Lossless Compression of NOAA-AVHRR Satellite Data

Seishi Takamura and Mikio Takagi
Institute of Industrial Science, University of Tokyo

Abstract

A high-performance lossless compression system for satellite NOAA data is developed. The data is called "high resolution picture transmission" (HRPT) data, and consists of around 93% advanced very high resolution radiometer (AVHRR) multi-channel image data and 7% of miscellaneous data. In compressing the image portion, we classify each pixel into 10 different groups and apply a multi-channel prediction and a non-linear error conversion. The entropy coder is an arithmetic coder which is adaptive and regenerates the approximation of the statistical properties of the source as an initial probability table. To compress the non-image part, we used the general compressor (gzip). From experimental results, the original information is compressed down to 25% $\sim$ 40%.

1 Introduction

The remotely sensed NOAA satellite "high resolution picture transmission" (HRPT) data provide very useful and important information in meteorology, oceanography and many other scientific fields. To date there have been many studies on image compression, particularly on lossy and very low bit rate compression. For image databases, a high compression ratio is important for storage and also for rapid transmission, but to deal with various kinds of users demands lossless image transmission is indispensable.

Also for this HRPT data, we must store them as they are. But one difficulty of this data is its size: one datum has more than 90MBytes, and we receive 5 $\sim$ 8 data each day. Reducing the size of the stored data is desired by both archiver and receiver.

The dominant part of this HRPT data is AVHRR (advanced very high resolution radiometer) image data. So we utilize the property of multi-channel 2-dimensional data of AVHRR data for compression. In the literature, some approaches of lossless compression (ex.[4, 7]) have been presented, but they are not very efficient in terms of compression ratio. For our database purposes, the compression ratio is more important than compression time, because decompression is a more common operation than compression.

In this paper we propose a method to losslessly compress the HRPT data which is somewhat computationally expensive but compresses much better.

<table>
<tr><td colspan="2">Header(ID,time,etc.)</td><td>1500 bytes</td><td rowspan="11">One line of HRPT data
22180 bytes</td></tr>
<tr><td>Channel 1</td><td>2 bytes</td><td rowspan="10">One line of
AVHRR data
20480 bytes</td></tr>
<tr><td>2</td><td>2 bytes</td></tr>
<tr><td>3</td><td>2 bytes</td></tr>
<tr><td>4</td><td>2 bytes</td></tr>
<tr><td>5</td><td>2 bytes</td></tr>
<tr><td>Channel 1</td><td>2 bytes</td></tr>
<tr><td>2</td><td>2 bytes</td></tr>
<tr><td colspan="2">⋮
repeat 2048 × 5 times</td></tr>
<tr><td colspan="2">Footer(synchronize)</td><td>200 bytes</td></tr>
</table>

Table 1: HRPT data format of one line

1.1 The Satellite NOAA and Its HRPT Data

The meteorological satellite NOAA-11 and NOAA-12 go around the earth at the average altitude of 810km in about 101.2 minutes. They have AVHRR sensor on board, which has five channels covering the wavelength from 0.55μm (channel 1, visible) to 12.5μm (channel 5, infra red). Reception of the observation data requires about 13 minutes when it is at its highest orbit and one transmission yields $3,000 \sim 4,000$ data lines. The word size for HRPT data is 10bits, but for simplicity of handling, our receiving system represents it by sixteen bits (two bytes). The 6 MSB's are padded with '0's. Therefore each HRPT data amounts to 90Mbytes. We receive $5 \sim 8$ HRPT data a day, so the total amount in a year exceeds 1TByte.

Each line of HRPT data is independent from all others, and contains 2048 pixels. The structure is shown in table 1.

Figure 1 shows an example of an AVHRR image obtained from NOAA-HRPT data.

2 Our Method

In this section our method will be explained. In short, our method for compressing AVHRR image portion is a kind of predictive coding such as DPCM. But we use many kinds of techniques to reduce its entropy and to code efficiently.

