The Skip-Innovation Model for Sparse Images

Paul J. Ausbeck Jr.
Caravian Software Designs
74 Carlyn Ave
Campbell, CA 95008
paula@alumni.cse.ucsc.edu

Abstract

On sparse images, contiguous runs of identical symbols often occur in the same coding context. This paper proposes a model for efficiently encoding such runs in a two-dimensional setting. Because it is model based, the method can be used with any coding scheme. An experimental coder using the model compresses the CCITT fax documents 2% better than JBIG and is more than three times as fast. A low complexity application of the model is shown to dramatically improve the compression performance of JPEG-LS on structured material.

1. Introduction

Scanned facsimile documents typically contain far more white pixels than black. Synthetic material such as charts, graphs, maps, figures, and clip-art tend to contain relatively large contiguous areas of uniform color. The more significant bitplanes of a wavelet decomposition may contain relatively more zeros than ones. All are examples of the class of *sparse images*.

An image need not be dominantly one color to be considered sparse. In this paper sparse simply means that pixel color changes occur significantly more infrequently than do pixels. An image does not need to be uniformly sparse to be considered sparse. For example, composite material may contain sparse and non-sparse regions.

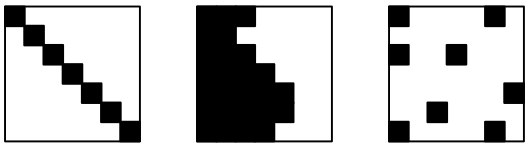

Figure 1
Toy Sparse Images

A related property that many sparse images exhibit is that of *raster structure*. A structured image contains features of significant extent whose boundaries can be tracked from scanline to scanline using a compact neighborhood template. For example, images (a) and (b) from Figure 1 both exhibit raster structure. A conditioning context of say the four nearest causal neighbors places all color changes in a non-uniform context. While sparse, image 1(c) may or may not be structured. If each “star” in the field exhibited significant extent, it would be considered structured, However, if each “star” contained only a single pixel, this image would be unstructured. While the methods of this paper are applicable to all sparse images, they are most effective on sparse images that exhibit some raster structure.

When coding sparse images, the key problem is to code the space between features as quickly as possible while compressing both features and non-features alike as compactly as possible. The first contribution of this paper is a model for switching back and forth

between two-dimensional coding for features and one-dimensional coding for non-features. This model naturally leads to an adaptive run length code that is inherently embedded in a two-dimensional setting. Because of this embedding, arithmetic coding and context models can be applied to each run-length code bit to further reduce the descriptive length of non-features.

The structure of the paper is as follows. First, the applicable prior art is examined. Next the skip-innovation model is introduced. Skip-innovation codes for use with this model are then presented. A context model for compressing such codes is then described. Experimental results are given followed by conclusions.

2. Review of Sparse Image Coding Techniques

One dimensional run-length coding was first applied to sparse-images in 1959[1]. Two-dimensional extensions to this basic technique led to the CCITT Group 4 Fax standard[2]. These early constructions were not adaptive and were tuned for images of a specific size.

A major advance came with the application of multiple context adaptive arithmetic coding to compression of black and white images[3]. Very early, it was recognized that more attention should be paid to accelerating the coding path of the most probable symbol (mps) even at the expense of the least probably symbol (lps). The most important developments along this line have been the skew-coder[3], the Q-coder[4], and most recently the Z-coder[5]. However, even with fastest possible mps path, arithmetic coding remains an inherently serial task. This is especially an issue for software implementations.

Accordingly, significant attention has been focused on so called “low complexity” approaches. One of the tenants of the low complexity approach has been the use of Golomb codes[6] to represent contiguous runs in the same context. This basic idea has been refined by techniques for adaptively changing the Golomb parameter, *m*, within a single run. The *block*-MELCODE of the proposed JPEG-LS standard[7] is the best known example of this technique.

As long as the average run length within the same context is long enough, the compression efficiency of Golomb codes and arithmetic codes is indistinguishable. However, once a Golomb run mode has been entered, it is difficult to take advantage of any changes in the coding context during a run. Runs are continued until the next lps occurrence which typically occurs in a different coding context. This results in context mixing and loss of compression efficiency.

