Decoding Convolutionally Encoded Images

G. H. Pitt, III, and L. Swanson
Communications Systems Research Section

Maximum Likelihood Convolutional Decoding, which is used by the Deep Space Network for short constraint-length convolutional codes, assumes that all strings of information bits are equally likely. In some cases, like image data, this is not the case. We examine the use of information about an adjacent pixel in decoding convolutionally encoded Voyager images, and discover that, in a region of interest, as much as 2 dB may be gained.

From the standpoint of digital data, the coding system for Voyager (and an international coding standard) consists of a Reed-Solomon encoder (optional), a convolutional encoder, a noisy channel, a maximum likelihood convolutional decoder (Viterbi decoder), and, if necessary, a Reed-Solomon decoder (see Fig. 1). With bit signal-to-noise ratio (SNR) as low as 2.3 dB, the concatenated scheme produces a bit error rate of 10^{-5}, and the convolutionally coded only scheme produces a bit error rate of 3×10^{-3}, while an uncoded scheme produces a bit error rate of 3×10^{-2} at this signal-to-noise ratio.

The Viterbi decoder finds the codeword which is closest to the received string. This is called maximum likelihood decoding because, under the assumption that all codewords are *a priori* equally likely to be transmitted, this decoding scheme retrieves the most likely sent codeword. In some cases, though, codewords are not all equally likely to be transmitted. In Voyager images, for example, pixel to pixel variations are not completely random: They are much more likely to be small than large. In this case, a decoder which makes use of the source statistics should perform better than a Viterbi decoder. (Image compression uses these statistics to lower the transmission rate and thus raise symbol SNR, but some Voyager images are sent uncompressed because of spacecraft limitations; also, an alternative would be valuable in the unlikely event of a data compressor failure before Neptune encounter in 1989.)

In Ref. 1, Korwar investigates the use of source statistics in convolutional decoding hard-quantized images, i.e., images in which each pixel is represented by one bit. Using hypothetical data, she found that using source statistics makes substantial improvement in decoder performance. Our work is different in two ways: We use 8-bit quantized pixels and real Voyager imaging data. Using 8-bit quantized pixels requires a byte-oriented rather than a bit-oriented decoder, which is more complicated and runs much more slowly.

We obtained a tape of image data from Voyager project. It consisted of several pictures, each 800 by 800 8-bit quantized pixels. A C-language program was written on a VAX running UNIX to extract the frequency of various absolute differences in value between adjacent pixels. That is, thinking of each pixel as an integer X between 0 and 255, the values of $|X_i - X_{i-1}|$ were tabulated. We were not surprised to discover that small values for this difference were much more common than would be expected in random (independent) data (see Fig. 3).

With the frequencies of $|X_i - X_{i-1}|$ obtained from one picture, a new decoder based on thse statistics was written, again using the UNIX VAX (see Appendix A for the algorithm). Another picture was then convolutionally encoded, Gaussian noise was added, and the picture was decoded with the new decoder. The performance of this decoder on this picture is shown in Figs. 4 and 5. These figures show that performance improves substantially for signal-to-noise ratios below 2.0 dB. This is a region of interest for images which have been Reed-Solomon encoded. (Byte error rates are shown in Fig. 5 because this is of interest for data which is Reed-Solomon encoded.)

The curves show that our decoder does not perform as well as a conventional Viterbi decoder for signal-to-noise ratios much above 2.0 dB. At least part of this can be attributed to the edges of pictures, which our tape shows as dark on one edge and white on the other. These strips may be the result of data compression, in which case they would not occur in images being sent uncompressed (the only kind of images for which the new decoder is useful). Our lower performance may also be partly a result of "rizzomarks" which are added for calibration and whose effects could be discounted in a scheme like ours.

We have not yet examined other possible schemes, such as considering both the pixel to the left of the current pixel (which the new decoder uses) and the pixel above it. Such a slightly more complicated scheme may slightly improve performance at the cost of slightly slower decoding. Using a software decoder on a VAX 750, decoding is very slow (640,000 bytes/month); if our scheme were to be used for real transmitted data, we would need big gains from hardware and algorithm improvements.

During Voyager's Uranus encounter, some undecoded channel symbols may be saved for the symbol stream combining experiment. We hope to use these to test our decoder under "real life" conditions.

Reference

1. Korwar, V., Viterbi Decoding Modified for Sources with Memory, *DSN Progress Report 42-55*, pp. 97–110, Jet Propulsion Laboratory, Pasadena, Calif., November and December 1979.

