Suppressing Error Floors in SCPPM via an Efficient CRC-aided List Viterbi Decoding Algorithm

Amaael Antonini*, Wenhui Sui,*, Brendan Towell,*, Dariush Divsalar, †, Jon Hamkins, †, and Richard D. Wesel*
*Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA
†Jet Propulsion Laboratory, California Instute of Technology, Pasadena, CA, 91109 USA
Email: {amaael, wenhui.sui, brendan.towell, wesel}@ucla.edu, Dariush.Divsalar@jpl.nasa.gov, Jon.Hamkins@jpl.nasa.gov

***Abstract*—List Viterbi decoders are a very effective way to improve the performance of block codes in combination with an error detection outer code. In this work, we combine an efficient serial list Viterbi decoder design with an existing serially-concatenated, convolutionally-encoded, pulse position modulated code (SCPPM) used in space communication, that exhibits poor performance because of an error floor. The SCPPM code features a 32-bit CRC that provides powerful error detection capability and an outer four-state convolutional code that makes it suitable for a list Viterbi decoder. The system's code is very long, consisting of $15,120$ bits, which renders a high complexity decoder impractical, while the high error detection allows for a list decoder with very low undetected error probability. We use a very efficient list Viterbi decoder algorithm to avoid most of the redundant operations to produce low complexity serial list Viterbi decoder. The combined system reduces the error floor, moderately for the original version of the system, and completely suppresses it when the code length is increased to four times longer.**

***Index Terms*—Channel codes, list decoding, space Comms.**

I. Introduction

In this work, we attempt to use a serial list Viterbi decoder (SLVD) to improve the performance of an existing space communication system by suppressing and reducing an error floor. The system consists of a serially concatenated convolutional code with pulse position modulation (SCPPM) over a free-space optical channel.

The Viterbi decoder was introduced by G. D. Forney in [1] using the Viterbi algorithm developed by A. Viterbi [2]. List decoding was introduced earlier by P. Elias [3] and by J. M. Wozencraft [4]. P. Elias [5] later developed an (n, e, L) list decoder that corrects all sets of e or fewer errors per block of n binary symbols. F. Soong and E. Huang proposed a fast trellis search method to obtain a list of the best L hypotheses in speech recognition. The best overall hypothesis was found in a forward pass and the other $L-1$ in a backward pass. A list Viterbi decoder was later developed by N. Seshadri and C. Sundberg [6] to find the best L decoding estimates. They proposed a parallel list Viterbi decoder (PLVD), that produces the L best paths at the same time, and a (SLVD) that finds the $k+1$-th best candidate only after the k-th best candidate is found. Seshadri and Sundberg combined the LVD with an error-detecting code to obtain significant improvement in the block error probability. To reduce the complexity of the LVD, M. Roder and R. Hamzaoui [7] pioneered a LVD that used Soong and Huang's trellis-search method but replaced a sorted list with multiple unsorted lists collecting candidates of the same metric. Lou *et al.* [8] proposed using a error-detecting cyclic redundancy check (CRC) code designed for specific convolutional code (CC) in a concatenated code. Their design attained a significant reduction in the undetected error probability compared to other CRC choices of the same degree. Yang *et al.* [9] studied an optimal joint design of a CRC and CC pair to achieve a target frame error rate. Yang *et al.* used the signal-to-noise ratio (SNR) gap between the joint design and the random coding union (RCU) bound as optimization criterion and proposed that the optimal CRC and CC pair is the one that achieves a target SNR gap with the least complexity. Yang *et al.* [10] also analyzed the trade-off between undetected error probability and maximum list size of a SLVD decoder. After Lou *et al.* [8], Yang *et al.* further identified the CRC polynomial that maximizes the minimum distance between codewords of the concatenated code where the CC used the polynomials $(561, 753)$ octal in the 3GPP standard [11]. Other recent works on list decoding of CCs include Sui *et al.* [12] and Wang *et al.* [13].

