Neural Network Based Spread Spectrum PN Code Acquisition System

Said E. El-Khamy
Department of Electrical Engineering
Alexandria University
Alexandria, Egypt, 21544
elkhamy@alex.eun.eg

Erol Gelenbe, Hossam Abdelbaki
School of Computer Science
University of Central Florida
Orlando, FL 32816
{erol,ahossam}@cs.ucf.edu

Abstract

In this paper, a neural network based direct sequence spread spectrum code synchronization system is proposed. This system is based on training a recurrent random neural network (RNN) model on all the possible phases of the used spreading code. The trained network can then be used at the receiver for the initial coarse alignment of the local code phase and the received code. One advantage of this technique over the conventional synchronization techniques is that the phase of the received PN code can be decided without searching the potential code phases. Also the RNN, after being trained, can have a simple hardware realization that makes it candidate for implementation as a dedicated chip. This makes the neural network based technique faster and more robust than the conventional techniques. Computer simulations, carried out on maximal length sequences of length $N = 7$ and $N = 15$, show that the proposed system can effectively indicate the phase of the received code even with very low signal to noise ratios.

1 Introduction

Spread spectrum communication system requires its locally generated pseudo-noise code to be properly time aligned with the code sequence imbedded in the received signal. If absolutely precise clocks were available and the range between the transmitter and the receiver was known, the alignment problem would not exist, however, there is always an initial timing uncertainty due to range uncertainty. The task of achieving and maintaining code synchronization is delegated to the receiver. The initial coarse alignment of the local code phase is called acquisition while the fine adjustment of the code phase is called tracking.

In this paper, we considered the application of the Random Neural Network (RNN) model [1, 2] to the problem of direct sequence spread spectrum (DSSS) PN code acquisition in Gaussian channel with additive white noise (AWGN channel). The proposed system consists of two layers, the input layer or what can be called the association layer and the output layer which acts as a classification layer.
The sections of this paper are organized as follows. Section 2 provides a brief overview of the conventional techniques used for PN code acquisition. Section 3 introduces the RNN model. The structure of the proposed RNN based code synchronizer is presented in Section 4. Section 5 presents an experimental evaluation of the proposed system as well as the conventional systems. Finally, Section 6 offers the conclusion.

2 Conventional Techniques for PN Code Acquisition

Two approaches are used for PN code acquisition: serial search and parallel search. In this Section, these approaches will be reviewed.

2.1 Serial Search Technique

In this technique [3], the receiver searches serially through all the potential code phases that differ from each other by one chip shift until synchronization is achieved. The output of this system is processed to determine whether or not despreading has occurred, indicating that the phase of the locally generated PN code was correct.

A false alarm occurs when the acquisition circuits decide, because of the noise and interference, that the local code phase is close to the correct value, when in fact, it is not close. It is important to minimize the probability of false alarms because much time is lost when the system enters the tracking mode with the incorrect code phase.

2.2 Parallel Search Technique

This is a highly efficient technique for initial synchronization in which, all the possible code phases are tested during the same time interval using separate matched filter correlators. Each of the matched filters is adjusted on a certain code phase. The matched filter continuously correlates the received signal and generates maximum output when it receives the code phase that matches its adjusted code. This maximum output can be sensed by the exceeding of a certain threshold value and used to start the receiver code generator at the appropriate phase. In some situations the threshold is exceeded because of the noise and therefore the code generator is triggered by a false alarm. The complexity of practical implementation of such filters is too great [4]. Most often, the serial search technique is used although it can lead to very long acquisition times.

2.3 Notations and Definitions

The received signal, \( r(t) \), can be expressed as

\[
r(t) = s(t - \tau) + n(t), \quad t \in [0, T_c],
\]

\[
s(t) = \sqrt{2P} a(t) \cos(\omega_c t + \theta), \quad \theta \in [0, 2\pi],
\]

\[
a(t) = \prod_{i=0}^{N-1} (t - iT_{ch}), \quad a_i \in -1, +1.
\]

The notation introduced in (1), (2), and (3) is defined as follows.
\begin{itemize}
  \item $T_c$ is the code duration.
  \item $\tau$ is the channel delay.
  \item $s(t)$ is the direct sequence modulated signature waveform.
  \item $n(t)$ is white Gaussian noise with unit power spectral density. It models thermal noise plus other noise sources unrelated to the transmitted signals.
  \item $P$ is the carrier power, $\omega_c$ is the carrier frequency, and $\theta_c$ is the phase angle.
  \item $a(t)$ is the spreading code.
  \item $\prod(t)$ is a rectangular pulse of unit height and duration $T_{ch}$ It is assumed that $N$ is the length of the spreading code and one code period $a = [a_0, a_2, \cdots, a_{N-1}]$ is used for spreading the signal per symbol so that $T_c = NT_{ch}$.
\end{itemize}