2.1 Noise-Line and Non-Imagery Part Treatment

Usually the AVHRR data contains noise lines(ex.figure 2). The number of noise lines in each datum is independent from all others. Such lines should be detected and removed from the image array, put together and compressed using a non-imagery compression method (gzip).

Figure 1: Example of AVHRR image (Channel 4, 2048 $\times$ 3736)

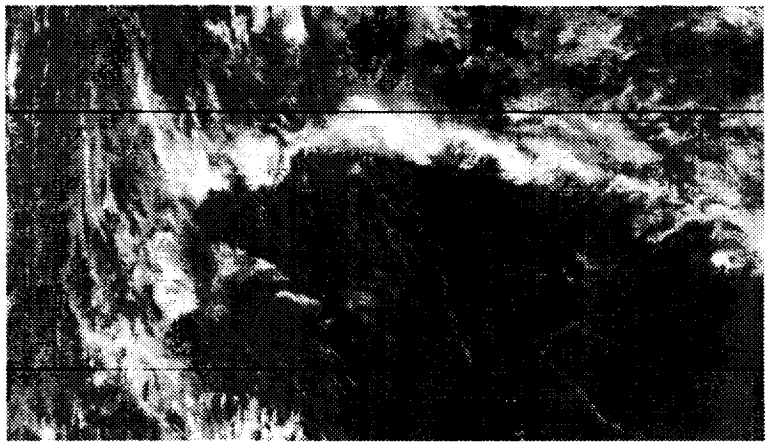

Figure 2: Example of noise lines in AVHRR data (A magnified part of channel 3, 544$\times$314). The two black horizontal lines are the noisy lines.

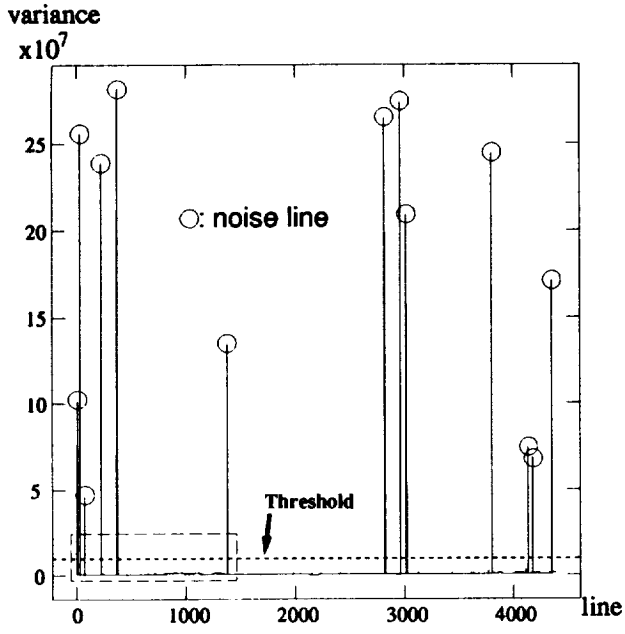

Figure 3: Calculated variance for each line. circles: detected noise lines, dotted line: threshold, rectangle with broken line: magnified area in figure 4

