End-to-End System Consideration of the Galileo Image Compression System

K. Cheung K. Tong M. Belongie
Mail Stop 238-343 Jet Propulsion Laboratory, California Institute of Technology,
Pasadena CA 91109

February 6, 1996

Abstract

In this article we describe the image compression scheme of the Galileo spacecraft and other downstream signal processing algorithms that enable efficient and robust transmission of data to Earth.

Introduction

We present a robust implementation of the Galileo image compression system. Due to the loss of the high gain antenna, the mission is replanned to incorporate advanced data compression, error correction, and other downlink communication functions to maximize data return. Most instrument data will be highly edited and compressed before downlink through a noisy and unpredictable channel. The data compression schemes share a common disadvantage, that channel errors in the compressed data tend to propagate in the reconstructed data. To ensure reliable communication we developed a Feedback Concatenated Decoder (FCD) for error-correction and error-detection, and a concealment strategy to prevent error propagation and realign the reconstructed data after decoder failure. We also developed a frame repair scheme that combats phase inversions and symbol insertion/deletion in the channel, and a frame merging scheme that prevent data loss due to station hand-over.

Image Compression

The Galileo image compression scheme is a block based lossy image compression algorithm which uses an 8x8 Integer Cosine Transform (ICT) [1]. The algorithm is similar in functionality to the baseline JPEG standard which converts a fixed length string into a variable length string. A major difference between the JPEG standard and the Galileo compression scheme is that the JPEG standard uses an 8x8 Discrete Cosine Transform (DCT). The Integer Cosine Transform (ICT) is an integer approximation of the DCT. ICT requires only integer arithmetic, making it much simpler to implement. Although the ICT has a much lower complexity, the rate-distortion performance is not sacrificed. This lower complexity of the ICT is necessary because the Galileo spacecraft uses late 1970's microprocessor technology.

Error Correction

The Galileo error-correction coding scheme uses a (255,k) variable redundancy Reed-Solomon (RS) code as the outer code, and a (14,1/4) convolutional code as the inner code. The RS codewords are interleaved to depth 8

in a frame. The redundancy profile of the RS codes is (94, 10, 30, 10, 60, 10, 30, 10). The staggered redundancy profile was designed to facilitate the novel feedback concatenated decoding strategy [2]. This strategy allows multiple passes of channel symbols through the decoder. During each pass, the decoder uses the decoding information from the RS outer code to facilitate the Viterbi decoding of the inner code in a progressively refined manner. The FCD is implemented in software on a multiprocessor workstation. The code is expected to operate at bit signal-to-noise ratio (SNR) of 0.65 dB at a bit error rate of 10^{-7}. The FCD code selection and performance analysis are discussed in detail in [3].

If the FCD cannot decode the data due to errors beyond its capability to correct, the FCD will invoke the "frame repair" utility and the "gap processing" utility. Gap processing involves using data from either side of good decoded data and attempts to fill in data that was missing (the gap) because of loss of synchronization or hardware glitches. The frame repair utility attempts to correct any symbol insertions, deletions or inversions due to transmission reception problem. The reprocessed data is then fed back to the FCD for another attempt at decoding.

Frame Repair

Sometimes a frame is received that has adequate SNR, but the frame markers at the beginning and end of it have too many or too few symbols in between them and/or have opposite phase. This indicates that some symbols were inserted or deleted and/or a phase inversion occurred, somewhere in the frame. (A phase inversion is an event where the symbols after it are detected with the opposite phase of the symbols before it. So, if the transmitted symbols were all +1, the received symbols, in the absence of noise, would be +1 up to the inversion, and -1 for all symbols after that, or at least until the next phase inversion.) Because a high redundancy Reed-Solomon codeword with too many errors is vastly more likely to not be decodable than to decode incorrectly, one can make guesses about where the errors are and safely assume that a guess is correct if it results in a decodable codeword. However, while guessing may be safe, it is not likely to be successful unless there is some reason for making a particular guess. The assumption in frame repair is that n bits were inserted or deleted, and/or a phase inversion occurred, at one particular but unknown location in the frame. The guessing involved is restricted to guessing this location. A guess is tested by creating a modified version of the received frame with the appropriate number of symbols inserted or deleted, and/or a phase inversion made, at the guessed location. The frame is then Viterbi decoded, the result is deinterleaved, and Reed-Solomon decoding is attempted on the resulting eight words. If the highest redundancy Reed-Solomon decoder succeeds, it is extremely likely that the guess was approximately correct. The reason the guess need not be exact is that the effect of an incorrect guess is that essentially all of the Reed-Solomon symbols in between the actual anomaly location and the guessed one are incorrect, but if the gap is small enough, the Reed-Solomon decoder can correct those additional errors. Once the location is approximately known, the guess can be refined by trying nearby locations and choosing the one that minimizes the number of errors the Reed-Solomon decoder corrects. Then the frame, modified to reflect the refined guess, is sent on to the FCD for redecoding. But how are the guesses made in the first place? One rather simple minded method is to guess each of the 65536 symbol locations in the frame, one at a time. This, however, would take an unreasonable amount of time, because Viterbi decoding is slow. Slightly less simple-minded is to do a

