Timing Recovery in Magnetic Recording Systems

Piya Kovintavewat
Faculty of Science and Technology, Nakhon Pathom Rajabhat University
85 Moo 3, Malaiman Rd., Muang District, Nakhon Pathom 73000, Thailand

Abstract— **Timing recovery is a crucial component in a magnetic recording channel detector. Conventional timing recovery techniques are sufficient only when operating at high signal-to-noise ratio (SNR). However, iterative error-correction codes allow reliable operation at very low SNR, where conventional techniques fail. In this paper, we summarize the timing recovery schemes based on a per-survivor processing technique that are capable of working at low SNR. Results indicate that they perform better than conventional schemes.**

I. Introduction

Timing recovery is the process of synchronizing the sampler with the received signal. Sampling at wrong times can have a devastating impact on overall system performance. Therefore, the quality of synchronization is very important in a magnetic recording channel detector.

Conventional timing recovery techniques are based on a decision-directed phase-locked loop (PLL) [1], which consists of a timing error detector (TED), a loop filter, and a voltage-controlled oscillator (VCO). They are adequate only when the operating signal-to-noise ratio (SNR) is sufficiently high. Nonetheless, recent advances in error-control coding have made it possible to communicate reliably at very low SNR, where conventional techniques fail. Many timing recovery schemes that are capable of working at low SNR have been proposed in the literature [2], [3]. In this paper, we summarize all timing recovery schemes that are based on *per-survivor processing* (PSP) [4], a technique of jointly estimating a data sequence and unknown parameters, and also investigate their performances when operating in magnetic recording systems.

This paper is organized as follows. Section II briefly describes a system model. Section III explains how conventional timing recovery works and shows its performance. Per-survivor timing recovery and per-survivor iterative timing recovery are summarized in Sections IV and V, respectively. Finally, Section VI concludes the paper.

II. System Description

We consider the perfectly equalized PR-IV channel model shown in Fig. 1, where the read-back signal can be written as

$$p(t) = \sum_k a_k h(t - kT - \tau_k) + n(t), \tag{1}$$

where $a_k \in \{\pm 1\}$ is an input data sequence with bit period T, $h(t) = q(t) - q(t-2T)$ is a PR-IV pulse, $q(t) = \sin(\pi t/T)/(\pi t/T)$ is an ideal zero-excess-bandwidth Nyquist pulse, and $n(t)$ is additive white Gaussian noise (AWGN) with two-sided power spectral density $N_0/2$. The timing offset, τ_k, is modeled as a random walk model [5] according to $\tau_{k+1} = \tau_k + w_k$, where w_k is an independent and identically distributed (*i.i.d.*) zero-mean Gaussian random variable with variance σ_w^2, and σ_w determines the severity of the timing jitter. The random walk model is chosen because of its simplicity to represent a variety of channels by changing only one parameter.

At the front-end receiver, the read-back signal $p(t)$ is filtered by a low-pass filter (LPF), whose impulse response is $q(t)/T$, to eliminate the out-of-band noise, and is sampled at time $kT + \hat{\tau}_k$, creating

$$y_k = y(kT + \hat{\tau}_k) = \sum_i a_i h(kT + \hat{\tau}_k - iT - \tau_i) + n_k, \tag{2}$$

where $\hat{\tau}_k$ is the receiver's estimate of τ_k, and n_k is *i.i.d.* zero-mean Gaussian random variable with variance $\sigma_n^2 = N_0/(2T)$.

III. Conventional Timing Recovery

Conventional timing recovery is based on a PLL. A decision-directed TED [1] is used to compute the receiver's estimate of the timing error $\epsilon_k = \tau_k - \hat{\tau}_k$, which is the misalignment between the phase of the received signal and that of the sampling clock. In this paper, we consider the well-known Mueller and Müller (M&M) TED algorithm [6], where the estimated timing error is given by

$$\hat{\epsilon}_k = \frac{3T}{16}\{y_k \hat{r}_{k-1} - y_{k-1}\hat{r}_k\}, \tag{3}$$

where $\hat{r}_k$ is the k-th estimate of the noiseless *channel output* $r_k \in \{0, \pm 2\}$. The constant $3T/16$ ensures that there is no bias at high SNR so that $E[\hat{\epsilon}|\epsilon] = \epsilon$, for small ϵ. For simplicity, we assume perfect acquisition by setting $\tau_0 = 0$. Because our model has no frequency offset component, the sampling phase offset can then be updated by a first-order PLL according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi \hat{\epsilon}_k, \tag{4}$$

where ξ is a PLL gain parameter [1]. Eventually, the Viterbi detector [7] performs maximum-likelihood (ML) equalization to determine the most likely input data sequence.