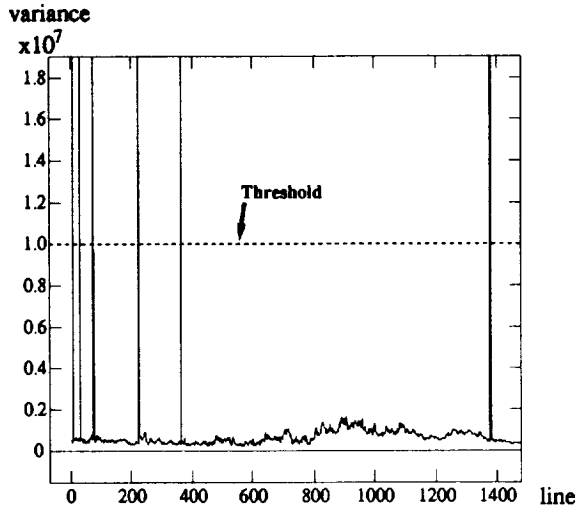

Figure 4: Magnified part of figure 3

To detect the noisy lines, we use a simple but reliable criteria. First we check the *ID* bytes in the header area. If the ID of a line is irregular, we regard the line as noisy. If a line passes this test, we calculate the variance of the difference between horizontally adjacent pixels in channel 1. If the variance is greater than a certain threshold (we use 1×10^7), we regard it as a noisy line. From experimental results, the first ID check is noise sensitive enough (see figure 3 and 4). We just use this second check to be doubly sure. The time for this check is significantly shorter than the total processing time.

The HRPT data contains about 7% of non-imagery data such as the fixed ID code, time stamp, fixed synchronizing code and so on (see table 1). The fixed or predictable portion is cut off. The remaining unpredictable portion and the noise lines are compressed separately from the image data. They are passed to gzip and compressed. Gzip is invoked with the '-9' option, which specifies the best compression.

This process is HRPT data dependent.

2.2 Pixel Classification

Each image pixel has different properties under certain criterions. From the point of image compression, grouping similar-propertied-pixels and encoding them respectively generates effective results. For grouping the pixels, we use the Q value:

$$Q \equiv |P_2 - P_1| + |P_3 - P_1| + |P_4 - P_1| + |P_5 - P_1| \quad (1)$$

Q	0–1	2–3	4–7	8–15	16–31	32–63	64–127	128–255	256–511	512–
Group #	0	1	2	3	4	5	6	7	8	9

Table 2: Grouping table

The position of the pixels ($P_1 ... P_5$) are shown in figure 5. This can also be calculated during decoding process, because only the upper or left pixels are used in equation 1. Using this Q value, we classify each pixel into several groups according to table 2.

2.3 Multi-Channel Prediction

For each classified group, we predict the value of the current pixel using linear combination of its neighbors' pixel values. The coefficients are calculated by the least square error method and use a constant to let the mean error be zero.

The neighbor pixels used for prediction are shown in figure 5 ($P_1 \ldots P_{10}$). For the pixels in the first channel, we use these 10 neighboring pixels.

The already decoded pixels are used to predict the pixels in the next channel ($R_1 \ldots R_9$ in figure 5). Using these pixels, something like interpolative prediction is achieved. This prediction contributes to the compression.

It is possible to use more than one of the previous channels if they are already decoded, but this increases processing time. We consider that looking for the optimal encoding order is more important. This is what we are investigating now.

2.4 Error Conversion

As each pixel has 10 bits, the prediction error $e(= \hat{x} - x)$ can have a real number value between -1024 and $+1024$ (roughly). After prediction, e should be expressed as an integer. An easy way to convert is to simply round the value off to an integer (calculate $\lfloor e + 0.5 \rfloor$) and consider it as a 2's complement 10-bit number.

