A Computationally Intelligent Fast Acquisition Algorithm for Deep Space Optical Communications

Kourosh Rahnamai[*], Payman Arabshahi, and Tsun-Yee Yan

Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, MS 238-343
Pasadena, CA 91109 USA
[*rahnamai,payman,yan*]*@dsp.jpl.nasa.gov*

Abstract— **An essential requirement in the design and implementation of optical receivers for deep space communications is the development of a low-complexity, fast, and intelligent frame and symbol synchronization technique. In this paper we present the results of a sub-sequence pattern matching technique that detects and isolates the data frame boundaries based on partially received frames. This is in contrast to the traditional correlation based methods which are not as computationally efficient, and require full reception of the acquisition sequence prior to synchronization. This technique allows fast convergence and detection of the boundaries with as low as 2-6 bytes of received synchronization word. In addition, a confirmation method is proposed which allows tuning of the receiver detection module for a given probability of false boundary detection.**

I. Introduction

One of the NASA technology development programs at the Jet Propulsion Laboratory aims to increase the information return capability from deep-space missions by at least an order of magnitude through the use of laser communications. The optical receiver under design for this project employs pulse position modulation (PPM). An integral part of the receiver is its signal acquisition subsystem which will perform and maintain the necessary decoder timing through subsequent slot, symbol, and frame synchronization. This is done initially by acquiring a specific, known, pulse sequence, known as the acquisition sequence.

Previous work in this area has included theoretical studies of slot timing in photodetecting receivers [1]; design of analog correlators and slot gating, or design of digital synchronizers in which time samples are used for loop control [2]; and use of full record of acquisition sequences [3]. As part of a novel approach, this report addresses two important criteria in designing an acquisition and symbol synchronization algorithm. The first issue is the desirability of a reduced-complexity acquisition technique that does not employ correlations or transform domain processing. Secondly, the technique should be able to perform fast acquisition of the signal, without having to wait for the full word length (256 in our case).

[*] Kourosh Rahnamai's work was performed at JPL while he was on leave from Western New England College, Springfield, MA.

II. Signal Characteristics

In PPM, information is conveyed by the time interval in which the pulse is transmitted. Fast acquisition of the signal is dependent on detection of the pulses in noise, and correct estimation of the time location of the detected pulse.

Figure 1 illustrates the layered data format structure for the received signal. The source data packets are grouped into frames of 1115 bytes. The frame size is selected to match the standard, interleaving depth 5, (255,223) Reed-Solomon code (used for encoding). In addition to encoding, a transfer frame header is attached to each of the transfer frames. Also, to facilitate symbol recovery, a word synchronization sequence is inserted periodically into the data stream. This sequence will be recognized on the receiving end, and used to determine the PPM symbol boundaries.

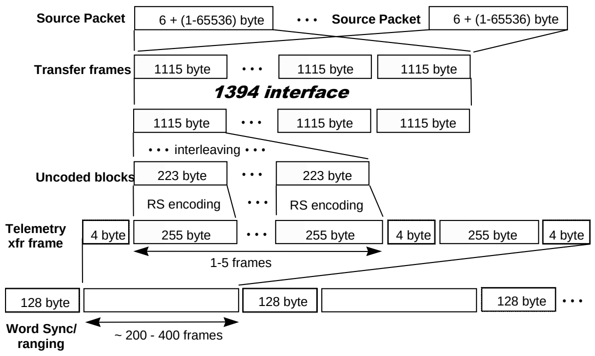


Fig. 1. Layered telemetry format showing the structure of source packets, transfer frames, Reed-Solomon encoding, and word synchronization sequence headers.

After encoding, the data is then PPM modulated on the downlink. The transmitter module uses a fixed slot time period (20 ns), and controls the downlink data rate by modifying the amount of dead time between PPM data words (see Fig. 2).

III. Receiver Structure

A block diagram of the receiver's hybrid front-end is shown in Fig. 3.