The spreading PN codes are assumed to be maximal length sequences of period $N$ \cite{5, 6}. Such sequences have $N$ different phases which can be defined by the $N$-tuple sequence:

\begin{align*}
  a^0 &= [a_0, a_2, \cdots, a_{N-1}], \\
  a^1 &= [a_1, a_2, \cdots, a_{N-1}, a_0], \\
  a^i &= [a_i, a_{i+1}, \cdots, a_{i-1}], \\
  a^{N-1} &= [a_{N-1}, a_0, \cdots, a_{N-1}, a_{N-2}].
\end{align*} \tag{4}

The aim of the detector is to detect the presence of the signal $s(t)$ in the received signal $r(t)$. Taking delays into consideration, the output of the matched filter $y(t)$ can be given by:

\begin{equation}
  y(t) = \int_0^{T_c} r(t-T_d) s(t-T_d) dt
\end{equation} \tag{5}

Where $T_d$ is the delay due to ranging and $\hat{T}_d$ is the initial estimate of the delay at receiver.

In vector form, if the conventional matched filter is adjusted to a certain code phase $C'$ and assuming that the received signal, $R$, is binary phase shift keying (BPSK) direct sequence spreaded, the sampled output of the filter is given by,

\begin{equation}
  Y = \text{sgn}(R.C')
\end{equation} \tag{6}

and the probability of symbol error is

\begin{equation}
  P_{e1} = \frac{1}{2} erfc \left( \sqrt{\frac{E_b}{N_0}} \right)
\end{equation} \tag{7}

Where $E_b$ is the energy per bit and $N_0$ is the spectral density of the additive white Gaussian noise. Note that Equation 7 assumes the reception of the correct phase of the code.
3 The Random Neural Network (RNN) Model

In the random neural network model [1] signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation and negative signals represent inhibition. Each neuron’s state is a non-negative integer called its potential, which increases when an excitation signal arrives to it, and decreases when an inhibition signal arrives. The RNN model has a general training algorithm [2] that can be used for feedforward, recurrent, or arbitrary defined network structures. We have developed a free simulator, RNNSIM v.2, available with full documentation and simulation examples at the MathWorks Inc. public directory [7].

![Neuron Representation](image)

Fig. 1. Representation of a neuron in the RNN.

By appropriately mapping external signals and neuron states into certain physical quantities, the RNN has been successfully applied to several engineering problems including solving NP-complete optimization problems [8, 9] and texture image generation [10]. The empirically observed success of the network can be justified using the theoretical results reported in [11], where the authors showed that for any continuous multivariate function $f$ on a compact set, it is possible to construct a RNN model with a predefined structure that can implement a mapping close enough to $f$ in some precise sense to a given degree of accuracy.

4 Neural Network Based Acquisition system

In this section, we first introduce the neural matched filter detector (NMF) as a technique to mimic the function of the conventional matched filter detector through training. Then we extend the idea of the NMF so that it can be used as an efficient DSSS acquisition technique.

4.1 The Neural Matched Filter (NMF)

The matched filter is an optimal linear detector in the sense that it maximizes the signal-to-noise ratio of the observed signal. If the noise is Gaussian, it can be proved that the matched filter minimizes the probability of error among all linear detectors [12]. In this subsection we will introduce the neural matched filter (NMF) detector. This filter is based on training a RNN on a certain code and its inverse so that it can be used, after training, as a demodulator for the data modulated with this code.
The NMF consists of two layers as shown in Figure 2. The input layer, which has $N$ neurons ($N$ is the code length), is a fully interconnected layer. This layer is connected to the output layer, which consists of 2 neurons, through feed-forward connections. The input layer accepts the bits of the received perturbed codeword at the end of the transmission channel and acts as an association layer which, through training, can extract the relation between the different bits of each codeword and encode this relation into the weight matrix. It produces the output $[1 \ 0]$ when a certain code is received and $[0 \ 1]$ when the inverse code is received. The final decision is made by simply subtracting the two outputs.