As depicted in Fig. 1, a conventional receiver performs timing recovery and ML equalization separately. Therefore, the overall performance of this system (also referred to as an *uncoded* system) is mainly determined by how good conventional timing recovery is. It has been shown in [2] that conventional timing recovery does not perform well in uncoded systems when the timing error is large. Thus, the need for efficient timing recovery schemes becomes increasingly crucial. Such timing recovery schemes can be developed using per-survivor processing (PSP) techniques [4].

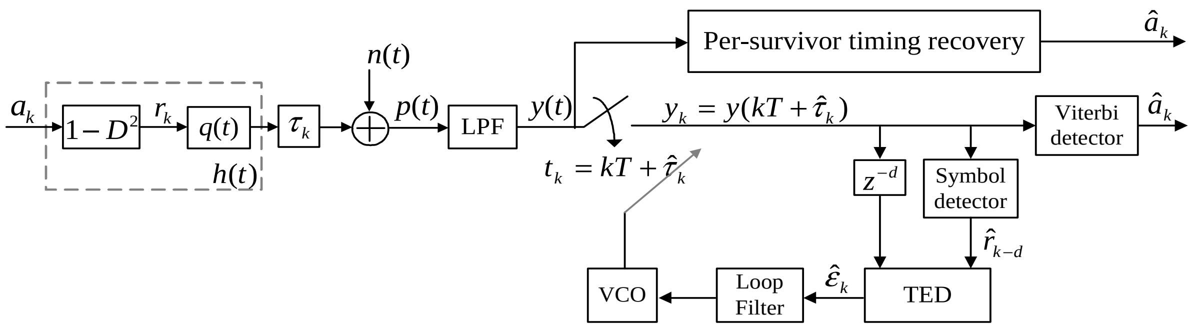

Fig. 1. The perfectly equalized PR-IV channel model with timing recovery.

IV. Per-Survivor Timing Recovery

To improve the performance of the conventional timing recovery, a reliable decision with zero decision delay can be extracted by utilizing the already-given information inside the trellis structure [7]. Specifically, each state transition in the trellis uniquely specifies a corresponding symbol. Hence, at least one state transition in each trellis stage will correspond to a correct decision. Utilizing that decision for the timing update operation will improve the performance of timing recovery.

A. Algorithm Description

With PSP, a new timing recovery scheme called *per-survivor timing recovery* was developed in [8], which jointly performs timing recovery and ML equalization, as also shown in Fig. 1. This scheme works in a similar fashion as the Viterbi algorithm does, except with an additional timing update operation. The key idea is to sample the received analog signal using different sampling phase offsets associated with each state transition. Additionally, each survivor path has its own PLL to update the sampling phase offset. In pseudocode, per-survivor timing recovery can be summarized as follows, where L is the length of an input sequence, and Q is the total number of trellis states in one stage. For details, a reader can refer to [8].

```
Initialize the sampling phase offset and
the path metric
for $k = 1 : L$ (each time instant)
    for $q = 1 : Q$ (each trellis state)
      sample the waveform $y(t)$ using a PLL
      and the sampling phase offset, $\hat{\tau}_k$,
      associated with the starting state
      to get samples $\{y_k\}$
      compute the branch metrics
      compute the path metric
      update the survivor path
      update the next sampling phase
      offset, $\hat{\tau}_{k+1}$, using the information
      obtained from the survivor path that
      leads to state $q$
    end
end
decode a data sequence from the survivor
path that has the minimum path metric
```

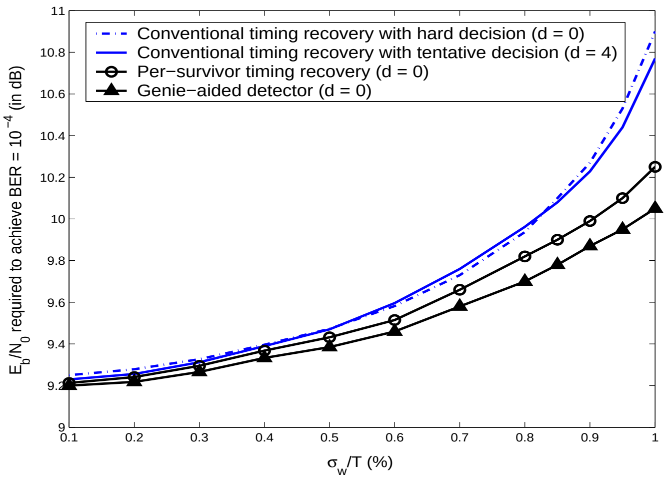

Fig. 2. Performance comparison of different timing recovery schemes [8].