Because adaptive Golomb codes cannot match the compression performance of arithmetic codes, recent work has focused on an accelerated arithmetic coding mode for contiguous symbol runs in the same context. This technique was first proposed in the context of the QM coder[8]. More recently a similar technique for accelerating the skew-coder has been suggested[9].

The basic idea is to scan ahead to obtain the length of contiguous runs of uniform context. Groups of mps’s are then coded within the extended uniform context. The length of each group is the number of occurrences necessary to force an mps renormalization or the residual if it is less than that necessary to force the next renormalization. Obtaining

the length of each group, especially the initial and residual groups, requires a division in the Q-coder. In the skew-coder this can be done with shifting and masking.

3. The Skip-Innovation Model

The first model for accelerated coding of sparse images was proposed in the context of palette images[10]. There it was recognized that whereas acquisition of image features largely takes place in uniform contexts, coding of previously acquired features largely takes place in non-uniform contexts.

To take advantage of this characteristic, a new decision is introduced into the coding model. When a uniform context is encountered, its length is determined and a decision is coded as to whether or not it can be skipped entirely. Runs that cannot be skipped are coded in the normal unary fashion. For example the two black features on Figure 2 are separated by a run of seven uniformly white coding contexts. Using the skip model this run is coded as a single affirmative skip decision.

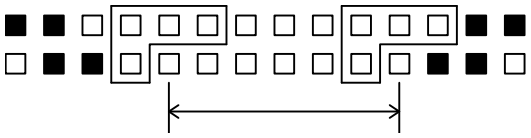

Figure 2
Skip

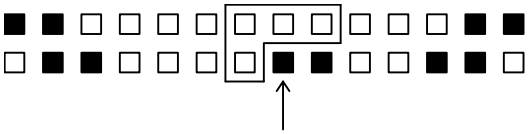

Figure 3
Innovation

Skip failures are the result of new features needing introduction to the model. The position of these new features, or *innovations*, is the information that must be conveyed by the coder. The example of Figure 3 shows an innovative central feature located four pixels into a contiguous context of length seven.

This paper advances on the skip plus unary model by proposing binary coding for innovations. The combined codes, called skip innovation codes, are described next.

4. Skip-Innovation Codes

Skip innovation codes are related to Golomb codes. The skip decision, *S*, can be viewed as the magnitude or unary portion of the code, the innovation, *I*, as the binary portion. The length of the unary portion of the code is always one. The basic length of the binary portion is the ceiling of the base two logarithm of *S*. As with Golomb Codes, the basic length of *I* can be significantly reduced if *S* is not a power of two. The procedure used for constructing skip-innovation codes is as follows:

- Count the number, *S*, of uniform contexts that occur before the next occurrence of a non-uniform context.
- Counting no more than *S*, count the number of pixels, *I*, to be coded whose value is identical to that populating the coding context.
- If $I = S$, encode a one, otherwise encode a zero and:
- Determine $D = \lceil \log_2(S) \rceil$, the number of binary digits required for a maximal *I*.
- Form a *D* digit binary representation of *I*.
- For each digit of the binary representation of *I* starting from the most significant:

- Determine the minimum value *T* that would be encoded if that digit took on a value of one and previous digits took on their previously encoded values.
- If $T < S$ encode the digit.

The codes generated by the aforementioned procedure for values of *S* and *I* up to seven are shown in Table 1.

I	*S*=1	*S*=2	*S*=3	*S*=4	*S*=5	*S*=6	*S*=7
0	0	00	000	000	0000	0000	0000
1	1	01	001	001	0001	0001	0001
2		1	01	010	0010	0010	0010
3			1	011	0011	0011	0011
4				1	01	010	0100
5					1	011	0101
6						1	011
7							1

Table 1
Skip-Innovation Codes for *S* = 1–7

For *m* not equal to a power of two, Golomb assigned the shorter binary sequences to shorter run lengths. Perhaps somewhat counterintuitively, the shorter skip-interval codes are assigned to longer runs. The first reason for this is obvious. The most frequent value for *I* is *S*, representing the lack of innovation. The second reason is more subtle.

Certain irregular or low slope features may be untrackable by a reasonably sized context model. The smaller the context model, the more likely a feature is to fall into this category of *pseudo-innovations*. For example, on Figure 4 the pixel labeled with a arrow is a pseudo-innovation. It is connected to a larger feature already known by the model but the model is too small to make a local determination. In this case, *S* is six and *I* is five resulting in the *SI* code of 011, one bit shorter than the basic code length of $D+1$.