An avalanche photodiode (APD) converts the received optical signal into electronic pulses. After the optical

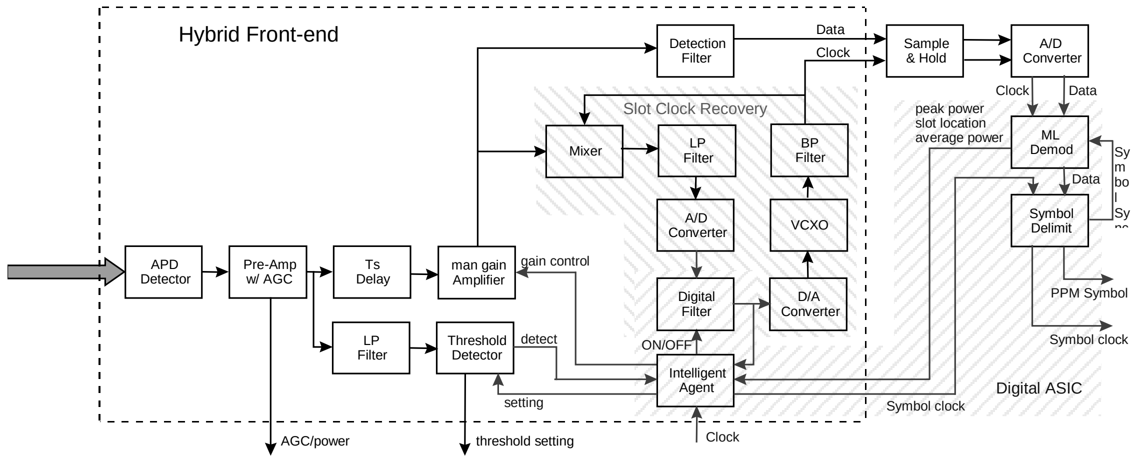


Fig. 3. Block diagram of the PPM receiver.

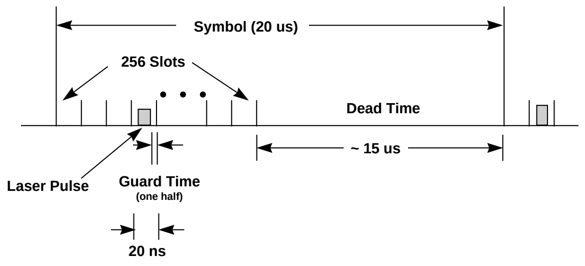


Fig. 2. Pulse Position Modulation timing diagram: the slot width T_l is 20 ns; the symbol width T_s is 20 μs; there are $N_s = 256$ slots in a symbol, within which a signal pulse can occur; there are $N_d \approx 750$ dead-time slots in a symbol; and there are approximately 1,000 $(N_s + N_d)$ total slots per symbol.

PPM pulsed are detected, an automatic gain control (AGC) circuitry amplifies the signal and delays it by one time slot.

The signal is then threshold detected. This aids slot synchronization and helps control the operation of the phase lock loop used for slot clock recovery. The recovered slot clock is then used for subsequent symbol and frame synchronization, and maximum-likelihood demodulation.

Operation of the receiver is controlled by an intelligent agent (IA). The IA will adaptively adjust the threshold by monitoring the signal and noise statistics collected over a period of time. In addition, it will “predict” the time of arrival of the next pulse in the acquisition sequence. This will reduce the false alarm probability and significantly narrow down the time interval over which the threshold detector will operate.

This will also reject spurious spikes that may happen during dead-time, masquerading as a valid signal. By selectively turning the digital filter on or off, based on the above information, the IA will dramatically boost the effective slot clock recovery phase lock loop signal-to-noise ratio by “skipping” integration over noise-only periods, where one is certain of the absence of the pulse signal.

The IA has furthermore a built-in capability to determine, within a short time, the location of a given pulse within its symbol, and within the overall acquisition sequence. The algorithm outlined in section IV is precisely this function of the IA, used to establish symbol synchronization through sub-sequence pattern matching.