B. Performance Results

Fig. 2 compares the performance of different timing recovery schemes by plotting the E_b/N_0 (in dB) requirement for bit-error rate (BER) 10^{-4} as a function of σ_w/T's, where a parameter d in the parenthesis denotes the total delay in the timing loop. The curve labeled "genie-aided detector" represents conventional timing recovery whose PLL has access to all correct decisions, thus serving as a lower bound for a timing recovery scheme that is based on a PLL. The tentative decision is obtained from the Viterbi detector with a (short) decision delay of $d = 4$. This is done by choosing the *best* survivor path at each time instant, and then the *tentative decision*, $\hat{r}_{k-d}$, is found by moving d steps backward along that survivor path.

Clearly, per-survivor timing recovery performs better than conventional timing recovery, especially when σ_w/T is large. This can be intuitively explained as follows. At each time instant, at least one state transition in each trellis stage will correspond to the correct decision. Using that decision to perform the timing update operation will help improve timing recovery performance. In other words, the PLL is *fully trained* if the correct path is chosen. By following this idea for an entire received signal, the overall performance is improved.

V. Per-Survivor Iterative Timing Recovery

Iterative error-control codes (ECCs) allow reliable operation at low SNR because of their large coding gains [9]. This means that timing recovery must also function at low SNR. A conventional receiver performs timing recovery and error-correction decoding separately. Specifically, conventional timing recovery ignores the presence of ECCs. Thus, it fails to work properly at low SNR.

Theoretically, joint ML estimation of timing offsets and message bits, which will jointly perform timing recovery, equalization, and decoding, is a preferred method of synchronization [10] but its complexity is huge. A solution based on the expectation-maximization (EM) algorithm [11] is also complex. Fortunately, a solution to this problem with complexity comparable to the conventional receiver has been proposed in [12], which will be referred to as *the NBM scheme*. It is realized by embedding the timing recovery step inside the turbo equalizer so as to perform timing recovery, equalization, and decoding jointly. Nonetheless, this scheme requires a large number of turbo iterations to provide a good performance even with a cycle slip [1] detection and correction algorithm as used in [12], especially when timing error is large.

To improve the performance of the NBM scheme, an iterative timing recovery scheme called *per-survivor iterative timing recovery* has been proposed in [13]. It is realized by first applying the per-survivor concept to the Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [14], resulting in PSP-BCJR. Then, per-survivor iterative timing recovery iteratively exchanges soft information between PSP-BCJR and a soft-in soft-out (SISO) decoder.

Consider the *coded* PR-IV channel model shown in Fig. 3. The message bits $\{x_k\}$ are encoded by a serial concatenation of an error-correction encoder, an s-random interleaver [9] (i.e., the π block), and a $1/(1 \oplus D^2)$ precoder. The read-back signal $p(t)$ expressed as given in (1) is filtered by an LPF and is sampled at time $kT + \hat{\tau}_k$, creating y_k according to (2).

We also assume perfect acquisition. Because our model has no frequency offset component, the sampling phase offset can then be updated by a first-order PLL according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi \frac{3T}{16}\{y_k \tilde{r}_{k-1} - y_{k-1}\tilde{r}_k\}, \tag{5}$$

where $\tilde{r}_k$ is the k-th *soft estimate* of the *channel output* $r_k \in \{0, \pm 2\}$ given by [12]

$$\tilde{r}_k = E[r_k|y_k] = \frac{2\sinh(2y_k/\sigma_n^2)}{\cosh(2y_k/\sigma_n^2) + e^{2/\sigma_n^2}}. \tag{6}$$

In the conventional receiver, conventional timing recovery is followed by a turbo equalizer [15] (see Fig. 3), which iteratively exchanges soft information between the SISO equalizer for the precoded PR-IV channel and the SISO decoder.