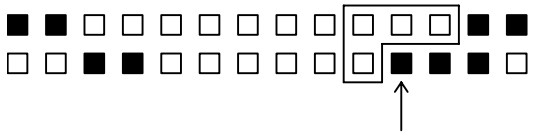

Figure 4
Pseudo-Innovation

Note that the *SI* code generation mechanism is extremely simple and efficient. In fact, it was originally chosen just for these properties. Only after several failed attempts at improvement was it recognized that skewing the code distribution towards *S* was a natural way to take advantage of the increased likelihood of pseudo-innovations appearing near *S*.

5. Context Models

Since skip-interval codes are part the image model, adaptive airthmetic coding can be used to further match them to the source material. A convenient context for coding *S* is *D*, previously calculated to determine the maximum possible number of binary code digits. To the extent that skips of various sizes are not uniformly distributed, using *D* as a context model for *S* can reduce the overall code string length.

Often, multiple innovations are located in a failed skip. Due to the structure of the *SI* codes, *I* will contain multiple leading zeros when the distance between innovations is substantially less than the skip length. The following context model can be used to capture this structure:

- Allocate one context for each bit of the maximum possible skip length.
- Designate one additional context the lumped context.
- Initialize a variable, ONE_SEEN to zero.
- From the most significant bit position of *I* to the lowest:
  - Code the bit under its positional context if ONE_SEEN is zero and under the lumped context otherwise.
  - If the coded bit is a one, set ONE_SEEN to one.

On black and white documents, the average black run often differs substantially from the average white run. Therefore it is useful to double number of contexts used for coding the *SI* bits.

6. Mixing *SI* Codes and Unary Codes

SI codes are designed for use as an alphabet extension mechanism in line oriented image codes. Such extensions allow a coder to switch between normal pixel at a time, or *unary*, coding and run length coding. The *SI* mechanism differs from conventional one-dimension run length coding in that it is inherently embedded in a two-dimensional coding process. *SI* never codes information in more than one context and therefore does not lose any of the benefit of any two-dimensional model that may be in use.

The two-dimensional nature of *SI* creates one subtlety that may not be immediately apparent. A typical one-dimensional run length code, in addition to encoding a run of identical pixels, also imparts some information about the pixel immediately following a coded run. The additional information imparted is that the following pixel is different from the coded run of pixels. On a black/white image this additional information is enough to completely determine the following pixel's value.

The *SI* mechanism is slightly different in that it only imparts information about the following pixel when a skip failure has occurred. The pixel immediately after a successful skip may or may not be of the same color as the just skipped run. The follow procedure shows how a decoder intermixes *SI* and unary codes on a single scanline of a black/white image:

- For each pixel location on the scanline
  - If the current pixel's coding context is non-uniform
    - Decode the pixel in the conventional unary fashion.
    - Advance the current pixel pointer one location.
  - Otherwise
    - Decode a skip-innovation code
    - If $I = S$, skip forward *S* pixel locations filling skipped pixels with the current color.
    - Otherwise:
      - Skip forward *I* locations filling skipped pixels with the current color.

- Fill the current pixel using the information that it is different from the current color and advance the current pixel pointer one column.

7. Experiments

7.1 Behavior of Skip-Innovation Codes on Highly Structured Material

A set of four synthetic CCITT-sized black and white images was devised to evaluate the performance of skip innovation coding on large, smooth features. The first of these images is uniformly white, the second is composed of several randomly oriented lines, the third, randomly oriented edges and the fourth, ellipsoidal edges.

Two versions of the depth one Piecewise-Constant codec[10] were then modified to use skip-innovation codes. The first of these uses a ten pel neighborhood context with arithmetic coding and is designed to closely match the default behavior of the JBIG codec. The second uses a four pel template without arithmetic coding and is designed to match the complexity of the JPEG-LS codec. In the following discussion they are designated SI and SI-jls respectively.