An alternative proposed solution for symbol synchronization employs a correlation based method, as summarized below.

The PPM symbol synchronization uses a modified pseudo-noise (PN) sequence known as a maximal length shift register sequence, or *m*-sequence. This sequence of length N (taking on values ± 1) has the property that when it is correlated against a cyclically shifted version of itself, the resulting correlation value is always -1, unless there is no shift, in which case the correlation value is N.

Once the signal is digitized to one value per slot, the receiver collects 128 frames, each consisting of $256 + N_d$ consecutive slot values, and multiplexes them together to form a single sequence of length $256 + N_d$. Note that the starting time slot for the collection of frames is random (since this time is the very quantity that we are trying to estimate). The multiplexed word consists of slot by slot additions of the 128 frames, and is essentially a shifted, noisy version of the transmitted *m*-sequence. The multiplexed frame is then correlated with every integer shift of the known *m*-sequence. The shift corresponding to the maximum correlation value is the estimated PPM symbol offset.

IV. Fast Symbol Synchronization

In this study we use a sub-sequence pattern matching technique to identify the received symbol and frame boundaries for establishing symbol synchronization.

To demonstrate the algorithm concept and in order to simplify the illustration, we refer to a 32-byte *m*-

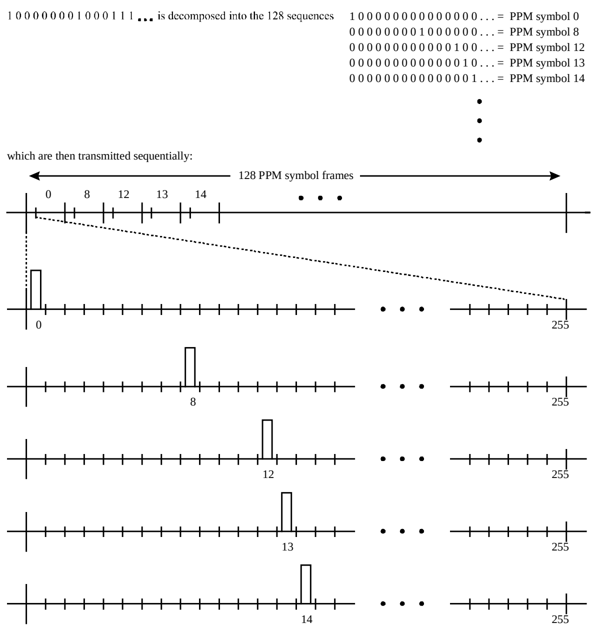
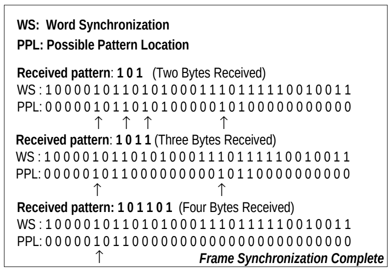

1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 ... is decomposed into the 128 sequences
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . = PPM symbol 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 . . . = PPM symbol 8
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 . . . = PPM symbol 12
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 . . . = PPM symbol 13
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 . . . = PPM symbol 14
which are then transmitted sequentially:
128 PPM symbol frames
0
8
12
13
14
0
255
8
255
12
255
13
255
14
255

Fig. 4. Concept of the PN coded frame synchronization sequence. This sequence will be inserted between super frames to allow synchronization of the PPM decoder.

sequence as shown in Fig. 5 (all subsequent simulation results are based on a 128 byte *m*-sequence).



WS: Word Synchronization
PPL: Possible Pattern Location

```
Received pattern: 1 0 1   (Two Bytes Received)
WS : 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1
PPL: 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
               ↑     ↑   ↑               ↑
Received pattern: 1 0 1 1 (Three Bytes Received)
WS : 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1
PPL: 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
               ↑                         ↑
Received pattern: 1 0 1 1 0 1  (Four Bytes Received)
WS : 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 1
PPL: 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
               ↑                   Frame Synchronization Complete
```

Fig. 5. Pattern matching on a 32-byte *m*-sequence.