Specific to space communication systems, Schiavone *et al.* [14] studied the distance spectrum of the joint CRC and convolutional code of the code options provided in the consultative committee for space data systems (CCSDS) telemetry recommendation [15]. Schiavone *et al.* estimated that the coding gain of maximum likelihood decoding of the joint code is 3 dB higher than with Viterbi decoding.

In this work we attempt to use a list decoder to suppress or reduce the error floor on an existing SCPPM system, from the consultative committee for space data system (CCSDS) recommended standard [16]. To maintain backward compatibility, modifications to the SCPPM standard should be avoided, but few with low implementation impact could be considered. The SCPPM system, however, exhibits particular properties that make a list Viterbi decoder suitable to aid in the decoding process. The most important of these properties are an outer convolutional code where a list Viterbi decoder could be applied; a powerful error-detecting, 32-bit CRC; and an error floor region where many codeword error (CWER) events

This work was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the NASA, and National Science Foundation (NSF) grant CCF-2008918 at UCLA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect views of NSF.

have very sparse bit errors after the execution of the existing decoder. The information sequence in this system is very long, either $5,006$, $7,526$ or $10,046$ depending on the mode the system is operating in. The length of the information sequence demands a complexity-conscious implementation of the list decoder.

We implement a list Viterbi decoder with a new algorithm that significantly lowers the computational complexity when the number of states in the convolutional code is small compared to the length of the code. Simulation results show that our list decoder does reduce the error floor, very moderately in the original version of the system and completely when some modifications are included. The results also show that the new algorithm executes very fast when the maximum list size is set below 2000, which is the region where most of the performance gain is attained.

A. Contributions

The main contributions of this work are as follows:

- We demonstrate how a list Viterbi decoder can be used to eliminate or significantly reduce the error floor in the SCPPM system when the code length is allowed to increase to four times the original length.
- We propose a method to significantly reduce complexity for a serial list Viterbi decoder when the number of trellis states is small compared to the number of trellis stages. The reduced complexity is especially significant for the first few hundred candidate codewords, which is where most of the list decoding improvement is obtained.
- We further propose a method to reduce the memory and run-time complexities required to search for candidate codewords. This method stores the message bits and metrics for the detours of all previous candidate codewords. This allows a low complexity differential approach to computing the metric of subsequent candidate codewords while using only a constant amount of memory equal to the number of edges in the trellis since each trellis edge is only traversed once.

II. SCPPM STANDARD

The encoder and decoder of the SCPPM system recommended by the CCSDS in [16] is shown in Fig 1. For the purpose of this work, we consider the information sequence from the "source" a randomized slice of a CCSDS Transfer Frame. We also consider the processing that starts with the channel interleaver up to the guard slot insertion, shown in Fig. 3-1 of the CCSDS standard [16] to be part of the channel. We proceed with a high-level description of the encoder and decoder and direct interested readers to the standard [16] for a detailed description.

The SCPPM encoder is depicted on the left side of Fig. 1. The outer code is a zero-terminated convolutional code (ZTCC) with 2 memory elements and its encoder polynomials are $\{05, 07, 07\}$ in octal. The inner code is an accumulator with a single memory element. Between the two codes is quadratic interleaver with polynomial $11s + 210s^2$.