Image	SI	JBIG	QM	jpeg-ls	SI-jls
white	21	22	24	343	292
lines	1,429	1,920	1,869	11,729	4376
edges	1,348	1,687	1,646	10,714	3705
circles	1,037	1,135	1,239	8,017	3064
Total	**3,835**	**4,764**	**4,778**	**30,803**	**11437**
Time	**0.7**	**4.1**	**6.0**	**2.3**	**0.5**

Table 2
Highly Structured B/W Images

In Table 2, SI and SI-jls are compared against JBIG, JBIG without typical prediction (designated QM) and JPEG-LS. Each table column contains compressed file sizes (bytes) with compression times comprising the final row. The results show that SI is a much more robust acceleration mechanism than JBIG typical prediction. Typical prediction works well on the uniformly white image where all scanlines are identical but fails completely on the randomly oriented features.

Perhaps somewhat surprising is that SI compresses ~20% better than JBIG. This is fairly impressive since all the savings must come from the two uniform contexts. JBIG places ~2,400 bytes in the uniform contexts, so SI compresses them ~38% better.

The adaptive golomb code used by JPEG-LS is not competitive on these structured images. The SI-jls results show what could be expected if skip interval codes were used as the alphabet extension mechanism JPEG-LS*.

7.2 Adversarial Behavior of SI codes

Innovation with no extent is the adversary to the skip-innovation model. So for the next experiment, CCITT-sized “star fields” of varying density were synthesized using a noise plus quantization process. At the highest “star” density approximately half the image information resides in the uniform context of a ten pel neighborhood. The

* To exactly match the SI-jls results, JPEG-LS would also have to to use slightly modified predictive codes that take advantage of the limited number of colors in the image. However, in this experiment the number of redundant predictive bits emitted by JPEG-LS is not very significant.

approximate number of “stars” in each field is shown in the first column of Table 3. Coding results for the five methods used in the previous experiment are shown in subsequent table columns.

Features	SI	JBIG	QM	jpeg-ls	SI-jls
368,953	229,862	231,923	231,910	318,778	343,842
91,613	82,881	82,136	82,136	104,570	127,619
21,068	25,079	24,767	24,760	30,049	36,554
8,987	12,202	11,935	11,944	14,429	17,087
1,695	2,854	3,092	2,898	3,389	3,755
107	260	348	260	547	518
Time	**4.6**	**8.2**	**9.0**	**5.0**	**3.7**

Table 3
Star Fields

The surprising result, at least to the author, is that the compression performance of SI is virtually indistinguishable from the QM coder. This indicates that SI can be used as a general purpose acceleration model. And since it is model based, SI can be used with any arithmetic coder. The down side is that the use of SI changes the coded bitstream, so unlike coder specific speedup mechanisms its use is not optional.

The compression performance of the non-arithmetic coding SI variant is approximately 20% worse than JPEG-LS. The difference is due to the inherent two dimensional nature of the SI mechanism. A one-dimensional model is actually better matched to this type of image. Once in run mode, JPEG-LS pays no attention to the surrounding context so its average run length can be longer. SI-jls makes more unary decisions which without arithmetic coding provide no compression. Fortunately, images composed of isolated single pixel features are relatively rare in practice.

7.3 CCITT Fax Documents

Table 4 shows compression results for the CCITT Fax documents. Remarkably, SI compression is almost 2% better than JBIG. This despite the fact that the JBIG probability estimator is tuned for these images and despite the fact that only about 20% of the information in these images lies in uniform contexts.

ccitt#	SI	JBIG	jpeg-ls	SI-jls
1	12,675	12,884	35,840	21,829
2	7,726	8,008	30,439	13,221
3	19,494	20,052	71,211	40,110
4	48,461	49,039	126,450	84,595
5	22,647	23,272	73,769	42,589
6	11,493	11,764	51,664	24,983
7	51,085	52,306	133,423	77,349
8	12,932	13,288	55,053	25,152
Total	**186,513**	**190,613**	**577,849**	**329,828**
Time	**3.3**	**9.5**	**7.3**	**2.5**

Table 4
CCITT Fax Reference Documents

The speed up achieved by SI over JBIG (with typical prediction) is ~3:1. To the author’s knowledge this is the first published experimental data for any sparse image arithmetic coding acceleration mechanism. However, coder specific acceleration methods should be capable of similar results.