As shown in Fig. 5, this procedure is based on *selective elimination* of unlikely starting points for the received sequence. Let us assume that at the beginning of signal reception the first two received PPM pulses are `1 0 1`. In this case there are four equally likely possible starting-point scenarios. However, after reception of the next byte, and measurement of its location relative to the initial received sub-sequence, two starting-point positions can be eliminated. In this example, following reception of the fourth byte, the starting position can be uniquely determined and the symbol boundaries can readily be calculated using the detected offset.

The algorithm was simulated for the additive white Gaussian noise channel, as well as a noise-less channel. The channel input consists of the proper 128-byte *m*-sequence, transmitted by the spacecraft for establishing symbol synchronization.

The receiver aims to establish acquisition of the *m*-sequence in as short a time as possible. The pulse detection threshold is set at a level such that a specified bit error probability is achieved. The receiver is switched on at a random time location during transmission of the acquisition sequence by the spacecraft.

To simulate this initial random location within the *m*-sequence, a random number uniformly distributed between 1 and 128 is generated to indicate the starting location of the first received byte. Since a single received byte is insufficient to pinpoint a unique location within the sequence, the receiver waits for at least one more PPM pulse. The objective of the algorithm is to uniquely identify the location of the received subsequence, within the *m*-sequence template, at which point the receiver was switched on.

Initially, the search algorithm proceeds to match the received 2-byte sequence against the *m*-sequence template. This is done by identifying the relative location of the second received byte with respect to the first received byte, and marking all possible sub-sequences that match this exact pattern.

At this point however, one has multiple candidate solutions for the initial startup time/location. Additional bytes need to be received in order to uniquely determine the location at which point the receiver was switched on.

Thus the process continues with additional reception of bytes, with the algorithm matching 3-byte, 4-byte, . . ., sub-sequences, until it arrives at a single unique sub-sequence. Once the location of the initial received byte is determined, one can readily determine the symbol boundary.

1,000 runs of the algorithm with different random starting points were simulated for the case of a 128-byte acquisition sequence under noise-less conditions. Results are summarized in Fig. 6. As can be seen, at most six bytes are required to uniquely identify the frame boundaries. This is in contrast to traditional correlation based methods which require the full 128 bytes of the *m*-sequence to establish synchronization and determine the symbol boundary.

When real data is interleaved between periodic transmissions of the acquisition sequence, there could be situations whereby a certain received data bit pattern will be identical to a subset of the *m*-sequence. Additionally under noisy conditions (random bit flips), a certain bit pattern can possibly alias as the correct unique pattern, and results in faulty computation of the symbol boundary.

To address this problem one can introduce the concept of “confirmation”. This is defined as the number of additional received bytes required to achieve a certain probability of error prior to computing the symbol boundaries. The probability of false boundary detec-

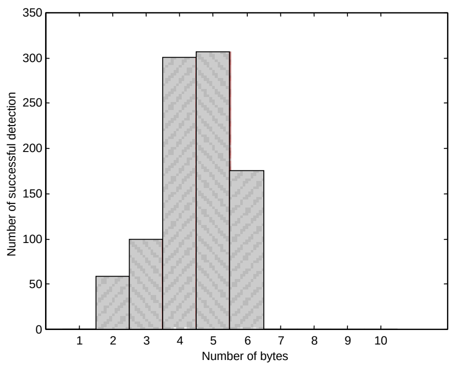


Fig. 6. Distribution of the number of bytes needed to uniquely identify a random starting location within a 128 byte *m*-sequence under noiseless conditions.

tion will be inversely proportional to the length of the confirmation sequence.

The procedure for confirmation is as follows. Let us assume successful identification of a unique sub-sequence, and determination of the starting location. The detection period is now complete and the confirmation period starts. The algorithm can now "predict" the time of arrival of the next PPM pulse. If the next detected pulse is at the expected location, confirmation continues with the subsequent pulse, for as many pulses as required.