A. Algorithm Description

PSP-BCJR [13] is developed by embedding the timing recovery process inside the BCJR equalizer so as to perform timing recovery and ML equalization jointly. It works in a similar manner as per-survivor timing recovery does. Nevertheless, we update the timing estimate at each state based on the incoming branch that contributes the most to the state information. In pseudocode, PSP-BCJR can be summarized as follows.

```
FORWARD RECURSION:
Initialize the forward sampling phase
offset and the forward state information
for $k = 1 : L$ (each time instant)
    for $q = 1 : Q$ (each trellis state)
      sample the waveform $y(t)$ using a PLL
      and the forward sampling phase
      offset, $\hat{\tau}_k$, associated with the
      starting state to get samples $\{y_k\}$
      compute the BCJR branch metrics
      compute the forward state
      information
      update the next forward sampling
      phase offset, $\hat{\tau}_{k+1}$, based on the most
      likely incoming branch that leads to
      state $q$
    end
end

BACKWARD RECURSION:
Initialize the backward sampling phase
offset and the backward state information
for $k = L : 1$ (each time instant)
    for $p = 1 : Q$ (each trellis state)
      sample the waveform $y(t)$ using a PLL
      and the backward sampling phase
      offset, $\hat{\tau}^b_{k+1}$, associated with the
      starting state to get samples $\{y^b_k\}$
      compute the BCJR branch metrics
      compute the backward state
      information
      update the next backward sampling
      phase offset, $\hat{\tau}^b_k$, based on the most
      likely incoming branch that leads to
      state $p$
    end
    compute a posteriori log-likelihood
    ratio (LLR) of the input bit, $\lambda_k$,
    using the forward and the backward
    state information
end
```

B. Performance Results

The per-survivor iterative timing recovery scheme is easily obtained by discarding the front-end PLL in Fig. 3 and replacing the BCJR equalizer with PSP-BCJR.

Consider a rate-8/9 system in which a block of 3636 message bits is encoded by a rate-1/2 recursive systematic convolutional encoder with a generator polynomial $[1, \frac{1\oplus D\oplus D^3\oplus D^4}{1\oplus D\oplus D^4}]$, and then punctured to a block length of 4095 bits by retaining only every the eighth parity bit. The punctured sequence passes through an s-random interleaver with $s = 16$ to obtain an interleaved sequence of b_k. Both the SISO equalizer and the SISO decoder are implemented based on BCJR. We use the PLL gain parameter designed based on minimizing the RMS

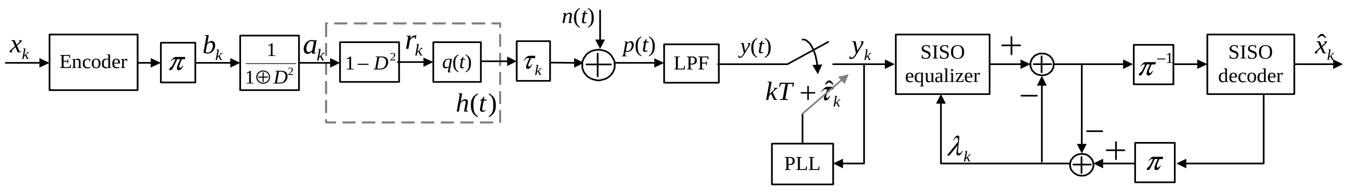

Fig. 3. A coded PR-IV channel model.

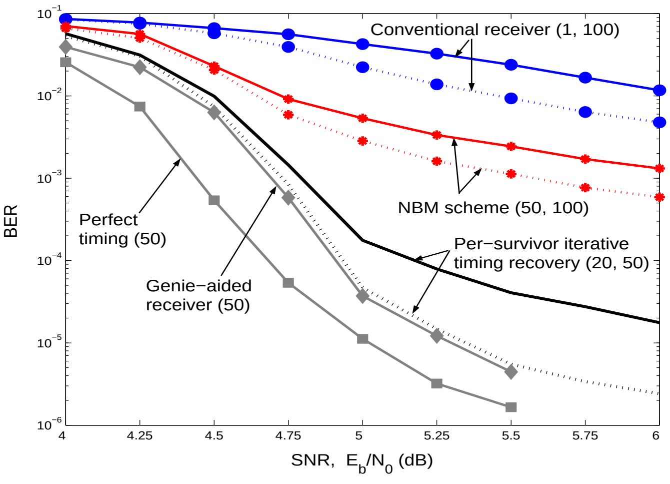

Fig. 4. Performance comparison with $\sigma_w/T = 1\%$ [13].

timing error, $\sigma_\epsilon = \sqrt{E[(\tau_k - \hat{\tau}_k)^2]}$, at $E_b/N_0 = 5$ dB. Each BER point was computed using as many data packets as possible until at least 100 sectors in error were collected at the 100-th iteration.