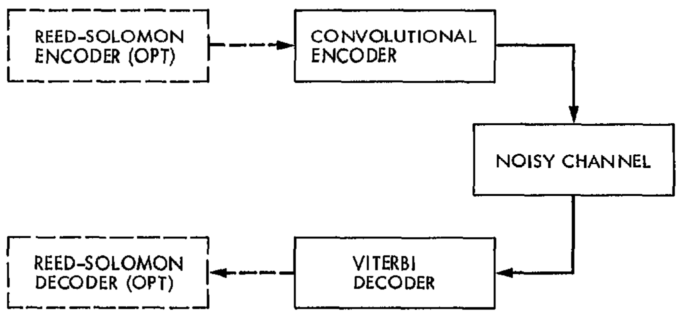

Fig. 1. Digital data coding scheme

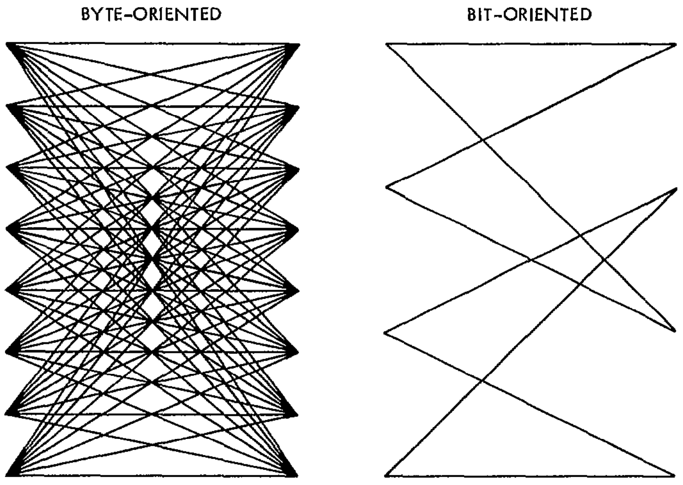

Fig. 2. An illustration of the difference in complexity of a byte-oriented decoder vs a bit-oriented decoder for 3-bit bytes and a *constraint length 3 code*

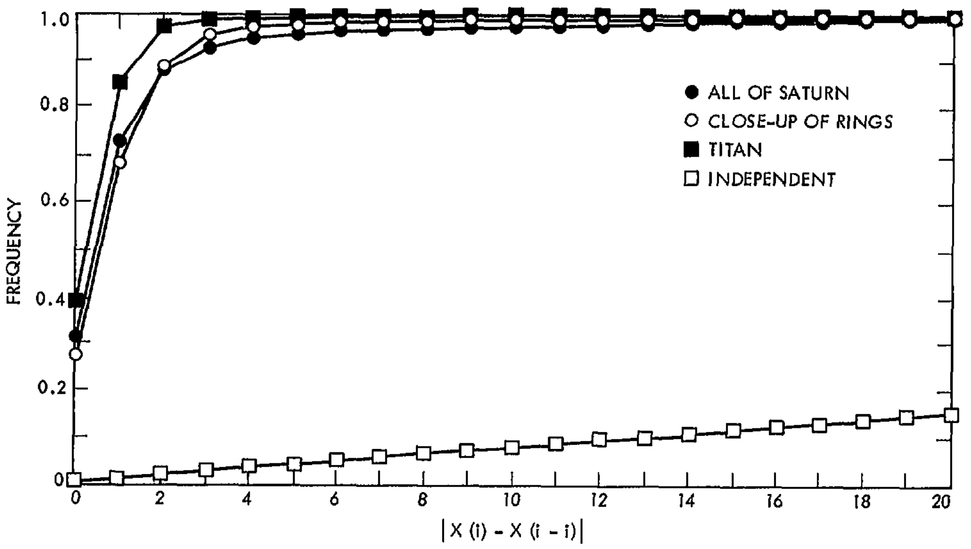

Fig. 3. A graph of the observed frequency of different values of $|X_i - X_{i-1}|$ for several Voyager images compared with the predicted frequency assuming independence (randomness) between adjacent pixels

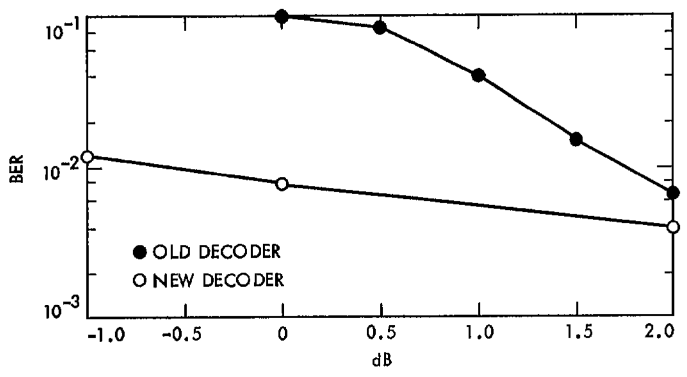

Fig. 4. A graph of bit error rates for the conventional (old) and the modified (new) maximum likelihood decoder