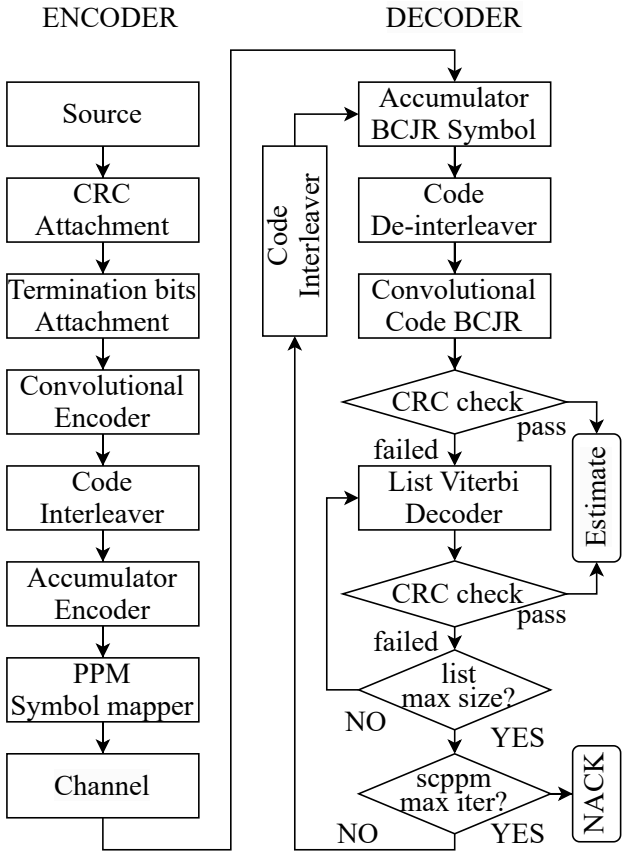

Fig. 1. SCPPM Encoder (left side) from [16] Fig 3-1 and Iterative Decoder coupled with a list Viterbi decoder.

The output of the accumulator is a 15120-bit sequence that is mapped to $M-$ary PPM symbols $C_1, C_2, \ldots$ with $M \in \{4, 8, 32, 64, 128, 256\}$ each encoding m bits, where $m \in \{2, 3, \ldots, 8\}$. The encoder's output is a vector of symbols $\mathbf{C}_1, \ldots, \mathbf{C}_{n/m}$, where symbol $\mathbf{C}_i$ is a vector $\mathbf{C}_i = \{C_{i,1}, C_{i,2}, \ldots, C_{i,M}\}$, where $C_{i,j} \in \{0, 1\}$ and $\mathbf{C}_i$ has a single 1 entry. There are three operating rates, $1/3$, $1/2$, and $2/3$, with the latter two obtained by puncturing the output of the outer code with the patterns $\{1, 1, 0, 1, 1, 0\}$ and $\{1, 1, 0, 0, 1, 0\}$. The outer encoder receives an input sequence of length 5040, 7560, or 10080 bits depending on the rate. This sequence includes the information sequence, a 32-bit error-detecting CRC, and two terminating zero bits.

The channel is assumed to be a memoryless Poisson channel defined by a background noise rate of K_b average photons per non-pulsed PPM slot and $K_s + K_b$ average photons per pulsed PPM slot. The received signal for each transmitted symbol i is vector $Y_{i,j}$ and for each PPM slot $j \in \{0, \ldots, M-1\}$ $Y_{i,j}$ is modeled by:

$$\Pr(Y_{i,j} = y \mid C_{i,j} = 0) = \frac{K_b^y e^{-K_b}}{y!} \tag{1}$$

$$\Pr(Y_{i,j} = y \mid C_{i,j} = 1) = \frac{(K_b + K_s)^y e^{-(K_b+K_s)}}{y!}\,. \tag{2}$$

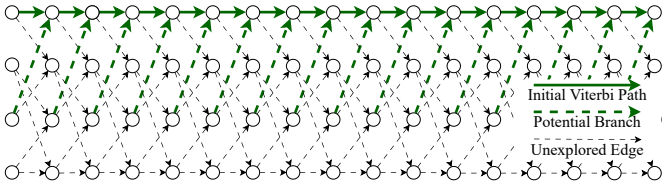

Fig. 2. Trellis segment showing the "initial Viterbi path," solid green arrows and the potential branches diverging from it, dashed green arrows. The "initial Viterbi path" is used to construct the most likely overall estimate, while its branches are used to construct subsequent paths.

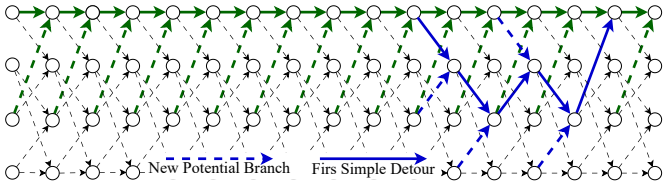

Fig. 3. Trellis segment showing the first detour from the "initial Viterbi path," solid blue arrows, and the potential branches that diverge from it, dashed blue arrows. The "first simple detour" is taken by the second most likely path, while new and existing "potential branches" are taken by subsequent paths.