![Fig. 2. Structure of the neural matched filter (NMF).](image)

To illustrate the response of the NMF and its relation with the response of the conventional matched filter, we designed a NMF to decode an m-sequence of length $N = 7$. Simply during the training, we associate the binary input patterns $C_1 = 0100111$ and $C_2 = \overline{C_1} = 1011000$ with the output patterns $O_1 = [1, 0]$ and $O_2 = [0, 1]$ respectively (note that during the training, the input patterns are transformed to bipolar levels). Thus the final output, given by $O_1 - O_2$, will be 1 or $-1$. Before the training, the weight matrices $W^+$ and $W^-$ were initialized to small random numbers between 0 and 1. The rate of the output neurons and the learning rate were set to 1 and 0.1 respectively. The training stopped after 1000 iterations when the average mean squared error has reached 0.003.

To test the proposed NMF, all the possible codes with $N = 7$ starting from 0000000 (which represent the code with index 0) and ending with 1111111 (which represent the code with index 127) were applied to both the NMF and the conventional matched filter adjusted to $C_1$. The response of the NMF is shown in Figure 3(a)-(e) after 1, 10, 50, 100, and 1000 training iterations respectively. The response of the conventional matched filter (CMF) is shown in 3(f).

From Figure 3 we notice that the NMF and the CMF both produce the correct decision which is 1 and $-1$ when the codes $C_1$ (with index 39) and $\overline{C_1}$ (with index 88) were applied respectively. The interesting notice is that although the NMF is trained only on 2 codes, its response reaches the response of the CMF when the average mean squared error becomes very small (Figure 3(e)). Also from Figure 3(a)-(e), the learning process of the NMF can be noticed.

By using this structure, we have overcome the known dilemma of choosing the number of hidden neurons in the traditional feed forward neural network models. One advantage of the proposed NMF over CMF is that it can be trained not only to detect a certain code but also to suppress the effect of some other codes. This can be of great advantage in communication systems.
involving multiple users with different spreading codes (code division multiple access) [12, 13].

Fig. 3. Response of the neural matched filter after (a) one iteration (b) 10 iterations (c) 50 iterations (d) 100 iterations (e) 1000 iterations. (f) Response of the conventional matched filter.
4.2 Neural Matched Filter Synchronizer

In a previous paper [14], we introduced a code acquisition system using a conventional three layer feed forward neural network. The network was trained on back-propagation algorithm. That model produced good classification results but the problem is that it was not immune to channel noise since it was based on training a matched filter to recognize a certain code and suppress all other codes with the same length. This means that if the noise produced an error in the transmitted code, the code will not be detected correctly.

The current proposed model, as shown in Figure 4, consists of two layers, the input or association layer is similar to the input layer of the NMF introduced in Section 4.1. On the other hand, the output layer, which will act as a classification layer, consists of $N$ neurons. The purpose of this model is to simulate the function of the parallel search synchronizer introduced in section 2.2. Instead of acquiring the initial synchronization through using a bank of matched filters, the proposed network is trained on all the available potential code phases and gives one decision indicating the estimated phase of the input code. In the next subsections we introduce the classification technique and the training set used in designing the proposed NMF synchronizer.

![Fig. 4. Structure of the neural matched filter (NMF) synchronizer.](image)

4.3 Classification Technique

Let $\mathcal{T} = \{ (\mathbf{x}, c) \}$ be a training set of $N$ vectors, where $\mathbf{x} \in \mathcal{R}^N$ represents a certain phase of the code under consideration and $c \in \mathcal{I}$ is its class label from an index set $\mathcal{I}$. Let $\mathbf{y} \in \mathcal{R}^N$ be the desired output vector associated with the input vector $\mathbf{x}$. In our application, we relate $c$ to $\mathbf{y}$ via the following relation: $c = \{ j : y_j < y_k \quad k = 1, 2, \ldots, N \}$ ($c$ is the index of the output neuron that produces the minimum value). The NMF synchronizer acts as a mapping $\mathcal{R}^N \rightarrow \mathcal{R}^N$, which assigns a vector in $\mathcal{R}^N$ to each vector in $\mathcal{R}^N$ or equivalently, it can be considered as a mapping $C : \mathcal{R}^N \rightarrow \mathcal{I}$, which assigns a class label in $\mathcal{I}$ to each vector in $\mathcal{R}^N$. The RNN specifies a partitioning of the different phases of the code into regions $R_j \equiv \{ \mathbf{x} \in \mathcal{R}^N : C(\mathbf{x}) = j \}$, where $\bigcup_j R_j = \mathcal{R}^N$ and $\bigcap_j R_j = \emptyset$. It also induces a partitioning of the training set into sets $\mathcal{T}_j \subset \mathcal{T}$ where $\mathcal{T}_j \equiv \{ (\mathbf{x}, c) : \mathbf{x} \in R_j, (\mathbf{x}, c) \in \mathcal{T} \}$. A training pair $(\mathbf{x}, c) \in \mathcal{T}$ is misclassified if $C(\mathbf{x}) \neq c$. The performance measure of interest is the empirical error fraction $E$ of the classifier, i.e. the
fraction of the training set or testing set which is misclassified:

\[
E = \frac{1}{N} \sum_{(x,c) \in T} \delta(c, C(x)) = \frac{1}{N} \sum_{j \in \mathcal{I}} \sum_{(x,c) \in T_j} \delta(c, j)
\]

where \( \delta(c, j) = 1 \) if \( c \neq j \) and 0 otherwise.