If all the confirmation information matches the expected values, boundary detection is successful and the frame boundaries are calculated. But if a mismatched byte of information is received the algorithm resets, and starts a new search with the next pulse location.

Conflicting situations are defined to arise in one of two ways: (1) when a mismatched pulse is detected *during the confirmation period*; and (2) when due to random bit flips, no unique sub-sequence can be identified *during the detection period*. In both cases the algorithm resets as explained previously.

More advanced confirmation methods based on partial correlation of the received sequence and the expected template can be used to deal more efficiently with conflicting data cases. However in this study only the exact confirmation sequence matching is explored and all the results are generated using this confirmation method. Use of confirmation creates the flexibility of adjusting and reducing the probability of false boundary detection to an acceptable level.

To test performance of the algorithm under noisy conditions, sixty 50,000 runs with different received bit error probabilities and different required confirmation bytes were simulated.

Figures 7-11 summarize the results: *confirm 0* means that no confirmation pulses were used; *confirm 1* means that a single pulse was used for confirmation after any unique detection, etc.

The results provide a means for selecting the proper confirmation period. For instance, with bit error probability of 0.007, which is the nominal design point for this receiver, and no confirmation period, the probability of false detection is about 1.5%, which is very high. However, for the same bit error probability and five confirmation pulses the error reduces to about 0.02%.

Figures 8 and 9 can be used as receiver design tools to select the number of required confirmation bytes to achieve a desired probability of incorrect boundary computation. Once this design parameter is selected, the remaining probabilities can be calculated for a give channel bit error rate using Figs. 7,10, and 11.

As the figures show, increasing the confirmation count will reduce the probability of false detection but will increase the probability of encountering a noisy byte which results in conflicting data status and resetting of the search algorithm. This is the reason for the observed pattern of lines in Fig. 8.

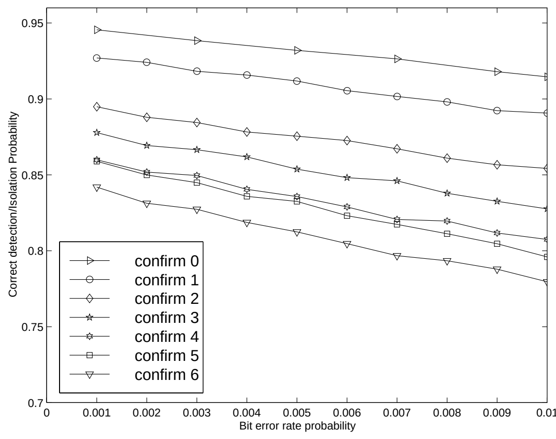


Fig. 7. Probability of correct isolation/detection as a function of bit error probability and confirmation period.

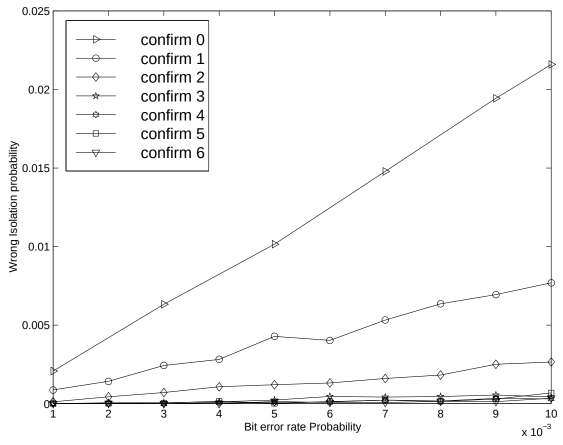


Fig. 8. Probability of wrong isolation/detection as a function of bit error probability and confirmation period.