Some of the SI-jls improvement over JPEG-LS can be attributed to redundant

JPEG-LS predictive bits. However, the SI-jls results do show that relatively simple changes would more closely match JPEG-LS to structured images. While not as good as Group 4 fax, the compression achieved by SI-jls is significantly better than that achievable by one dimensional methods.

7.4 PWC Palette Image Corpus

Table 5 shows results for compressing the Piecewise-Constant palette image corpus[11] with three different versions of the Piecewise-Constant Image Model. PWC uses edge maps to determine where color changes occur in palette images and can be accelerated with the skip innovation model.

Column PWC shows the performance of the original streaming version of PWC. Column PWC-S shows the performance of a second version that incorporated skips but used unary coding for innovations. PWC-SI is a recent version of the PWC coder that uses the complete skip-innovation model.

The compression gain shown in the table is to a significant extent due to other improvements. However the performance gain from left to right is largely attributable to sparse image acceleration.

Image	PWC	PWC-S	PWC-SI
benjerry	2,418	2,473	2,399
books	8,616	8,719	8,153
ccitt01	12,881	12,685	12,683
cmpndn	54,780	53,725	53,021
cmpndu	40,917	40,354	39,556
flax	107	171	142
gate	15,784	15,671	15,282
music	755	721	696
netscape	10,786	10,839	10,533
pattern	1,178	1,096	1,099
sea_dusk	646	696	678
stone	4,268	3,982	3,637
sunset	52,923	53,173	51,623
winaw	11,459	11,700	10,853
yahoo	4,443	4,461	4,350
Total	**221,961**	**220,466**	**214,705**
Time	**8.0**	**4.1**	**2.4**

Table 5
Adding Sparse Image Models to Streaming PWC

7.5 High Speed Compression/Decompression of Vector-based Material

Metric	PNG	PWC
Comp. Bytes	33,614	13,558
Comp. Rate	29.2:1	72.4:1
Encode (sec)	1.1	0.6
Decode (sec)	0.6	0.6
Enc/Dec (sec)	1.7	1.2

Table 6
Dynamic Content Examples

Because the skip-innovation coding model is symmetric it lends itself to compression of dynamically created content of the type commonly used on the Internet. Such content is often synthetic or composite and as such is often both sparse and highly structured. Examples of such material include charts, figures, maps, clip art, page backgrounds, and user-interface elements.

In Table 6 an example from each of these classes was compressed using PWC-SI and PNG, its closest competitor in terms of compression rate and efficiency. PWC-SI's compression rate is about two and a half times better than that of PNG. Remarkably, PWC-SI matches PNG's extremely fast decode speed on encode as well.

7.6 Comparison of *SI* and JBIG2

Finally, it is interesting to compare *SI* with the emerging JBIG2[12] standard. As the first row of Table 7 shows, the symbol coding techniques used in JBIG2 perform quite well on textual material such as that in the CCITT Fax images. The first column of the table shows JBIG2's lossless compression of all eight images. With the addition of loss in column two, symbol coding becomes even more successful resulting in significantly better compression. Column three of the table restates *SI*'s compression and column four shows *SI*'s compression of JBIG2's lossy image. The second row of the table shows encode/decode times. This particular JBIG2 implementation[13] is very slow compared with *SI*. It is possible that *SI* or a similar mechanism could be used to significantly accelerate JBIG2, especially on decode.

	JBIG2	JBIG2+L	SI	SI+L
Size	148,376	103,322	186,513	167,122
Time	40/13.9	62/12.7	3.3/3.3	25/3.3

Table 7
JBIG2 on the CCITT Fax Documents

7.7 Experimental Notes

The arithmetic coder used in the experiments is the carry-free binary arithmetic coder written by Don Speck[14]. The coder's performance has been enhanced by replacing the multiply/divide operation with a table lookup and multiply. At 50% probability the table-lookup carry free coder is approximately 10% slower than the QM coder.

All compression times were obtained using a 200 MHz Intel Processor running Microsoft Windows 98 Second Edition. The color feedback PWC coder used in the experiments is available at http://www.caravian.com. PWC compression times were obtained using the "-flip" option of the coder. Older versions of the PWC coder are obtainable via email request from the author.

JBIG results were obtained using the JBIGKIT[15]. JPEG-LS results were obtained using the HP Labs LOCO-I implementation[16]. LOCO times were as reported by the program. PWC and JBIG results are time elapsed. PNG compression times were simulated using efficient command line versions of Zip[17] and Unzip[18].