The SCPPM code is decoded using an iterative decoder (ID) that is implemented using standard methods, i.e.. [17]. The decoder, depicted in the right side of Fig. 1, takes as input the signals received at each PPM slot and outputs an estimate of the information bits. The inner decoder (accumulator) takes as input the received signal and the a priori log-likelihood ratio (LLR) estimate of the coded bits from the outer decoder (convolutional code). The output of the inner decoder are estimated log-likelihood ratios (LLRs) of the coded bits (convolutional code) based on the structure of the inner code. The outer decoder takes as input the output of the inner decoder and produces its own LLR estimates of its coded bits, which are used to obtain an estimate of the information sequence. The estimated information sequence is then checked with the 32-bit CRC. If the CRC check fails the interleaved LLRs estimates of the outer code are used as the a priori input by the inner decoder for a subsequent attempt. The process repeats until the CRC check succeeds or the maximum number of iterations is reached, in which case an erasure is produced. We refer interested readers to the literature for a detailed description of an iterative decoder. One example of iterative decoding of serially concatenated CC can be found in chapter 7.3.2 of [17]. On the decoder side, the input signal is scaled and quantized to integer values. Also, look-up tables (LUTs) are used to speed up many of the operations and all the intermediate data is proceeded in integer values for efficient processing.

The proposed list Viterbi decoder (LVD) is included in Fig. 1 as a possible architecture to combine the standard SCPPM decoder. The LVD is not required to be executed after every iteration of the SCCC.

A. Problem Description

The existing SCPPM system sometimes exhibits an error floor that appears when the codeword error rate (CWER) falls between 10^{-4} and 10^{-6}, depending on the rate, modulation, and background noise rate K_b. The error floor is more significant for the rate-$2/3$ code. Modifications that have been investigated (but not exhaustively) before the current work include different interleaver and puncture pattern designs, termination to the inner code, and running the iterative decoder for more iterations. These yielded little success except that increasing the code (and interleaver) length improved the error floor in some cases. The objective of this work is to use a list Viterbi decoder (LVD) to suppress or reduce the error floor. Solutions involving the SCPPM decoder are preferred over changes to the encoder, since the code itself is an approved international standard.

III. List Decoder with Efficient Detour Management

We propose a serial list Viterbi decoder (SLVD) that significantly reduces the run-time complexity when the number trellis states is much smaller than the number of trellis stages. The key to the complexity reduction is that our algorithm does not trace back trellis detours that branch from a state already visited by previous paths. Instead, we store enough information to reconstruct these detours when they become relevant. Our approach is related to the method proposed by [6], where only the portion of a sequence that has not been discovered yet is stored. We remark that storing this sequence only requires an amount of memory equal to the number of trellis edges. This method allows for a very fast trellis search while using very little extra memory compared to a full trellis trace-back.

A. Serial List Viterbi Overview

The SLVD we propose, as other approaches do, first produces the most likely overall estimate, and then each next most likely one up to a maximum list size L. We initialize the list decoder with a forward and backward trellis search, similar to the one described in [18]. We keep a list of candidate paths called "detours", which are sorted in the order of highest likelihood. In the forward pass we store the edge and metric through which the most likely path reaches every trellis stage-state pair and separately the alternative edge and metric. In the backward pass we trace the most likely trellis path, the "initial Viterbi path" and make a list of the potential detours that branch from this "initial Viterbi path." Each potential detour

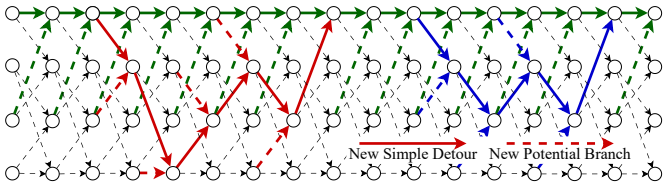

Fig. 4. Trellis segment showing the "initial Viterbi path" in Fig. 2, the first "detour" in Fig. 3 and a new "detour," solid red arrows, along with the potential branches diverging from it, dash red arrows. The blue and red "detours" also combine into a valid "compound path".