Our conversion algorithm is somewhat different from simple rounding. After this

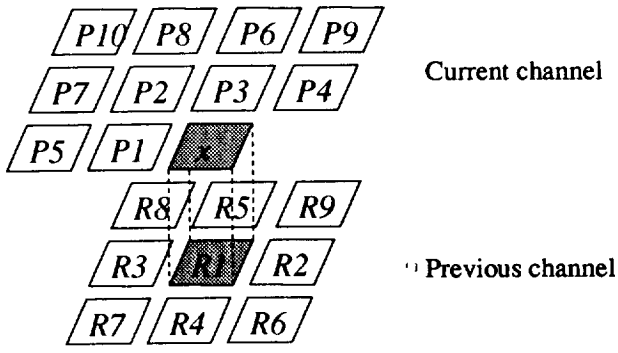

Figure 5: Pixels used for prediction

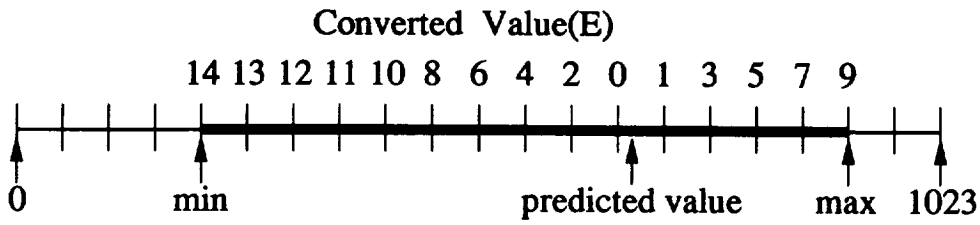

Figure 6: Algorithm for error conversion

conversion, we can also get the 10-bit non-negative integer E. First we obtain the upper and lower bound of the group (max, min). Then, as in figure 6, convert the prediction error into an positive integer. Within [min, max], the closest integer from the predicted value corresponds to $E = 0$, the second closest integer corresponds to $E = 1$, ... , and the n th closest integer corresponds to $E = n - 1$. (In figure 6, if the actual pixel value is equal to 'max', then $E = 9$.) For each group, we get the maximum and minimum pixel value and convert the prediction error respectively.

This conversion is reversible. If you get the predicted value and the converted number E (and also upper and lower bound), you can obtain the actual pixel value from the similar numerical rule.

2.5 Distribution Fitting and Entropy Coding

For natural images, the distribution of E is well approximated by the Gaussian distribution[8]. This distribution is used to generate the initial probability table for the encoder and decoder. The Gaussian distribution requires only one parameter – the variance – to be regenerated.

Figure 7 shows the graph of E vs. normalized distribution (probability) for each group (0...9). They do not exactly have the Gaussian shape, but for approximation and initial distribution generation, Gaussian curve fitting works well to reduce the code size. And it is clearly seen that from this figure, the curve of lower group (lower Q value) has more accurate peak (less variance) than that of upper group.

For the entropy coding, we adopt an arithmetic coder, because it has very effective performance and it is easy to make it adaptive.

3 Experimental Results

We programmed the compression program in C, on an HP9000/735. In this section the compression performance of our method and the time needed.

Table 3 lists the compression results made on 11 HRPT data obtained in December 1993. The column "stored size" means the whole file size of archived HRPT data in bytes, line-number × 22180 (the size of HRPT single line). The column "original size"

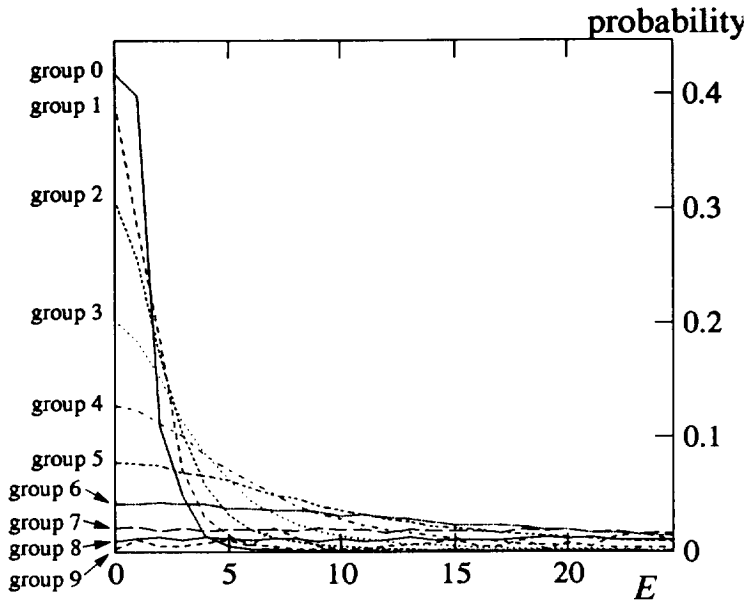

Figure 7: E vs. probability curve

means the actual amount of bits from NOAA satellite in bytes, "stored size" $\times \frac{10}{16}$. This difference is because we store one HRPT word (ten bits) by two bytes (see subsection 1.1). This value is used to calculate the column "compression ratio" (C.R.), i.e. "original size" divided by "compressed size".