Fig. 4 compares the BER performance of different schemes for the system with a severe random walk parameter $\sigma_w/T = 1\%$, which implies a high probability of occurrence of a cycle slip. Note that the number inside the parenthesis in Fig. 4 indicates the total number of iterations used to generate each curve. The curve labeled "Perfect timing" represents the conventional receiver that uses $\hat{\tau}_k = \tau_k$ to sample $y(t)$. Clearly, per-survivor iterative timing recovery outperforms both the conventional receiver and the NBM scheme. This is because per-survivor iterative timing recovery can correct a cycle slip much more efficiently than the NBM scheme [13]. In addition, it performs similar to the genie-aided receiver and loses approximately a 0.35 dB relative to the system with perfect timing at BER = 10^{-5}.

VI. Conclusion

We summarized the timing recovery schemes that are based on per-survivor processing. It is clear that per-survivor timing recovery can perform better than conventional timing recovery, especially when the SNR is low and the timing error is large. By exploiting the presence of error-correction codes, per-survivor iterative timing recovery can perform even better than per-survivor timing recovery, but at the expense of increased complexity and the requirement of batch processing.

Simulation results has illustrated that per-survivor iterative timing recovery performs better than other iterative timing recovery schemes, especially when the timing error is severe. This is because it can correct a cycle slip much more efficiently than other schemes. Additionally, Kovintavewat *et al* [16] have proposed a reduced-complexity version of per-survivor iterative timing recovery, which has been shown to perform better than (full-complexity) per-survivor iterative timing recovery at low to moderate complexity regime.

References

[1] J. W. M. Bergmans, *Digital baseband transmission and recording*. Boston, Massachusetts: Kluwer Academic Publishers, 1996.

[2] P. Kovintavewat, *Timing recovery based on per-survivor processing*. PhD thesis, Georgia Institute of Technology, Georgia, USA., Oct 2004.

[3] J. R. Barry, A. Kavčić, S. W. McLaughlin, A. R. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, pp. 89–102, Jan 2004.

[4] R. Raheli, A. Polydoros, and C.-K. Tzou, "Per-survivor processing: a general approach to MLSE in uncertain environments," *IEEE Trans. Commun.*, vol. 43, pp. 354–364, Feb/Mar/Apr 1995.

[5] A. N. Andrea, U. Mengali, and G. M. Vitetta, "Approximate ML decoding of coded PSK with no explicit carrier phase reference," *IEEE Trans. Commun.*, vol. 42, pp. 1033–1039, Feb/Mar/Apr 1994.

[6] K. H. Mueller and M. Müller, "Timing recovery in digital synchronous data receivers," *IEEE Trans. Commun.*, vol. COM-24, pp. 516–531, May 1976.

[7] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363–378, May 1972.

[8] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, "Per-survivor timing recovery for uncoded partial response channels," in *Proc. of ICC'04*, vol. 27, no. 1, pp. 2715–2719, Paris, Jun 2004.

[9] C. Heegard and S. B. Wicker, *Turbo Coding*. Boston, Massachusetts: Kluwer Academic Publishers, 1999.

[10] H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: John Wiley & Sons, Inc., 1997.

[11] C. N. Georghiades and D. L. Snyder, "The expectation-maximization algorithm for symbol unsynchronized sequence detection," *IEEE Trans. Commun.*, vol. 39, pp. 54–61, Jan 1991.

[12] A. R. Nayak, J. R. Barry, and S. W. McLaughlin, "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Trans. Magnetics*, vol. 38, pp. 2295–2297, Sept 2003.

[13] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, "Per-survivor iterative timing recovery for coded partial response channels," in *Proc. of Globecom'04*, vol. 4, pp. 2604–2608, Texas, November 29 – Dec 2004.

[14] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 248–287, Mar 1974.

[15] D. Raphaeli and Y. Zarai, "Combine turbo equalization and turbo decoding," in *Proc. of Globecom'97*, vol. 2, pp. 639–643, Nov 1997.

[16] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, "Reduced-complexity per-survivor iterative timing recovery for coded partial response channels," *IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP 2005), Philadelphia, USA, vol. 3, pp. iii/841 iii/844, Mar 2005.