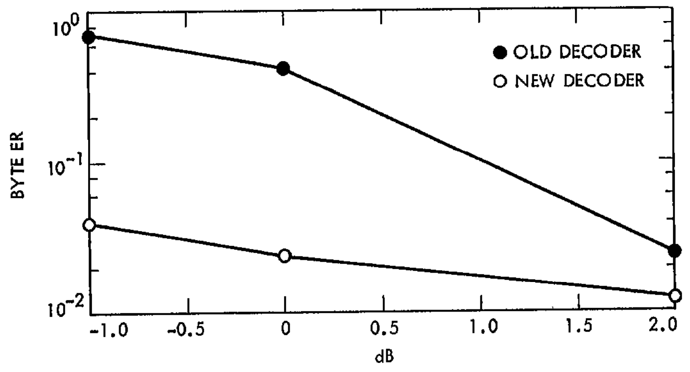

Fig. 5. A graph of byte error rates for the conventional (old) and the modified (new) maximum likelihood decoder

Appendix

We assume that symbols ±1 are transmitted over an independent Gaussian channel. That is, if $C = (c_1, c_2, \cdots, c_m)$ is transmitted, then $R = (r_1, r_2, \cdots, r_m)$ is received with probability density

$$P(R|C) = \frac{1}{(\sigma\sqrt{2\pi})^m} \prod_{i=1}^{m} \exp[-(r_i - c_i)^2/2\sigma^2] \tag{A-1}$$

where $\sigma^2 = 1/[2(E_s/N_0)]$ and E_s/N_0 is the symbol signal-to-noise ratio.

When a stream R is received, we wish to determine which was the most likely sent codeword C, or equivalently, which bit stream $X = (x_1, x_2, \cdots, x_k)$ was encoded to C, which was then transmitted ($2k = m$ in the case of our rate 1/2 convolutional code). We want to maximize $P(C|R)$ over all possible codewords C.

$$P(C|R) = \frac{P(R|C)P(C)}{P(R)} \tag{A-2}$$

so if all codewords are equally likely, we may equivalently maximize $P(R|C)$ over C. This in turn is the same as minimizing

$$\sum_{i=1}^{m} (r_i - c_i)^2 \tag{A-3}$$

since nothing else depends on C.

Current decoders use Viterbi's algorithm, which finds the string which minimizes Eq. (3) without searching every possible codeword.

If it is not the case that all codewords are equally likely, the analysis above is only valid through Eq. (2). Instead, when we maximize $P(C|R)$, we must maximize

$$\frac{1}{(\sigma\sqrt{2\pi})^m} \prod_{i=1}^{m} \exp[-(r_i - c_i)^2/2\sigma^2]\, P(C) \tag{A-4}$$

which is the same as minimizing

$$\sum_{i=1}^{m} (r_i - c_i)^2 - 2\sigma^2 \log(P(C)) \tag{A-5}$$

since nothing else depends on C.

As the signal-to-noise ratio decreases, σ^2 increases (for fixed signal energy), so the choice of model for the probabilities of various codewords ("source statistics") becomes more important.

We wish to add as little complexity as possible to the Viterbi algorithm, so we would like to assume that the source is Markov, meaning the probability distribution of x_i given x_1, $x_2, \cdots, x_{i-1}$ really only depends on $x_{i-r}, \cdots, x_{i-1}$ for some small r. Unfortunately, since each pixel is represented by one 8-bit byte, the natural statistics (of one bit to the next) are not even stationary, so we are forced to do all computations one byte at a time instead of one bit at a time. We have modeled the source statistics as byte-by-byte Markov; that is, if bytes are labeled X_i, with $X_i = x_{8(i-1)+1}, \cdots, x_{8i}$, we have modeled

$$P(X) = P(X_1, X_2, \cdots, X_\ell) = P(X_1) \prod_{i=2}^{\ell} P(X_i|X_{i-1}) \tag{A-6}$$

In addition, we have modeled $P(X_i|X_{i-1})$ as a function of the absolute differences in grey level (pixel value).

Each step of our convolutional decoder decodes one byte or pixel, and the process minimizes

$$\sum_{i=1}^{\ell} \left[\sum_{j=1}^{8} (r_{8(i-1)+j} - c_{8(i-1)+j})^2 - 2\sigma^2 \log(P(X_i|X_{i-1})) \right] \tag{A-7}$$

where we assume $P(X_i|X_{i-1})$ is actually distributed as modeled above.