In table 4 the comparison with 'gzip -9' is shown. Also the compression ratio is given by the ratio of original size to compressed size.

Date	lines	stored size	original size	C.R.	time(sec)
Dec.4, 15JST 1993	4400	97592000	60995000	3.219	4586
Dec.5, 15JST 1993	3390	75190200	46993875	3.506	3621
Dec.5, 16JST 1993	3023	67050140	41906337	3.458	3134
Dec.6, 15JST 1993	4400	97592000	60995000	3.219	4655
Dec.6, 16JST 1993	3607	80003260	50002037	3.245	3801
Dec.7, 14JST 1993	4289	95130020	59456262	3.277	4535
Dec.7, 16JST 1993	4023	89230140	55768837	3.309	4247
Dec.8, 14JST 1993	4358	96660440	60412775	3.338	4567
Dec.9, 14JST 1993	4087	90649660	56656037	3.194	4245
Dec.9, 16JST 1993	4400	97592000	60995000	3.189	4600
Dec.9, 17JST 1993	3053	67715540	42322212	4.713	3167

Table 3: Results of compression ratio (C.R.) and processing time

Date	lines	C.R.(gzip)	C.R.(proposed) and time(sec)
Apr.2, 15JST 1993	4400	1.204	3.018 (3303)
May 7, 14JST 1993	3187	1.188	3.066 (2381)
May 20, 15JST 1993	4400	1.137	2.730 (3297)
May 13, 20JST 1994	3267	1.686	4.280 (2522)

Table 4: Compression comparison with gzip and time

4 Conclusion

In this paper we proposed an effective method of lossless compression of NOAA HRPT images. Our method accomplishes the compression ratio of around 3 to 4. It actually means that in our receiving system the amount of HRPT data is reduced down to around one fifth. Though we don't use the same image as in other experiments found in the literature, the compression ratio by Kim's method[4] is around 2, and Tate's method for AVHRR data is around 2.7[7].

It is possible that the encoding order of channels has effect on compression ratio. Currently we encode five channels simply in channel order(1,2,3,...), and we only use the previous channel's pixels for prediction. Tate reports that for multi-spectral image, a well-chosen encoding order performs as well as optimal order[7]. He also reports that the effect of channel ordering makes slight difference especially for AVHRR data. We will seek for the optimal order and investigate if it is applicable to our data, and also examine using more channels for prediction.

It usually takes about one hour to compress a single HRPT datum. On the other hand, decompression is around six times faster. As mentioned in the introduction, the compression ratio matters more than the compression time, but this time might be considered too long. Therefore we are thinking of faster, more efficient and less redundant encoding algorithms.

References

[1] D. A. Huffman: "A method for the construction of minimum redundancy codes", Proceedings of IRE 40, pp. 411–420, 1951

[2] J. J. Rissanen et al.: "Arithmetic coding", IBM Journal of Research and Development, 23(2), pp. 188–193, 1976

[3] J. Ziv and A. Lempel: "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, IT-23(3), pp. 337–343, May 1997

[4] M. Kim, et al.: "NOAA Data Compression Using a Multi-Length DPCM Code and a Variable-Length Code", Proc. 11th Asian Conf. Remote Sensing, P-1, 1990

[5] P. G. Howard and J. S. Vitter: "New Methods for Lossless Image Compression Using Arithmetic Coding", Proc. Data Compression Conference '91, pp. 257–266, 1991

[6] T. Taniguchi et al.: "Variable-Length-Code-Selective Reversible Predictive Coding for Multi-Level Images", The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol.J70-B, pp.654–663, Jun. 1987

[7] S. R. Tate: "Band ordering in Lossless Compression of Multispectral Images", Proc. Data Compression Conference '94, pp. 311–320, 1994

[8] S. Takamura and M. Takagi: "Lossless Image Compression with Lossy Image using Adaptive Prediction and Arithmetic Coding", Proc. Data Compression Conference '94, pp. 166–174, 1994