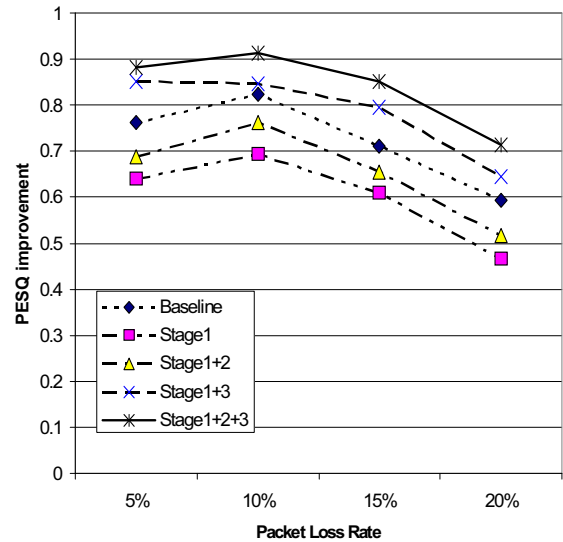


Figure 2: PESQ MOS-LQO improvements of packet loss concealment algorithms for packet size 30-sample at various loss rates

It can be seen from the figures that despite the reduced computational complexity, the proposed three-stage algorithm consistently outperforms the baseline system, which calculates pitch with a larger pitch range and without any decimation. For the three stages in the algorithm, each additional refining stage yields noticeable additional improvements across all the

loss rates, with stage three providing more significant benefits. It is interesting to note that for a packet size of 30 samples, the system with only stage one and stage three outperforms the baseline in all four packet loss conditions. This is only observed at 20 percent loss rate for a packet size of 60 samples. Overall, all systems perform better with the smaller packet size 30.

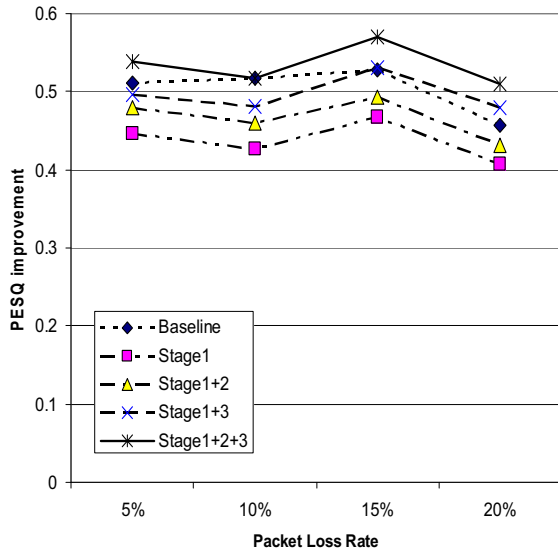


Figure 3: PESQ MOS-LQO improvements of packet loss concealment algorithms for packet size 60-sample at various loss rates

Consistent with the objective measurements, informal listening tests also showed that the system yields significantly better voice quality over the default muting or repeating sample schemes typically used by the Bluetooth CVSD codec.

The algorithm has been implemented on CSR's Kalimba DSP processor [7] for Bluetooth voice communication. To demonstrate the saving of computation, Table 1 presents the assembly instruction cycle count for the proposed algorithm and that of G.711 Appendix I. For the latter, Stage 1 employs a coarse pitch search over a pitch range from 66 Hz to 200 Hz and a fine pitch search is performed over range of -1 to 1 in Stage 2 [2]. It should be noted that the approximation form, as in Eq. (3), was used in estimation for both methods. The original formula used by G.711 Appendix I would be more computationally expensive on a fixed point platform due to the square-root operation. As demonstrated by Table 1, the estimated savings compared to the similarly decimated G.711 Appendix I approach are approximately 32 percent.

Table 1. Instruction cycle count for G.711 Appendix I and the proposed algorithm

	G711	Proposed	Saving Percentage (%)
Stage 1	5940	3720	37.37
Stage 2	580	250	56.90
Stage 3	n/a	460	n/a
<b>Total</b>	<b>6520</b>	<b>4430</b>	<b>32.06</b>

## 4. Conclusions

A highly efficient three-stage pitch estimation algorithm for packet loss concealment has been described. In stage one, pitch is calculated on the decimated signal in a very narrow range. Stage two selects the best pitch candidate based on the pitch determined in stage one and its integer multiples. In stage three the pitch is further refined by selecting the pitch that minimizes distortion at the concatenation boundary where an overlap-add is performed. Simulation results from an objective measure (PESQ) demonstrate that the proposed algorithm consistently outperforms the G.711 style pitch calculation with significantly lower computational cost. Future work will include more comprehensive evaluation of the algorithm under real Bluetooth communication environment prone to bursty errors.

## 5. Acknowledgements

The authors wish to thank Gary Spittle for helpful comments and colleagues at CSR for their support.

## 6. References

- [1] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," *IEEE Network Magazine*, Sept./Oct. 1998.
- [2] ITU-T Recommendation G.711 Appendix I, *A high quality low-complexity algorithm for packet loss concealment with G.711*, 1999.
- [3] A.D. Cheveigne, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, p. 1917-1930, 2002
- [4] M. Ross, H. Shaffer, A. Cohen, R. Freudberg, and H. Manley, "Average magnitude difference function pitch extractor," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, pp. 353-362, 1974
- [5] R. Zopf, "Decimated Bisectional Pitch Refinement" *US patent application 11/734,824*, 2007
- [6] H. Svensson, V. Owall, and K. Kuchcinski, "Implementation aspects of a novel speech packet loss concealment method", *Proc. ISCAS*, pp. 2867 - 2870 Vol. 3, 2005
- [7] Bluecore5-Multimedia Kalimba DSP User Guide, CSR support website, <http://www.csrsupport.com>
- [8] W. C. Chu, *Speech coding algorithms: foundation and evolution of standardized coders*, John Wiley & Sons, Inc., New York, NY, 2003
- [9] B. SIG, "Core specification v2.1 + EDR," *Bluetooth SIG Tech. Rep.*, Jul 2007.
- [10] ITU-T Recommendation P.862, *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, 2001.
- [11] ITU-T Recommendation P.862.1, *Mapping function for transforming P.862 raw result scores to MOS-LQO*, 2003
- [12] Y. Hu and P. Loizou, "Subjective evaluation and comparison of speech enhancement algorithms," *Speech Communication*, 49, 588-601, 2007.