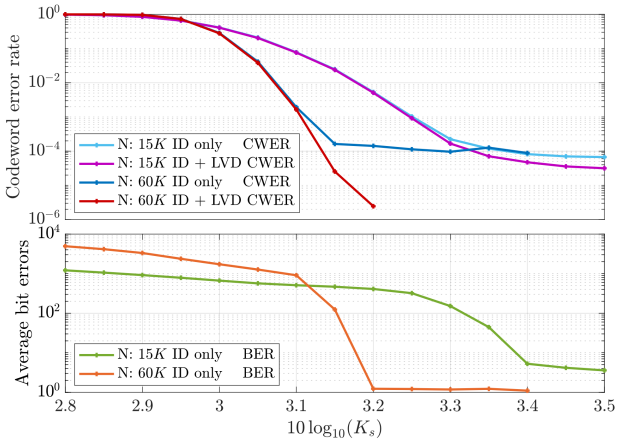

Fig. 5. CWER vs K_s(dB), i.e. $10\times\log_{10}(K_s)$, for iterative decoder (ID) with and without the list Viterbi Decoder (SLVD), top subplot. Average number of message bit errors (only for frames in error under ID), bottom subplot. Two interleaver lengths are shown, $N = 60480$ and $N = 15120$, all with 16-PPM modulation, transmission rate $2/3$ and background noise rate K_b of 0.1 photos per slot.

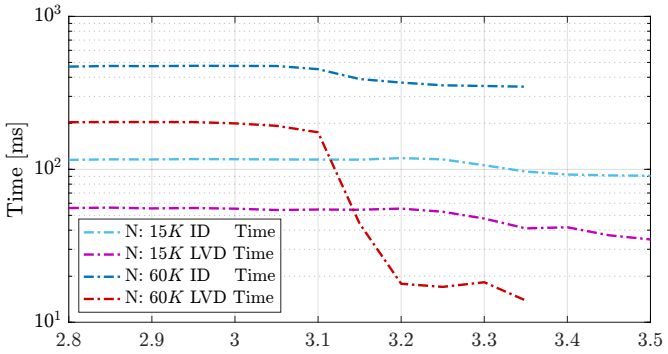

Fig. 6. Average decoding time vs K_s(dB), i.e. $10\times\log_{10}(K_s)$, for iterative decoder (ID) with and without the list Viterbi Decoder (SLVD). The code lenght, signal power and modulation are the same as Fig. 5.

is defined by the state-stage where it branches from the "initial Viterbi path" via the alternative source edge and state reaching it, and by the likelihood difference between the two edges. A segment of the "initial Viterbi path" and the potential branches is depicted in Fig. 2. The second overall most likely path will be one of the paths that branch from the "initial Viterbi path" and can be constructed by tracing back the most likely potential detour. We check each LVD estimate with the CRC and terminate the decoding if the check succeeds. Otherwise, the list decoder searches for the next most likely estimate. The process continues until a check succeeds, or the maximum list size L is reached, in which case an erasure is declared.

B. Successive Paths Construction and Candidate List Update

Each new list estimate is constructed from the next most likely candidate in the list of potential detours. The list of candidates is initialized in the backward path. Then, if the "initial Viterbi path" fails the CRC check, the next most likely candidate is selected from the list to construct the next most likely path. We start a trellis trace-back from the stage-state pair where the candidate branches from the "initial Viterbi path" through the alternative edge and source state. The trace-back when the new path reaches a state previously visited by the "initial Viterbi path". An example of a detour branching from the "initial Viterbi path" and merging back five stages later is depicted in Fig. 4 by the blue solid arrows. We construct the new estimate using the original estimate and replacing the segment branching point to the merging point with the new detour. For this we store the original estimate and the states visited by the "initial Viterbi path."