binary search, since you only need to get close enough to the correct location. Thus, you first guess the location halfway through the frame, then 1/4 of the way, then 3/4, then 1/8, and so on. This method, in fact, is used when everything fails, but it can still take a very long time unless you're lucky. Before resorting to this, some other educated guesses are made. If any of the following methods succeeds, meaning that the highest redundancy Reed-Solomon decoder succeeds, the guess is then refined and the modified frame is sent to the FCD, as explained above. If not, the next method is tried. First method: if a data rate change occured at some point during the frame, that location is guessed, since anomalies are known to be common when data rate changes occur. Second method: the end of the frame is tried as a guess, which means that the frame is Viterbi decoded as is. The branch metrics along the decoded path are saved, for use in the next step in case this method doesn't succeed. Third method: if insertions or deletions are known to have occurred, the branch metrics mentioned above are analyzed. Although the effect is subtle when SNR is low, the mean value of the metrics will be slightly higher after the location of the anomaly is reached than before it. This analysis results in an estimate of where this increase occurred, and this estimate is used as the next guess. Fourth method: if only a phase inversion is known to have occurred (i.e. the frame is the correct length, but the markers on either end have opposite phase) the transparency of the Galileo (14, 1/4) convolutional code means that the output of the Viterbi decoder, after a transient in the vicinity of the inversion, is the same as it would have been without an inversion, except with opposite sign. Thus, in this special case, one can make multiple guesses without having to Viterbi decode a whole frame each time. It is only necessary to invert all the Viterbi decoded symbols (from the original frame) after the guessed location and then attempt Reed-Solomon decoding. A binary search is used to make guesses throughout the frame, each of which can quickly be tested. Fifth method: the last resort is the binary search described previously. If this fails, frame repair is not accomplished.

Frame Merging

Because of the large frame size (65536 symbols) and low bit rates that Galileo will be using, a single frame will take close to half an hour to transmit at the lowest bit rate. The lowest bit rate is used near the beginning and end of a pass, when the SNR is lowest, and it is not unlikely that a frame that starts out with adequate SNR may not be completed before the signal is lost, or becomes noisy enough to render the whole frame undecodable. Because the ground stations have overlapping coverage, the next station may pick up the signal in the middle of the same frame. Thus, each of two stations may have an incomplete and/or too noisy version of the same frame. Because of the nature of the redecoding algorithm, a partial frame by itself is essentially useless. Since the best part of each frame is the worst part of the other, it seems obvious to try to paste them together somehow. If the SNR at the crossover point, where the falling SNR at the first station equals the rising SNR at the second, is high enough, one could make a good frame by just using the symbols from the first station up to that point and from the second station after that point. However, even if the SNR at the crossover point is significantly below the decoding threshold, by adding the symbols from the two frames together, 3 dB could theoretically be gained in the vicinity of the crossover point, assuming the noise at the two stations is independent, a reasonable assumption. Farther from the crossover point, one could still combine symbols from the two

frames, but with weights reflecting the relative SNR's. Note that this requires estimates of the SNR throughout both frames, and that the results can be expected to depend on the accuracy of these estimates. An additional problem arises, however: low SNR not only makes symbols noisy, it can also cause symbol insertions and deletions. One way this could be detected is if the frame markers both before and after a frame were detected and the number of symbols in between was greater or less than 65536. Since frame markers can be detected at SNR's way below the threshold of decodability, this is not unlikely. Notice, however, that this does not give any information about where in the frame symbols were inserted or deleted. To try to find a rough approximate location of any insertions or deletions, dynamic programming is used. Basically, the idea is to try to maximize the correlation between the two frames, by deleting or inserting various symbols. This is possible whether or not the existence of insertions or deletions in one or both frames was detected via counting symbols between frame markers; the latter just reduces the number of possibilities that must be considered. The result of this algorithm is a small number of candidate matched frames, that must then be sent to the FCD to attempt decoding.

Error Containment

If all these attempts fail to correct the data then the final step is to minimize error propagation. Because any corruption of data in a compressed data stream can cause magnified ("1"1"01"s due to propagation, safeguards were inserted into the compressed data stream to minimize these effects.

First, images are compressed independently by rows of 8, with a synchronization 11111111 inserted at the beginning, so any bit errors during data transmission can be isolated.

Second, the decompression algorithm includes provisions for detection of errors so that error containment mode will be activated.

These two safeguards are used together in the following manner: if an error is detected during decompression, the decompression algorithm will start a search for the synchronization marker. This marker is 32-bits wide which consists of 2 fields. A 7-bit sequencing field and a 25-bit fixed pattern field. The 7-bit field will identify where the 8-rows of data belongs relative to the whole image and the 25-bit field is used for synchronization. Using these safeguards, error propagation can at most be isolated to 16-rows of image data.

1 References

[1] W. Chen, "Development of Integer Cosine Transform by the Principle of Dyadic Symmetry," *IEE Proceedings*, Vol. 136, Pt. 1, No. 4, August 1989.

[2] E. Paaske, "Improved Decoding for a Concatenated Coding System Recommended by CCSDS," *IEEE Transaction of Communications*, Vol. COM-38, August 1990.

[3] S. Dolinar and M. Belongie, "Enhanced Decoding for the Galileo Low-Gain Antenna Mission," *Proceedings 1994 IEEE Symposium on Information Theory*, Trondheim, Norway, June 27 - July 1, 1994.