Figure 11 shows the probability of reaching the end of the synchronization sequence without successful iso-

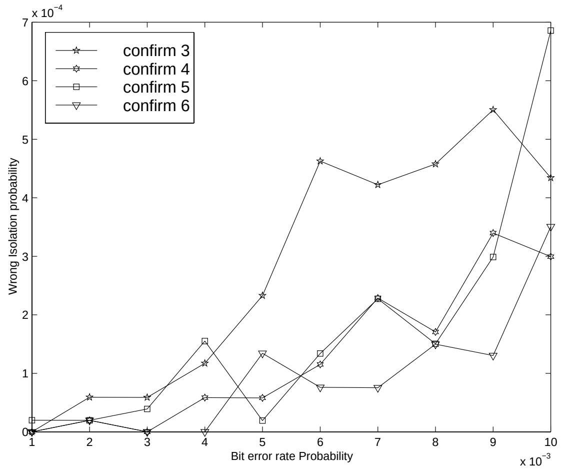


Fig. 9. Expanded view of Fig. 8 for 3-6 bytes of confirmation.

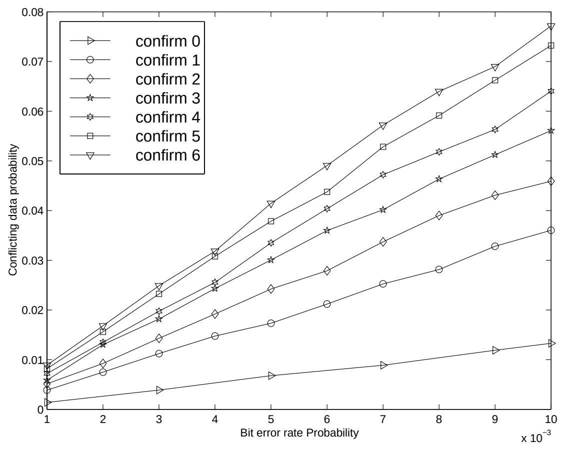


Fig. 10. Probability of conflicting data detection as a function of bit error probability and confirmation period.

lation of the received sequence. This probability increases as the confirmation length is increased. The longer the length of the required confirmation sequence, the higher the probability of reaching the end of the synchronization word without any isolation.

Notice that all the probabilities presented in this paper are related to the *first* received synchronization sequence. [1] This means that the value of 0.6×10^{-4} for no isolation with six confirmation pulses (Fig. 11), is only the probability of missing the first received synchronization sequence and not the probability of total synchronization failure. This is the probability of reaching the end of the *m*-sequence without isolating the boundaries.

REFERENCES

[1] R. Gagliardi and S. Karp, *Optical Communications*. New York: Wiley, 1976.

[1]The synchronization sequence is interleaved and re-transmitted periodically with the data frames.

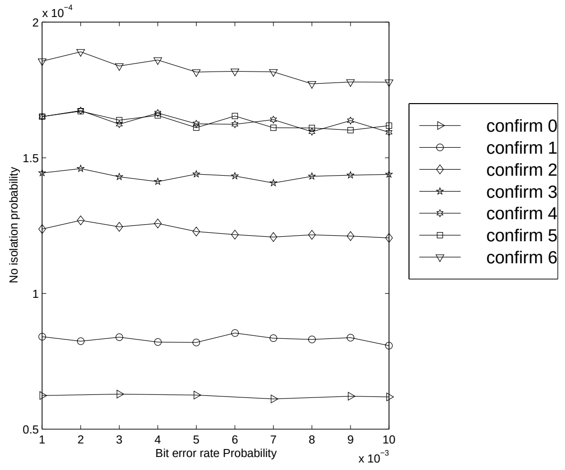


Fig. 11. Probability of failing to isolate as a function of bit error probability and confirmation period.

[2] Ger Ling and R. Gagliardi, "Slot Synchronization in Optical PPM Communications", *IEEE. Trans. Communications*, vol. 34, no. 12, pp. 1202–1208, Dec. 1986.

[3] R. Gagliardi. J. Robbins, and H. Taylor, "Acquisition Sequences in PPM Communications", *IEEE Trans. Information Theory*, vol. 33, pp. 734–738, Sept. 1987.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.