Once the second path is found, the next most likely one could either be a branch from the "initial Viterbi path," and be an item of the initial list of candidates, or it could branch out of a previous detour, as depicted by the "new potential branches" in Fig. 4. To account for the later case, the new potential branches from the new detour are added to the list of potential candidates. A path that branches from a previous path other than the "initial Viterbi path" will share a segment with that previous path. To avoid retracing any segment, we store the new segment of each path in a stack, and separately keep a list with the start and end stage-state pairs of the detour taken, as well as the source path it branches from, and a pointer to the address where the segment is stored in the stack. The amount of storage needed to store all possible segments is at most the number of edges in the trellis since each segment consists only of previously unused edges. Thus we can efficiently reconstruct a path using new segments and previously stored segments while keeping the storage requirement modest and fixed.

It is possible for a valid path to branch out and merge into the "initial Viterbi path," or any other path, more than once. These types of paths are commonly discovered by a trace-back that continues past the point where a new segment merges back with an existing path. However, before such a path can become the next most likely one, each of these segments will form paths of their own standalone. Thus we limit the trace-back to the segment through edges and states not encountered before. After a path is discovered, we check which of the previous paths could be combined with the new segment to form a valid path. To illustrate the complexity reduction attained, suppose that the trellis consists of four states and 5040 stages, and a new path branches from the "initial Viterbi path" at a stage around the middle of the trellis. This new path will likely merge back to the "initial Viterbi path" after a few stages, say around ten, which are the only ones we will trace back. A full trace-back would instead last for more than 2000 stages.

IV. Simulation Results

The CWER vs. K_s(dB) performance of the SCPPM decoder integrated with the new SLVD is shown in Fig 5 and. The code rate is $2/3$, with 16-PPM modulation and a background noise rate of $K_b = 0.1$ photons per slot. The figure shows CWER vs. K_s(dB) curves for the original code lengths: $N = 15120$ and a modified one with $N = 60480$, which changes the interleaver too. For each code we show the performance of the iterative

decoder (ID) with the SLVD integrated and also that of the ID standalone for comparison. The maximum iterations for the ID was set at 32 with a list decoding attempt every eight. The maximum list size of the SLVD was set at 2000.

The curves show a severe error floor for the standalone ID, both with the standard code and the modified code. For the original code with length $N = 15120$ a lower but severe error floor persisted after the SLVD was integrated with the ID. However, the error floor was suppressed or significantly reduced for the modified system with code length $N = 60480$ when the SLVD was integrated.

Most error events in the modified system were observed to contain less than three bits in error and the list decoder easily found the right codeword. For the original system, adding the SLVD was not sufficient to solve the error floor problem. Many error events in the original system contained four or more bit errors distributed along three or more separate segments in the code. These cases are difficult to correct with a LVD because the right codeword will not be found within the first thousand of most likely ones. The issue persisted even when the list size was allowed to grow up to $200K$. Suppressing the error floor would then require either modifying the system or making further changes to the decoder so that the SLVD can correct the remaining errors.

A. Running time performance of the new List Decoder

The running time results of our SLVD compared to the iterative decoder are shown in Fig. 6. The time results are averages, in milliseconds (ms) for only the cases where the iterative decoder failed to decode. We show the average time the ID took to fail and the average time SLVD took to either produce an estimate that passed the CRC check or reach the maximum list size and fail. Results for both the standard code of length $N = 15120$ and the modified code with $N = 60480$ are included. Both the ID and SLVD use the maximum number of iterations in the region where their respective codeword error rates for match, 32 iterations for the ID and a maximum list size of 2000 for the SLVD.