8. Conclusion

The skip-innovation model has been shown to be remarkably effective for accelerating arithmetic coding of sparse images. The model is robust against adversarial images and on images exhibiting two-dimensional structure it can significantly improve compression. On the CCITT fax documents, SI provides a 2% compression improvement and 3x speedup when measured against JBIG1.

Skip-innovation codes were developed and shown to be a two dimensional variant of adaptive Golomb codes. A variable parameter, *S*, plays a similar role to the Golomb parameter *m*. In contrast to *m*, *S* is instantaneously and inherently adapted to any two dimensional structure exhibited by the source. A low complexity method of using *SI* codes to match JPEG-LS to structured material was demonstrated. While the compression performance does not rival arithmetic alternatives, the results suggest the application domain of JPEG-LS might be expanded by using *SI* as an alphabet extension mechanism.

9. References

[1] J. Capon, "A probabilistic model for run-length coding of pictures", *IRE Transactions on Information Theory*, IT-5, pp. 157-163, December 1959.

[2] R. Hunter and A. H. Robinson, "International Digital Facsimile Coding Standard", *Proceedings of the IEEE*, 68(7) pp. 854-867, 1980.

[3] Glen G. Langdon, Jr. and Jorma Rissanen, "Compression of Black–White Images with Arithmetic Coding", *IEEE Transactions on Communications*, Vol. COM–29(6), pp. 858–867, (June 1981).

[4] W. B. Pennebaker, J. L. Mitchell, G.G. Langdon, Jr., R. B. Arps, "An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder", *IBM Journal of Research and Development*, Vol. 32, Number 6, pp. 717-726, November 1988.

[5] Leon Bottou, Paul G. Howard, and Yoshua Bengio, "The Z-Coder adaptive binary coder", *Proceedings Data Compression Conference*, pp. 13-22, March 1998, IEEE Press, Los Alamitos, California.

[6] S. W. Golomb, "Run-Length Encodings", IEEE Transactions on Information Theory, Vol. IT-12, pp. 399-401, July 1966.

[7] Marcelo Weinberger, Gadiel Seroussi, Guillermo Sapiro, and Michael W. Marcellin, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS", Hewlett-Packard Computer Systems Laboratory, HPL-98-193, November 1998.

[8] W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, New York, New York, 1993.

[9] Erik Ordentlich, David Taubman, Marcelo Weinberger, and Gadiel Seroussi, "Memory Efficient Scalable Line-based Image Coding", *Proceedings Data Compression Conference*, March 1999, pp. 218-227, IEEE Press, Los Alamitos, California.

10 Paul J. Ausbeck Jr, "A Streaming Piecewise-Constant Model", *Proceedings Data Compression Conference*, March 1999, IEEE Press, Los Alamitos, California.

[11] Paul J. Ausbeck Jr, "Context Models for Palette Images", *Proceedings Data Compression Conference*, March 1998, IEEE Press, Los Alamitos, California.

[12] Howard, P.G.; Kossentini, F.; Martins, B.; Forchhammer, S.; Rucklidge, W.J., The Emerging JBIG2 Standard, IEEE Transactions on Circuits and Systems for Video Technology,vol.8, (no.7), IEEE, Nov. 1998. p.838-48.

[13] AT&T Labs DjVu Codec, http://www.djvu.com/

[14] Don Speck, "Local Activity Level Classification Model for Continuous–tone Coding", document N198 submitted to ISO/IEC JTC1/SC29/WG1 June 29, 1995.

[15] Markus Kuhn, Version 0.9 of the JBIG–KIT, available via anonymous ftp at ftp.informatik.uni–erlangen.de/pub/doc/ISO/JBIG/jbigkit-0.8.tar.gz.

[16] HP Labs LOCO-I/JPEG-LS Home Page, http://www.hpl.hp.com/loco.

[17] Mark Adler, Richard B. Wales, Jean-loup Gailly, Onno van der Linden and Kai Uwe Rommel, *Zip*, http://www.cdrom.com/pub/infozip/Zip.html.

[18] Greg Roelofs, *Unzip*, http://www.cdrom.com/pub/infozip/UnZip.html.