### 4.4 Training Patterns

In the training process, all the potential code phases of the code under consideration were used. Thus, for m-sequence of length \( N \), we have \( N \) inputs, \( N \) patterns, \( N \) outputs and \( \mathcal{I} = \{1, 2, \ldots, N\} \). To evaluate the performance of the NMF synchronizer and compare it with the parallel search technique, the two m-sequence codes, 0100111 and 111010110010001 were used in the training.

As an example illustrating the training patterns, the second phase of the first code, 1001110 will be associated with the target output 1011111 and hence its class label is \( c = 2 \). The training and the evaluation processes are carried out on the bipolar versions of the codes. RNNSIM v.2 [7] is used for the training and evaluation.

### 5 Simulation Results

Two neural networks have been trained. One is trained on the potential phases of the m-sequence of length \( N = 7 \), 0100111, and the other on the potential phases of the m-sequence of length \( N = 15 \), 111010110010001. For each network, two different simulations have been carried out.

In the first simulation, the probability of error is recorded, in an AWGN channel, versus the signal to noise ratio (SNR) in \( \text{dB} \) as shown in Figure 5. Since the power spectral density of the DSSS signal is reduced below the noise level, very low SNRs have been used in simulation (-20dB). Figure 5a shows the probability of error for the NMF synchronizer, the parallel search synchronizer and the synchronized matched filter (theoretical value calculated from Equation 7) for the m-sequence of length \( N = 7 \). Figure 5b shows error probabilities for the m-sequence of length \( N = 15 \).

In the second simulation, AWGN with \( SNR = 3 \text{dB} \) has been added to the transmitted codes so as to produce bit errors ranging from 0 to 10. The resulting erroneous codes were then applied to the trained NMF synchronizer and the parallel matched filters and the probability of error has been recorded as shown in Figure 6.

From Figure 5 It can be noticed that the probability of error for the parallel search technique is less than that of the proposed NMF synchronizer, on the other hand, the complexity of NMF synchronizer is much less than that of the parallel search (only one network is used instead of \( N \) separate matched filters). Figure 6 shows that the response of the two techniques is almost same with respect to the number of erroneous bits in the applied code.

As an example of the phase classification technique, consider the first phase of the 7 bits m-sequence 0100111 is to be transmitted through an AWGN channel with \( SNR = -4 \text{dB} \). The received perturbed code, \([−1.2870, 0.8819, −2.7309, 0.5396, 1.4613, −1.1092, −1.5808]\), is then applied to the NMF synchronizer and each of the seven matched filter in the parallel search synchronizer. The corresponding output vector of the NMF synchronizer is \([0.20, 1.00, 0.88, 0.32, 0.98, 0.54, 0.49]\). The minimum appears in the first location indicating that the phase of the input
noisy code is 1. On the other hand, the outputs of the matched filters are [3.13, −9.59, 0.93, 0.47, −1.01, −0.7964, 3.0344]. The maximum value is produced by the first matched filter indicating also that the first phase of the code is received.

**Fig. 5.** Response of the neural matched filter synchronizer versus SNR for (a) 7 bit m-sequence and (b) 15 bit m-sequence.

**Fig. 6.** Response of the neural matched filter synchronizer versus number of erroneous bits for (a) 7 bit m-sequence and (b) 15 bit m-sequence.

### 6 conclusion

In this paper, an efficient acquisition technique for initial synchronization of direct sequence spread spectrum signals has been presented. First the neural matched filter (NMF) was introduced as a technique for decoding spread spectrum modulated data. The NMF is trained on the code under
consideration using the RNN learning algorithm. After sufficient training, it has been shown through simulation that the performance of the NMF mimics that of the conventional matched filter receiver. The NMF synchronizer is then introduced. The model is trained on all the potential phases of the spreading PN code. It has been shown through simulation that the neural network based synchronization is comparable to the parallel search synchronization. The proposed model can acquire acquisition in time periods which are much shorter than that of the serial search technique. It also has a simple hardware realization which is much simpler than that used to implement the parallel search technique.

References