The running times for the SLVD with maximum list size of 2000 is lower than 32 iterations of the ID, even when $SLVD$ fails to decode and reaches the maximum list size, ans can be noted in the region where both ID exhibit the same CWER performance. When the modified code is used, and the SLVD succeeds most of the time, the SLVD is much faster on average than the ID. The running times results show that the SLVD does not add a large delay when it executes. Also, since the list decoder only needs to run when the ID fails, the overall running time added by the SLVD is very low.

V. Conclusion

This paper demonstrates how list decoding can utilize the existing CRC in an SCPPM to lower the error floor. While results show that the improvement is moderate when the standard code of length $N = 15120$ is used, dramatic improvement is seen when the code length is increased to $N = 60480$. We did not initially find an interleaver that allowed the list decoder to remove the error floor for the 15K blocklength, but this is an area of ongoing research.

SLVD is able find a codeword when the iterative decoding estimate has a small number of bit errors, but typically cannot find a codeword when iterative decoding retains four or more bit errors. This paper also presents an efficient SLVD implementation for long blocklengths. In the region of interest, where the SLVD succeeds often, our implementation takes, on average, the same or less time than iterative decoding. Since the SLVD is only needed in the rare cases where the iterative decoding fails, decoding time for the combined system is similar to the original system without list decoding. Thus, SLVD improves the performance of the SCPPM while adding minimal decoding time, making SLVD suitable for practical use.

References

[1] G. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[2] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[3] P. Elias, "List decoding for noisy channels,," *Institute of Radio Engineers (now IEEE) Wescon Convention Record part 2*, pp. 94–104, 1957.

[4] J. M. Wozencraft, "List decoding," *Research Laboratory of Electronics Quarterly progress report*, vol. 48, 1958.

[5] P. Elias, "Error-correcting codes for list decoding," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 5–12, 1991.

[6] N. Seshadri and C.-E. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Tran. Comm. Theory*, vol. 42, pp. 313–323, 1994.

[7] M. Roder and R. Hamzaoui, "Fast tree-trellis list Viterbi decoding," *IEEE Trans. Comm.*, vol. 54, no. 3, pp. 453–461, 2006.

[8] C.-Y. Lou, B. Daneshrad, and R. D. Wesel, "Convolutional-code-specific CRC code design," *IEEE Trans. Comm.*, vol. 63, pp. 3459–3470, 2015.

[9] H. Yang, E. Liang, and R. D. Wesel, "Joint design of convolutional code and CRC under serial list Viterbi decoding," 2018.

[10] H. Yang, S. V. S. Ranganathan, and R. D. Wesel, "Serial list Viterbi decoding with CRC: Managing errors, erasures, and complexity," in *2018 IEEE Global Comm. (GLOBECOM)*, 2018, pp. 1–6.

[11] E. T. S. Institute, *3GPP TS 25.212 version 7.7.0-7*. ETSI, 2008-01.

[12] W. Sui, H. Yang, B. Towell, A. Asmani, and R. D. Wesel, "High-rate conv. codes with CRC-aided list decoding for short blocklengths," 2022.

[13] L. Wang, D. Song, F. Areces, and R. D. Wesel, "Achieving short-blocklength RCU bound via CRC list decoding of TCM with probabilistic shaping," in *ICC 2022 - IEEE Intl. Conf. on Commun.*, 2022, pp. 2906–2911.

[14] R. Schiavone, R. Garello, and G. Liva, "Application of list Viterbi algorithms to improve the performance in space missions using convolutional codes," in *2022 9th International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC)*, 2022, pp. 1–8.

[15] CCSDS, *TM Synchronization and Channel Coding*. Recommended standard. Blue Book, April 2022, no. 131.0-B-4.

[16] ——, *Optical Communications Coding and Synchronization*. Recommended standard. Blue Book, August 2019, no. 142.0-B-1.

[17] W. E. Ryan, *Channel codes : classical and modern / William E. Ryan, Shu Lin.* Cambridge :: Cambridge University Press, 2009.

[18] F. Soong and E.-F. Huang, "A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition," in *[Proceedings] ICASSP 91: 1991 Intl. Conf. Acoust., Speech, Signal Process.*, 1991, pp. 705–708 vol.1.